REFERENCES

Abkowitz, M. D.; "An Analysis of the Commuter Departure time decision", Transportation, Vol. 10, 1981.

Ahuja, R. K., Melhorn, K., Orlin, J. B. and Tarjan, R. E.; "Faster algorithms for the shortest path problem", Journal of the Association for Computing Machinery, Vol. 37, No. 2, pp. 213-223, April 1990.

Al-Deek, H., Martello, M., May, A.D. and Sanders, W. M.; "Potential Benefits of In-vehicle Information Systems in a Real life Freeway Corridor Under Recurring and Incident Induced Congestion", Proceedings of the First International Conference on Application of Advanced Technologies in Transportation Engineering, San Diego, CA, Feb. 1989, pp. 500-505

Arnott, R., de Palma, A. and Lindsey, R.; "Does Providing Information to Drivers Reduce Traffic Congestion ?", paper presented at the 69th annual meeting of the Transportation Research Board, Washington D.C., Jan. 1990.

Barcelo, J.; "A Simulation Tool for Evaluating Road Transport Informatics" presented at the 70th annual meeting of the Transportation Research Board, Washington, D.C., 1991.

Beckmann, M. J., McGuire, C. B., and Winsten, C. B.; "Studies in the Economics of Transportation", Yale University Press, New Haven, Conn., 1956.

Belcher, P. and Catling, I.; "Electronic Route Guidance by Autoguide: The London Demonstration", Traffic Engineering and Control, Vol. 28, No. 11, 1987.

Belcher, P. and Catling, I.; "AUTOGUIDE - Electronic Route Guidance for London and the U.K.", Second IEEE Int. Conf. on Road Traffic Monitoring, London, February 1989.

Bellman, R.; "On a routing problem", Quarterly of Applied Mathematics, Vol. 16, pp. 87-90, 1958

Ben-Akiva, M., Bergman, M. J., Daly, A. J. and Ramaswamy, R.; "Modelling

Inter-urban Route choice Behavior", 9th International Symposium on Transportation and Traffic Theory, VNU Science Press, pp. 299-330, 1984.

Ben-Akiva, M., De Palma, A. and Kanaroglou, P.; "Analysis of Peak-period Congestion with Elastic Demand", Transportation Science, Vol.21, No. 3, 1986.

Benshoof, J. A.; "Characteristics of Drivers' Route Selection Behavior", Traffic Engineering and Control, pp. 606-609, 1970.

Blum, Y. and van Aerde, M.; "A Comprehensive Approach to In-vehicle Route Guidance using Q-Route", presented at the 68th annual meeting of the Transportation Research Board, Washington, D.C., Jan. 1989.

Boyce, D.; "Route Guidance Systems for Improving Urban Travel and Location Choices", Transportation Research, Vol. 22A, No. 4, 1988.
Chang, G. L., Mahmassani, H. S. and Herman, R.; "A Macroparticle Traffic Simulation model to Investigate Peak-period Commuter decision dynamics", Transportation Research Record 1005, pp 107-120, 1985.

Boyce, D., Ran, B. and LeBlänc, L. J.; "Dynamic User-optimal Traffic Assignment: A New Model and Solution Technique", forthcoming in Transportation Science, 1991.

Cantarella, G. E., Casscetta, E. and DiGangi, M.; "Some Applications of a Doubly Dynamic Assignment Model", presented at the 70th annual meeting of the Transportation Research Board, Washington, D.C., 1991

Chang, G. L.; "Departure time Decision Dynamics in the Urban Transportation Network", Ph.D thesis, University of Texas at Austin, 1985.

Chang, G. L., Mahmassani, H. S. and Herman, R.; "Macroparticle Traffic Simulation Model to Investigate Peak-period Commuter Decision Dynamics", Transportation Research Record 1005, pp. 107-121, 1985.

Carey, M.; "Optimal Time-varying flows on Congested Networks", Operations Research, Vol. 35, No. 1, pp. 58-69, 1987.

Clarke, S., Krikorian, A. and Rausen, J.; "Computing N best loopless paths in a

Network", Journal of Soc. and Industrial Application of Mathematics, Vol. 11, pp. 1096-1102, 1963.

Cosslet, S.; "The Trip Time Decisions for Travel to Work by Automobile", Urban travel demand forecasting project, Phase-I final report, Vol. V, Institute of Transportation Studies, Univ. of California, Berkeley, 1977.

Dafermos, S. C. and Sparrow, F. T.; "The Traffic Assignment Problem for a General Network", Journal of Research of the National Bureau of Standards, Vol. 73B, pp. 91-118, 1969.

Daganzo, C. F. and Sheffi, Y.; "On Stochastic Models of Traffic Assignment", Transportation Science, Vol. 11, No. 3, pp. 253-274, 1977.

De Palma, A., Ben-Akiva, M., Lefevre, C. and Litinas, N.; "Stochastic Equilibrium Model of Peak Period Traffic Congestion", Transportation Science, Vol. 17, pp 430-453, 1983.

Derzko, N. A., Ugge, A. J. and Case, E. R.; "Evaluation of a Dynamic Freeway Model using Field Data", presented at the 62nd annual meeting of the Transportation Research Board, Washington, D.C., 1983

Dantzig, G. B., "All Shortest routes in a Graph", In "Theory of Graphs", Edited by P. Rosenstiehl, Gordon & Breach, NY, pp. 91-92, 1967.

Denardo, E. V. and Fox, B. L.; "Shortest Route Methods: 1. Reaching, Pruning and Buckets", Operations Research, Vol. 27, pp. 161-186, 1979

Dewar, R. E.; "In-vehicle Information and Driver Overload", International Journal of Vehicle Design, Vol. 9, No. 4/5, pp 557-564, 1988.

Dial, R.; "Algorithm 360. Shortest Path Forest with Topological Ordering", Communications of the ACM, Vol 12, pp. 632-633, 1969.

Dial, R., Glover, F., Karney, D. and Klingman, D.; " A Computational Analysis of Alternative Algorithms for finding Shortest Path Trees", Networks, Vol. 9, pp. 215-248, 1979.

Dijkstra, E.; "A note on two problems in connection with graphs", Numeric. Mathematiche., Vol. 1, pp. 269-271, 1959.

Dingus, T. A., Antin, J. F., Hulse, M. C. and Wierwille, W. W.; "Attentional Requirements of an Automobile Moving-map Navigation System", Transportation Research, Vol 23A, No. 4, pp 301-315, 1989.

Drouin, M., Abou-Kandil, H., Dib, G. and Bertrand, P.; "A New Approach for Real-time Control of Urban Traffic Networks", Control in Transp. Systems, Proc. of the IFAC/IFIP/IFORS Conference, Baden-Baden, Fed. Rep. of Germany, 1983.

Dudek, C. L., Weaver, G. D., Hatcher, D. R. and Richards, S. H.; "Field Evaluation of Messages for Real-Time Diversion of Freeway Traffic for Special Events", Transportation Research Record 682, pp.37-45, 1978.

Dudek, C. L. and Huchingson, R. D.; "Studies on Highway Advisory Radio Messages for Route Diversion", Transportation Research Record 904, pp.4-9, 1983.

Florian, M. (ed); "Traffic Equilibrium Methods", Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 1976.

Floyd, R. W.; "Algorithm 97 : Shortest paths", Communications of ACM, Vol. 5, p. 345, 1962.

Ford, L.; "Network Flow Theory", The Rand Corporation, 1956.

Fox, B. L.; "Data structures and Computer Science Techniques in Operations Research", Operations Research, Vol. 26, No. 5, 1978.

French, R. L; "The Evolving Roles of Vehicular Navigation", Navigation: Journal of the Institute of Navigation. Vol. 34, No. 3, pp.212-228, 1987.

Fuji, H.; "The CACS Project", Annual Meeting of the Transportation Research Board, Washington, D.C., January 1986.

Gilsinn, J. and Witzgall, C.; "A performance Comparison of Labelling Algorithms for Shortest Path Trees", NBS Technical Note 772, U.S. Department of Commerce, 1973.

Glover, F., Glover, R. and Klingman, D.; "Computational Study of Improved

Shortest path Algorithms", Networks, Vol. 14, pp. 25-37, 1984.

Glover, F., Klingman, D. and Phillips, N.; "A New Polynomially-bounded Shortest Path Algorithm, Operations Research, Vol. 33, pp. 65-73, 1985.

Goto, S., Ohtusi, T. and Yoshimura, T.; "Sparse matrix techniques for the Shortest paths problem", IEEE Transactions on Circuits and Systems, pp. 752-757, 1976.

Goto, S. and Sangiovanni-Vincentelli, A.; "A New Shortest-path Updating Algorithm", Networks, Vol. 8, pp. 341-372, 1978.

Greenshields, B. D., "A Study of Traffic Capacity", Highway Research Board Proceeding, Vol. 14, pp. 448-474, 1934

Haeussermann, P.; "On-board Computer System for Navigation, Orientation, and Route Optimization", Technical Paper Series 840485, Society of Automotive Engineers, 1984.

Hendrickson, C. and Plank, E.; "The flexibility of Departure times for Work trips", Transportation Research, Vol. 18A, pp. 25-36, 1984.

Herman, R., Montroll, E. W., Potts, R. B. and Rothery, R. W.; "Traffic Dynamics: Analysis of Stability in Car-following", Journal of the Operation Research Society, Vol. 7, pp. 86-106, 1959.

Janson, B. N.; "Dynamic Traffic Assignment for Urban Road Networks", presented at the Transportation Research Board Annual Meeting at Washington, D.C., Jan 1990.

Jayakrishnan, R.; "Application of a Dynamic Modelling Framework to the Analysis of Perturbations in a Congested Traffic System", M.S. Thesis, University of Texas at Austin, 1987.

Jeffery, D. J.; "The Potential Benefits of Route Guidance", TRRL Research Report LR 997, 1981

Jeffery, D. J., Russam, K., and Robertson, D. I.; "Electronic Route Guidance by AUTOGUIDE: Research Background", Traffic Engineering and Control, Vol. 28, No. 10, 1987.

Jeffery, D. J.; "Driver Route Guidance Systems: State of the Art", prsented at the Internatiuonal Symposium on Telematics-transportation and Spatial Development", Hague, Netherlands, April, 1988.

Johnson, D. B.; "Efficient Special Purpose Priority Queues", Proceedings of the 15th annual Allerton Conference on Communications, Control and Computing, University of Illinois, pp. 1-7, 1977.

Johnson, E. L.; "On Shortest Paths and Sorting", Proceedings of ACM National Conference, pp. 510-517, 1972.

Jones, E. G., Mahmassani, H. S., Herman, R. and Walton, C. M.; "Travel Time Variability in a Commuting Corridor : Implications for Electronic Route Guidance", presented at the first Intl. Conf. on Application of Advanced Technologies in Transportation Engg, San Diego, CA, 1989.

Jovanis, P. P. and Kitamura, R.; "User Perceptions and Safety Implications of In-vehicle Navigation Systems", presented at the VNIS Conference, Toronto, September 1989.

Kanafani, A.; "Transportation Demand Analysis", Mc-Graw Hill, 1983.

Koutsopoulos, H. N. and Lotan, T.; "Motorist Information Systems and Recurrent Traffic Congestion : A Sensitivity Analysis of Expected Benefits", presented at the VNIS Conference, Toronto, Sept. 1989.

Leboeuf, J., Tajima, T. and Dawson, J.; "A Magnetohydrodynamic Particle Code for Fluid Simulation of Plasmas", Journal of Computational Physics, Vol. 31, No. 3, pp. 279-408.

Leonard, D. R., Gower, P. and Taylor, N. B.; "CONTRAM : Structure of the Model", Research Report 178, Transport and Road Research Laboratory, Crowthorne, U.K., 1989.

Lighthill, M. J. and Whitham, G. B., "On Kinematic Waves", Proceedings of the Royal Society of London, Vol. 229A, pp. 317-345, 1955

Mahmassani, H. S. and Chang, G. L.; "Experiments with Departure time Choice Dynamics of Urban commuters", Transportation Research, Vol. 20B, No. 4, pp. 297-320, 1986.

Mahmassani, H. S. and Herman, R.; "Dynamic User Equilibrium, Departure Times and Route Choice in Idealized Traffic Arterials", Transportation Science, Vol. 18, pp. 362-384, 1984.

Mahmassani, H. S., Herman, R., Walton, C. M., Jones, E. G., Baaj, M. H., and Jayakrishnan, R.; "Travel time characteristics and Opportunities for Real-time Information Systems in an Urban Commuting Corridor", Research Report GM-1989-F, Center for Transportation Research, Univ. of Texas at Austin, 1989.

Mahmassani, H. S. and Jayakrishnan, R.; "Dynamic Analysis of Lane-closure Strategies", Journal of Transportation Engineering, ASCE, Vol. 114, No.4, pp. 476-496, 1988.

Mahmassani, H. S. and Jayakrishnan, R.; "System Performance and User Response under Real-time Information in a Congested Traffic Corridor", forthcoming in the Journal of Transportation Research (tentatively, Vol. 25, No. 5), 1990.

Mahmassani, H. S. and Stephan, D. G.; "Experimental Investigation of Route and Departure time dynamics of Urban Commuters", Transportation Research Record (in press), 1988.

Manheim, M. L., "Fundamentals of Transportation Systems Analysis", Vol.1: Basic Concepts, MIT Press, 1979.

Mauro, V.; "Evaluation of Dynamic Network Control: Simulation Results Using NEMIS urban Microsimulator", presented at the 70th annual meeting of the Transportation Research Board, Washington, D.C., 1991.

May, A.D., and Bonsall, P. W.; "Recommended Methodology for Evaluation of Route Guidance in London and the Potential Comparability with Berlin", Contractor Research Report 130, Transport and Road Research Laboratory, Crowthorne, U.K., 1989.

Merchant, D. K. and Nemhauser, G. L.; "A Model and an Algorithm for the Dynamic Traffic Assignment Problems", Transportation Science, Vol. 12, No. 3, pp. 183-199, 1978.

Merchant, D. K. and Nemhauser, G. L.; "Optimality Conditions for a Dynamic

Traffic Assignment Model", Transportation Science, Vol. 12, No. 3, pp. 200-207, 1978.

Merchent, B. G.; "Concepts for Vehicle Route Guidance", Second IEE International Conference on Road Traffic Monitoring, London, 1989.

Minieka, E.; "Optimization Algorithms for Networks and Graphs", Dekker Inc., New York, 1978.

Moore, E.; "The Shortest Path through a Maze", Proc. Intl. Symposium on Theory of Switching, part II, 1959.

MVA Consultants; "Estimates of AUTOGUIDE Traffic Effects in London", Contractor Research Report 129, Transport and Road Research Laboratory, Crowthorne, U.K., 1989.

Okamoto, H.; "Advanced Mobile Information and Communication System (AMTICS)", presented at the International Sysmposium on Automotive Technology and Automation (ISATA), paper 89103, June 1989.

Papageorgiou, M.; "Automatic Control Concepts in Traffic Networks', Springer Verlag, 1985.

Papageorgiou, M. and Messer, A.; "Dynamic Network Traffic Assignment and Route Guidance via Feedback regulation", presented at the 70th annual meeting of the Transportation Research Board, Washington, D.C., 1991

Payne, H. J.; "Models of Freeway Traffic and Control", Simulation Council Proceedings Series, Vol. 1, Mathematical Models of Public Systems, ed. George A. Bekey, 1971.

Payne, H. J.; "FREFLO: A Macroscopic Simulation Model of Freeway Traffic", Transportation Research Record, No. 772, Transportation Research Board, 1979.

Perko, A.; "Implementation of Algorithms for K shortest Loopless Paths", Networks, Vol. 16, pp. 149-160, 1986.

Phillips, W. F.; "A New Continuum Model of Traffic Flow", Report DOT-RC-82018, Utah State University, Logan, Utah, 1979.

Richards, P. J.; "Shock Waves on the Highway", Operations Research, Vol. 4, pp. 42-51, 1956.

Ran, B. and Shimazaki, T.; "Dynamic User Equilibrium Traffic Assignment for Congested Transportation Network", presented at the International Symposium on Traffic and Transportation Theory, Tokyo, 1990.

Rosen, D. A., Mammano, F. J., and Favort, R.; "An Electronic Route Guidance System for Highway Vehicles", IEEE Transactions on Vehicular Technology, Vol.VT-19, 1970.

Sheffi, Y. and Powell, W. B.; "An Algorithm for the Equilibrium Assignment problem with Random Link times", Networks, Vol. 12, No. 2, pp. 191-207, 1982.

Sheffi, Y.; "Urban Transportation Networks : Equilibrium Analysis with Mathematical Programming Methods", Prentice Hall Inc., 1985.

Shibano, Y., Norikane, T., Iwai, T., and Yamada, M.; "Development of Mobile Communication System for RACS", Proceeding of Vehicle Navigation and Communication Systems Conference, Toronto, September 1989.

Shirazi, E., Anderson, S. and Stesney, J.; "Commuters' Attitudes Towards Traffic Information Systems and Route Diversion", presented at the 67th annual meeting of the Transportation Research Board, Washington, D.C., 1988.

Shier, D. R.; "On Algorithms for Finding the K Shortest Paths in a Network", Networks, Vol. 9, pp. 195-214, 1979.

Shladover, S. E.; "Roadway Automation Technology : Research Needs", presented at th 68th annual meeting of the Transportation Research Board, Washington, D.C., January 1989.

Simon, H.; "A Behavioral model of Rational Choice", Quarterly Journal of Economics", Vol. 69, pp. 99-118

Smith, J. C. and Russam, K.; "Some possible effects of Autoguide on Traffic in London", proceedings of the first VNIS Conference, Toronto, Canada, pp. 282-287, 1989

Smith, M. J. and Ghali, M. O.; "Dynamic Traffic Assignment and Dynamic Traffic Control", Transportation and Traffic Theory, ed. M. Koshi, Proceedings of the 11th International Symposium on Traffic and Transportation Theory, Japan, 1990

Smith, R. W.; "Attitudes towards Route Choice in Urban Transportation Systems", Damas and Smith Limited, Toronto, Canada, 1969.

Stopher, P. R. and Meyburg, A. H.; "Behavioral Travel Demand Models", Heath and Company, 1976.

Tagliacozzo, F. and Pirizo, F.; "Assignment Models and Urban Path Selection Criteria : Results of a Study of Behavior of Road Users", Transportation Research, Vol. 7, 313-329, 1973.

Tarjan, R. E.; "Data Structures and Network Algorithms", Monograph, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.

Teply, S.; "Network Travel Behavior, 'Levels of Choice' and Urban Transportation Management", Traffic Engineering and Control, February, 1982.

Tsuji, H. and Kawashima, H.; "Stochastic Models for Estimating the Effectiveness of a Route Guidance System", Control in Transportation Systems, Proc. of the IFAC/IFIP/IFORS Conference, Baden-Baden, Fed. Rep. of Germany, 1983.

Tsuji, H., Takahashi, R., Kawashima, H. and Yamamoto, Y.; "A Stochastic Approach for Estimating the Effectiveness of a Route Guidance System and its Related Parameters", Transportation Science, Vol. 19, No. 4, pp. 333-351, Nov. 1985.

van Aerde, M. and Yagar, S.; "Dynamic Integrated Freeway/Traffic Signal Networks : Problems and Proposed Solutions", Transportation Res., Vol. 22A, No. 6, pp 435-443, 1988.

van Aerde, M. and Yagar, S.; "Dynamic Integrated Freeway/Traffic Signal Networks : A Routing-based Modelling Approach", Transportation Res., Vol. 22A, No. 6, pp 445-453, 1988.

Van Grol, H. J. M. and Bakker, A. F.; "A Special Purpose Computer for Traffic Simulations", presented at the 70the annual meeting of the Transportation Research Board, Washington, D. C., 1991.

Van Vliet, D.; "Improved Shortest path algorithms for Transport Networks", Transportation Research, Vol. 12, pp. 7-20, 1978.

Van Vuren, T. and Catling, D.; "A multiple User Class Assignment Model for Route Guidance", presented at the 70th annual meeting of the Transportation Research Board, Washington, DC., 1991.

Von Tomkewitsch, R.; "ALI-SCOUT : A Universal Guidance and Information System for Road Traffic", 6th IEEE International Conference on Automotive Electronics, London, UK, October 1987.

Vythoulkas, P. C.; "Two Models for Predicting Dynamic Stochastic Equilibria in Urban Transportation networks", Proceedings of the International Symposium in Transportation and Traffic Theory, M. Koshi (ed), Elsevier Science Publishing, 1990.

Wachs, M.; "Relationships between Drivers' Attitudes towards Alternative Routes and Driver and Route Characteristics", Highway Research Record, No. 197, pp. 70-87, 1967.

Wardrop, J. G.; "Some Theoretical Aspects of Road Traffic Research", Proceedings of the Institution of Civil Engineers, Vol. II, no. 1, pp. 325-378, 1952.

Wie, B. W., Friesz, T. L. and Tobin, R. L.; "Dynamic User Optimal Traffic Assignment: A Control Theoretic Formulation", presented at the US-Italy Joint Seminar on Urban Traffic Networks, Capri, Italy, June 1989.

Yen, J.; "Finding the K shortest loopless paths in a network", Management Science, Vol. 17, pp. 712-716, 1971.

APPENDIX - A (Simulation Program)

```
      program
ma2b(input,output,tape1=input,tape2=output,tape4
    2        ,tape5,tape6,tape7,tape8,tape9,tape10
    3        ,tape21,tape22
    4        ,tape33
    5        ,tape51,tape52,tape61,tape71,tape72
             7
,tape83,tape91,tape92,tape93,tape94,tape90,tape95
    6        ,tape37,tape38,tape39)
c
c  this model looks at the dynamic generation of vehicles
c  going from suburban sectors to the cbd. for this purpose we
c  use the lebeouf, tajima and dawson's particle code method

c  for the movement of macroparticles.
c  the original plasma simulation code has been extensively

c  rewritten and has been extended for the simulation of
c  traffic flow in a network.
c
c  the network has to be acyclic with unidirectional links.
c  the links should be numbered in such a way that the
c  successive links should have higher numbers than the

c  preceding links.
c
c  vlg=local generation of traffic.
c  xl=average length of the roads in each cell.
c  n=total number of links in the system.
c  ni=xni=total number of vehicles in each cell.
c  ntto=total time of the simulation run in minutes.
c  tii=simulation time step (in minutes).

c  ntt=number of time intervals in the simulation run =ntto/tii
c  ti=initial time of generation in the link
c  tf=final time of generation in the link
c  stime=stating times from a link to the cbd
c  atime=arrival times to the cbd
c  vmax=max allowed velocity in the link
c  s=link length
c  cmax=max concentration of vehicles in the link=ni/xl
c  c=actual concentration of vehicles in the link=npar/xl
c  p=parameter in the velocity formula
c  mnum=# of macroparticles created in each cell (subr. partco)
c  xpar=position of the macroparticles (subr.partco)
      dimension r(10)
      dimension xl(1781),vlg(1781)
      dimension ni(1781),cmax(1781),vmax(1781),p(1781)
      dimension c(1781),coef(1781)
      dimension iflag(1781),gen(1781),npar(1781),nout(1781)
      real a1,a2,a3,a4,a5,a6
      dimension sig(1781,10)
      dimension wk(100),numday(100)
      dimension isec(69000),stime(69000),atime(69000)
      dimension trt(1781,3),tag(1781,3),vtr(1781,3)
                                   *
,sqsumt(1781,3),stdrt(1781,3),vlag(1781,3),ssum(1781,3)
   *  ,std(1781,3),vlage(1781,5),vlagl(1781,5),tagerly(1781,3)
   *  ,taglate(1781,3)
cvar  dimension tsd(1781,335,3)
      dimension ck(1781,660)
c d                    d i m e n s i o n
conpern(69000),conperd(69000),conpert(69000)
      dimension vtotal(3),nswch(4),dswch(4)
      common/fni1/cmax,vmax,p,eps,prob,vlg,xl,c,kk
      common/fni2/isec,stime,atime,iflo,j,mtnum,sd,ist,npar,nout
      common/fni4/iflag,gen,noutfix(1781),entry(1781)
c     the mtxj dimension is dependent on the maximum
capacity
c     of the links (i.e., total number in all the lanes
c     at any time).
c     mtqj capacity is the maximum number allowed in any
entry
c     queue.

common/fni3/mtxj(1781,335),mtqj(1781,128),mxj(335),myj(335)
      common/fni6/numsec(1),isect(1,1781)
      common/fni7/ratio(1,1781,3),nupuls(1,1781)
      common/fni13/time1(1,1781,3)
      common/fni14/time2(1,1781,3)
c        d
common/fni8/congesn(69000),congesd(69000),congest(69000),
cd    *condist(69000),cflag(69000)
      common/fni5/conglim(1781),distans(69000),qflag(69000)
     * ,nsw(69000),dsw(69000,4)
c     the dimension of tsd depends on the maximum number of
c     vehicles/particles created in the link.
cn    common/fni11/confrac(1781)
      common/fni10/s(1781),v(1781),nlanes(1781),nmov(1781)
      common/fni9/conct(1781),secont(1781),cfrac(1781)
      common/fni12/xpar(69000),tleft(69000),tqwait(69000)
      common/fni15/ndque(1781),anqr(1781),expquet(1781)
     *        ,nqaccum(1781),ndqr(1781,30)
      common/fni16/totalql(1781),totendq(1781),nqactiv(1781),
     *nenqact(1781),ntryq(1781),nenac1(1781),nenaflg(1781),
     *nqac1(1781)
      common/jay1/link(1781,8),klink(1781,7),statmpt(1781)
     *        ,inlink(1781,8),intooi(1781,676,3),intoo(1781)
      common/jay2/tiinow(69000),jdest(69000),jpath(69000,90)
     *        ,ribf(69000),icurnt(69000),nottin(69000)

common/jay3/info(69000),itag(69000),jsw(69000),nexlink(690
00)

common/jay4/startm,endtm,numcars,iseed,fracinf,ribfa,bound
      common/jay5/influx(7,1781),limflux(7,1781),cayvee(7)
```

```
common/jn1/ifwdarc(1779,2),npoint(661),idests(32),nreach(661)

common/jn2/iwork1(1779),iwork2(1779),iunod(1779),idnod(17
79)
     common/jn3/lastnod(32,661,12,2),dist(32,661,12),
     *          lastno1(32,661,12,2),lastk(32,661),kclos(32,661)
     dimension begint(10)
c
c    36 demand zones and upto 401 generation links per zone

c    are assumed in the common jzone.
c    it is also assumed that the node numbers fall below 661

c    and the zone numbers fall below 802
c
     common/jzone/izlins(36),izlin(36,401),zdem(36,36,10)

,izone(661),ioz(36),idz(36),ipz(36),npz(801),zfdem(36,36,10)
     * ,ildem(1781),ztdem(36,10),int,totimz(36)
     * ,expgenz(36),expgen(1781)
c
c
c    some key parameters of the simulation are set next.
c
c    set ipert=1, if any perturbation is to be simulated.
c    wst =  the workstart time (for sch.delay calculations)
c    set lastday=1, for a single day simulation.
c    set iflux=1, for flux limits from link to link.
c    set entrymx as the maximum allowable ramp entry rate

c     in vehicles per minute.
c    linkmax is the maximum number of links in an arc-chain,

c     currently set as 1.
c    eps     = minimum velocity (jam speed) in miles/minute
c    mtnum   = size of macroparticles.
c    densmax = maximum (jam) vehicle density allowed.

c    starttm = statistics only on vehicles that leave after
c     this start-up time.
c
c    l1 to l12 are the selected links, in any order, for which
c    the vehicle densities for every minute are printed.
c
     data iseed/123457/
     data ipert/0/
     data wst/80./
     data lastday/1/
     data iflux/1/
     data entrymx/40.0/
     data linkmax/1/
     data eps/0.1/
     data mtnum/1/
     data densmax/160.0/
     data starttm/10.0/
     data endtm/35.0/
     data kspstep/20/
     data l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12/
     *92,145,277,321,531,971,1681,1693,1711,1721,1731,1741/
c
c    set nr=10 for random number vector.
     nr=10
c    send the random number seed to the imsl routines.

     call mset(iseed)
c
c    entry queue exceed flag, and initial-route-copy-flag
     lffl=0
     initcop=0
c
c    fix up the simulation time and the size of the
c    simulation time step.
c
     ntto=65
     tii=0.1
     ntt=ntto/tii
c
c    initialize the zonal counter for generation links
c
     do 219 iz=1,nzones
219  izlins(iz)=0
c
c    initialize the tagged (for stat-accumulation) cars count.
     numcars=0
c
c    read the number of zones, number of nodes, number of

c    arc-chains , number of links (arcs itself, right now),
c    number of destination nodes and the 'k' for the
c    k-shortest paths.
c    caution !! -  make sure that each zone with non-zero
c                  incoming demand has at least one
c                  destination node.
c
     read(1,14) nzones,nnodes,narcs,n,ndests,kay
c
c
c    read the zone numbers and fix-up pseudo zone numbers
in
c    the order in which the zones appear in the data file.
c    later on, the reduced demand matrix between these zones

c    will be read based on these pseudo numbers.
c    npz() has the pseudo-zone number of each zone.

c
     read(1,14) (ipz(i),i=1,nzones)
14   format(10i5)
c
     do 17 i=1,nzones
17   npz(ipz(i))=i
```

```
c
c     read the destination nodes of these zones, in the same

c     order as the zones.
c     there will be zeros for some zones, which means they do

c     not have destination nodes.
c
      read(1,14) (idz(i),i=1,nzones)
c
c     fix up the list of destination nodes. there could be a
c     lesser number of destination nodes than there are zones.
c     idests() has the list of these dest-nodes, and ioz()
c     has the pointer showing which node in idests corresponds

c     to each zone. if there is no dest-node, ioz() will have
c     a zero for that zone..
c
      ico=0
      do 43 i=1,nzones
      if(idz(i).gt.0) then
       ico=ico+1
       idests(ico)=idz(i)
       ioz(i)=ico
      else
       ioz(i)=0
      endif
43    continue
c
c     initialize the total lane miles in each zones.
c
      do 42 iz=1,nzones
       totlmz(iz)=0
42    continue
c
c     read the node numbers and the zone number of that
node.
c     then, depending on the pseudo-zone number fixed up
above,
c     store the psud0-zone of each node.
c
      do 70 i=1,nnodes
      read(1,13) i1,i2
70    izone(i1)=i2
13    format(2i5)
c
c
c     read the node-arc data to be used for path calculations
c     the above link data is used for traffic simulation. this
c     means that there could be chains of the above links
c     between the nodes being read below. there is one arc

c     between each node-pair (there can be a reverse arc too).
c     the storage is in forward star. this means that the
c     variable, npoint(n) shows the location where the set of
c     arcs into node-n starts. (this is really a reverse star
c     storage, for use in the the shortest-path calculations
c     which will be done backwards from destinations).
c     the upstream nodes of these
c     arcs will be stored in ifwdarc(k,1)  and the number of
c     the first link in the link-chain constituting that arc
c     will be stored in ifwdarc(k,2). similarly, the
c     number of the last link in the chain is in ifwdarc(k,3)
c
      do 221 i=1,n
      read(1,11) iunod(i),idnod(i),i3,i4,nlanes(i),vmax(i)
11    format(3i5,2i2,f6.3)
c
c     some adjustments on link lengths according to the vehicle
***
c     array dimensions.
c
c
c     if there is a single lane, a length of 10800 at 160 v/mile
c     jam concentration means about 330 vehicles. change the

c     statement based on the mtxj array dimension and the jam

c     concentration.
c
c     first, expand the austin network...(correct later !)
c
      i3=3*i3
c
      if(i3.gt.(10800.0/nlanes(i))) i3=ifix(10800.0/nlanes(i))
      s(i)=float(i3)/5280.0
c
c     see which zone the link gets the traffic volume from,
c     and add to the link-count of that zone. the zonal demand
c     will be divided over these links, later.
c
c     depending on if i4 is 0,1 or 2, the link will get no
c     volume, volume from the zone of the upstream-node or

c     volume from the zone of the downstream-node.
c
c     izlins has the number of genaration links for each zone
c     izlin  stores the link numbers of the generation links
c     of each zone.
c
      if(i4.eq.0) then
       ildem(i)=0
      else
       if(i4.eq.1) ildem(i)=iunod(i)
       if(i4.eq.2) ildem(i)=idnod(i)
       izlins(npz(izone(ildem(i))))=izlins(npz(izone(ildem(i))))+1
       izlin(npz(izone(ildem(i))),izlins(npz(izone(ildem(i)))))=i
       totlmz(npz(izone(ildem(i))))=totlmz(npz(izone(ildem(i))))
     *                        +nlanes(i)*s(i)
      endif
221   continue
c
```

```
c     read the number of intervals, and the start-times of
c     each interval.
c
      read(1,229) nints
c     write(4,229) nints
      read(1,231) (begint(i),i=1,nints)
c     write(4,231) (begint(i),i=1,nints)
229   format(i5)
231   format(10f6.1)
c
c     read the zonal demand data matrix.
c     the matrix is formed using the pseudo-zone numbers.

c
      do 223 iz=1,nzones
      do 223 int=1,nints
      ztdem(iz,int)=0.
      read(1,224) (zdem(iz,izz,int),izz=1,nzones)
c     write(4,224) (zdem(iz,izz,int),izz=1,nzones)
224   format(6f10.1)
      do 223 izz=1,nzones
      if(iz.ne.izz) ztdem(iz,int)=ztdem(iz,int)+zdem(iz,izz,int)
223   continue
c
c     fix up the cumulative probability curve for generation
c     of demand towards each zone, in zfdem(iz,izz,int)
c
      do 232 int=1,nints
      do 232 iz=1,nzones
      do 227 izz=1,nzones
      if(izz.eq.1) then
       if(ztdem(iz,int).gt.0.0005) then
        zfdem(iz,izz,int)=zdem(iz,izz,int)/ztdem(iz,int)
        if(iz.eq.izz) zfdem(iz,izz,int)=0.0
       else
        zfdem(iz,izz,int)=0.0
       endif
      else
       if(ztdem(iz,int).gt.0.0005) then
        zfdem(iz,izz,int)=zfdem(iz,izz-1,int)+zdem(iz,izz,int)/
     *                         ztdem(iz,int)
        if(iz.eq.izz) zfdem(iz,izz,int)=zfdem(iz,izz-1,int)
       else
        zfdem(iz,izz,int)=zfdem(iz,izz-1,int)
       endif
      endif
227   continue
c     write(4,11212) (zfdem(iz,izz,int),izz=1,nzones)
11212 format(6f9.5)
232   continue
c
c     the following block fixes up the number of connected
c     downstream links for all the links in the network
c     also, the number of incident links are also fixed up
c     in the inlink array along with the array itself
c
      do 222 i=1,n
      link(i,1)=i
      inlink(i,1)=i
      link(i,8)=0
      inlink(i,8)=0
      do 222 j=1,n
      if(iunod(j).eq.idnod(i)) then
        link(i,8)=link(i,8)+1
        link(i,link(i,8)+1)=j
      endif
      if(idnod(j).eq.iunod(i)) then
        inlink(i,8)=inlink(i,8)+1
        inlink(i,inlink(i,8)+1)=j
      endif
222   continue
c
c     fix up the forward-star and backward-star of the
c     network.
c
      k=1
      do 51 i=1,660
      npoint(i)=k
      do 52 j=1,n
       if(idnod(j).eq.i) then
        ifwdarc(k,1)=iunod(j)
        ifwdarc(k,2)=j
        k=k+1
       endif
52    continue
51    continue
      npoint(660+1)=k
c
      do 54 i=1,660
      nreach(i)=0
54    continue
c
c
      do 797 i=1,n
      xl(i)=nlanes(i)*s(i)
      ni(i)=float(ifix(xl(i)*densmax))
      if(ni(i).gt.335)
     *write(4,*) 'more cars in link than array size !  ',i,ni(i)
      entry(i)=entrymx*nlanes(i)
797   continue
c
      if(ipert.eq.1) then

cpr   the following block reads the data on perturbations

cpr   the whole block is unexecuted when there are no pertur-

cpr   bations at all ;  that is when idays is zero.

cpr   idays is the total number of days with perturbations.
cpr   numday array stores the serial numbers of the days with
```

```
cpr  perturbations.
cpr  numsec array stores the numbers of links with perturba-

cpr  tions corresponding to each day with perturbations. this
cpr      means that the array has no values defined
corresponding
cpr  to the days without perturbations.
cpr  isect array  has the link numbers corresponding to each

cpr  separate perturbation on each day.
cpr  nupuls array has the number of different pulses in each

cpr  perturbation on each day.
cpr  ratio array has the perturbation ratio corresponding to
cpr  each pulse of each perturbation on each day.
cpr  time1 and time2 arrays have the starting and ending time
cpr  respectively of each pulse of each perturbation on each

cpr  day.


     read(1,179) idays
179  format(i3)

     if(idays.gt.0) then


     do 175 inq=1,idays
     read(1,179)numday(inq)
175  continue
     do 195 inw=1,idays
     inu=numday(inw)
     read(1,179)numsec(inu)
195  continue
     do 180 ini=1,idays
     ins=numday(ini)
     do 180 inj=1,numsec(ins)
     read(1,178)isect(ins,inj),nupuls(ins,inj)
178  format(i3,i3)
180  continue
     do 181 m1=1,idays
     m2=numday(m1)
     do 181 m3=1,numsec(m2)
     do 181 m4=1,nupuls(m2,m3)

read(1,182)ratio(m2,m3,m4),time1(m2,m3,m4),time2(m2,m3,
   1m4)
182  format(f8.5,2f6.1)
181  continue

     endif

     endif
c
c    initialize the link input volume array with zeros.
c
      do 666 l=1,10
      do 666 i=1,n
      sig(i,l) = 0.0
  666 continue
c
c
      t=0.0
      jj=0
c
c
c    initialize the arrays for link-link flux limits with
c    high numbers to prevent that check, if needed.
c
     if(iflux.ne.1) then
      do 72 i=1,n
       do 72 j=1,inlink(i,8)
        limflux(j,i) = 9999
        influx(j,i) = 0
72      continue
     endif
c
c
c
c
c set up initial conditions for each run.
     akf=0.
     inchek=1
c
c
c
c
609  akf=akf+1.
     idy = ifix(akf)
c
c
c    initialize the arrays for the link-entry and the link-
c    end queues.
c
     do 686 i=1,n
     ntryq(i)=0
     totalql(i)=0.0
     nqactiv(i)=0
     nqac1(i)=0
     totendq(i)=0.0
     nenqact(i)=0
     nenac1(i)=0
686  continue
c
c
c
     write(2,939) akf
939  format(//,5x,'simulation days=',2x,f4.0)
     if(akf.gt.1)go to 149
149  do 6767 j=1,69000
cd   congest(j)=0
```

```
cd    congesd(j)=0
cd    congesn(j)=0
cd    condist(j)=0
cd    cflag(j)=0
      qflag(j)=0.0
      tqwait(j)=0.0
      nottin(j)=0
      itag(j)=0
      nsw(j)=0
6767  continue
c
c
c
      do 76 i=1,n
c     (if(vmax(i).lt.0.918) p(i)=0.8597
c     (if(vmax(i).lt.0.834) p(i)=0.8973
c     (if(vmax(i).lt.0.751) p(i)=0.9372
c     (if(vmax(i).lt.0.668) p(i)=1.0
c     (if(vmax(i).lt.0.584) p(i)=1.0959
      p(i)=log((vmax(i)/2-eps)/(vmax(i)-eps))/log(1.-2./3.)
      gen(i)=0.0
      iflag(i)=0
      npar(i)=0
      ndque(i)=0
      do 99 kj=1,335
99    mtxj(i,kj)=0
      do 98 kj=1,128
98    mtqj(i,kj)=0
      c(i)=0.0
      v(i)=vmax(i)
      statmpt(i)=s(i)/v(i)
      cmax(i)=ni(i)/xl(i)
c     the arrays for linkwise congestion fractions(per time)
      secont(i)=0.0
      conct(i)=0.0
76    conglim(i)=(2.00/3.00)*cmax(i)
c
c     fix up the (n+1)th link for a dummy calculation in the
c     vector processing of arc trip times... ignore...
      s(n+1)=0.001
      v(n+1)=10000.0
      link(n+1,8)=2
c
      do 152 j=1,69000
      stime(j)=0.0
      atime(j)=0.0
152   ttinow(j)=0.0
c
c     fix up the flux limits into the destination link.
c     no limits at this point.
c
      do 18 j=1,inlink(n+1,8)
      limflux(j,n+1) = 99999
18    continue
c
c     go to 432
c
c
c     initialize int to show the demand interval.
      int=0
      tnext=begint(1)
c
c
c     the time step loop.
c
c
c
      do 12 l=1,ntt
      tend=l*tii
      t=tend-tii
c
c     see if a new demand interval is starting. if so, update
c     the interval number int and the expected demand to be

c     generated in each zone.
c
      if(tend.gt.(tnext+0.005)) then
       int=int+1
       if(int.lt.nints) tnext=begint(int+1)
       if(int.eq.nints) tnext=float(ntto)
       do 92 iz=1,nzones
       expgenz(iz)=ztdem(iz,int)/((1/tii)*(tnext-begint(int)))
       if(int.eq.1) expgenz(iz)=expgenz(iz)/4.0
c      write(4,11313) int,begint(int),ztdem(iz,int),expgenz(iz)
11313  format('int,beg,ztdem,expgz =',i3,f5.1,2f10.3)
92     continue
      endif
c
c     if(l.ge.23) go to 432
c
      if((t.ge.endtm).and.(numcars.eq.0)) go to 433
c
c
c     call addchain(l,linkmax,narcs,n)
c
c
      if((l/kspstep)*kspstep.eq.l-1) then
        call kshort(narcs,nnodes,ndests,kay,n,t)
      else
        call routetm(narcs,nnodes,ndests,kay,n)
      endif
c
c
c     if the time has just gone over the start-up time, then
c     copy the shortest paths for initial routes of the
c     vehicles. these routes are stored unchanged till the end
c     of simulation.
c
      if(l.eq.ifix((starttm/tii)+0.5)) then
       do 19 io=1,32
       do 19 jo=1,12
       do 19 ko=1,2
```

```
      do 19 lo=1,661
       lastno1(io,lo,jo,ko)=lastnod(io,lo,jo,ko)
19     continue
      initcop=1
      endif
c
c     loop over the links before the simulation of this
c     time step.
c
      if(ipert.ne.1) go to 67
c
      do 21 i=1,n
c
c     first, check and see if the link capacities have to
c     be adjusted to simulate perturbations.
c
      if(ifix(akf).eq.numday(inchek)) then
        if(inchek.lt.idays) inchek=inchek+1
        do 66 i1=1,numsec(idy)
        do 66 i2=1,nupuls(idy,i1)
        if(((i.eq.(isect(idy,i1))) .and.
     1      (t.gt.(time1(idy,i1,i2)-tii))) .and.
     2      (t.lt.(time1(idy,i1,i2)+tii)))
     3   call change(i1,i2,idy,xl,i)
        if(((i.eq.(isect(idy,i1))) .and.
     1      (t.gt.(time2(idy,i1,i2)))) .and.
     2      (t.lt.(time2(idy,i1,i2)+(2.0*tii))))
     3   call restor(i1,i2,idy,xl,i)
        c(i)=(npar(i)*mtnum)/xl(i)
c     the perturbations are assumed to set in gradually,
c     at least gradual enough for the concentration not
c     to exceed the maximum. this is equivalent to assuming

c     that the vehicles which have already entered the
c     link  are allowed to move at the maximum concentration

c     the next statement makes sure of this.
66        if(c(i).ge.cmax(i)) c(i)=cmax(i)
      endif
21    continue
c
c     initialize the link volume array
c
67    do 77 i=1,n
77    vlg(i)=0.0
c
c
c     divide the volumes from each zone into the generation

c     links of the zone.
c     find the expected number of vehicles to be generated in

c     each link by dividing the total zonal generation per
c     time step in proportion to the lane_miles in each link.
c     to avoid round_off errors, the integer part of this
c     expected number is always generated in the link, but the
c     remaining part is assigned only in some time steps,
c     based on a random number that is called.
c
c
      do 68 iz=1,nzones
      do 68 il=1,izlins(iz)
      expgen(izlin(iz,il))=expgenz(iz)*xl(izlin(iz,il))/totlmz(iz)
      call mun(nr,r)
      if((expgen(izlin(iz,il))-ifix(expgen(izlin(iz,il))))
     *   .ge.r(5)) then
        vlg(izlin(iz,il))=vlg(izlin(iz,il))+
     *                   ifix(expgen(izlin(iz,il))) + 1
      else
        vlg(izlin(iz,il))=vlg(izlin(iz,il))+
     *                   ifix(expgen(izlin(iz,il)))
      endif
c     if(iz.gt.8) write(4,12113) izlin(iz,il),iz,
c   * expgenz(iz),expgen(izlin(iz,il)),r(5),vlg(izlin(iz,il))
12113   format('link,zone,expzone,explink,rand,vlg  =
',2i4,3f10.4,f5.1)
68    continue
c
c     go to 12
c
      do 22 i=1,n
c     initialize the link-influx-demand-counter array.
      intoo(i)=0
c     initialize the link-end-active-time-step flag array.
      nenaflg(i)=0
22    continue
c
c
c  the particle code calculations begin here
c
c
      write(4,13198) jj,numcars
13198 format('vehicles = ',i5,'     tagged ones still in = ',i5)
      call partco(initcop,lfl,n,nzones,kay,t,tii,idy,akf)
      if(lfl.eq.1) go to 433
      do 32 i=1,n
      ck(i,l)=c(i)
 32   continue
c
c
12    continue
c
433   write(4,19898) (nreach(k),k=1,660)
19898 format(10i6)
      ntreach=0
      do 598 k=1,660
      ntreach=ntreach+nreach(k)
598   continue
      write(4,19899) ntreach
19899 format('total gone out =',i7)
c     go to 432
      write(21,164) fracinf,ribfa,bound
```

```
      write(2,164)  fracinf,ribfa,bound
      write(4,164)  fracinf,ribfa,bound
c     write(5,164)  fracinf,ribfa,bound
      write(6,164)  fracinf,ribfa,bound
      write(7,164)  fracinf,ribfa,bound
      write(8,164)  fracinf,ribfa,bound
      write(37,164) fracinf,ribfa,bound
      write(39,164) fracinf,ribfa,bound
      write(9,164)  fracinf,ribfa,bound
      write(10,164) fracinf,ribfa,bound
      write(51,164) fracinf,ribfa,bound
      write(52,164) fracinf,ribfa,bound
      write(61,164) fracinf,ribfa,bound
164   format('fraction with info =',f6.3,'  avg.ib-fraction ='
     *,f5.2,'  bound =',f5.2/)
c
      if(ipert.ne.0) then
      do 141 m1=1,idays
      m2=numday(m1)
      do 141 m3=1,numsec(m2)
      do 141 m4=1,nupuls(m2,m3)
      m5 = isect(m1,m3)
      em6= ratio(m2,m3,m4)
      write(21,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(2,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(4,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
c     write(5,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(6,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(7,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(8,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(37,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(39,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(9,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(10,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(51,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(52,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
      write(61,167)
m2,m5,m4,time1(m2,m3,m4),time2(m2,m3,m4),em6
167   format('day =',i3,'.  link =',i4,'.'/'perturbation',i2,
     1' is from',f6.1,' to',f6.1,'.   reduction = ',f7.5//)
141   continue
      endif
c
```

```
c     write(2,122)
122   format(1h1,' the following is the information for each
macroparic
     1le in the
system'//1x,'macroparticle',5x,'gen.sector',5x,'starting
     1 time',5x,'trip time',5x,'arrival time')
c     write(15,132)
132   format(1h1,'the following is the information on how much
congestion
     2each particle
experienced'//1x,'macroparticle',5x,'gen.sector',5x,
     1'cong.number',5x,'cong.time',5x,'cong.distance')
c     write(16,113)
113   format(1h1,'the following is the information on how much
congestion
     1each particle
experienced'//1x,'macroparticle',5x,'gen.sector',5x,
     2'cong.number',5x,'cong.time',5x,'cong.distance'/29x,'
per dist.',
     34x,'per triptime',3x,' per dist.')
c     write(16,137)akf
c     write(15,137)akf
137   format(//'for day = ',f3.0)
c
c     the concentration profiles are written in the files
c     tape51.
c
      write(51,536) l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12
 536  format('vehicle densities in selected links, at the end of',
     */'every minute of simulation, in veh per lane-mile.'//
     *'time                         links'/
     *'step  ',12i6/)
      do 579 k=1,ntt
       l=k
c      l=(1.0/tti)*k
      write(51,537) k,ck(l1,l),ck(l2,l),ck(l3,l),ck(l4,l)
     *      ,ck(l5,l),ck(l6,l),ck(l7,l),ck(l8,l),ck(l9,l)
     *      ,ck(l10,l),ck(l11,l),ck(l12,l)
 537  format(i3,3x,12(f6.1))
 579  continue
c
      do 900 i=1,n
      do 899 m=1,3
      tag(i,m)=0.
      tagerly(i,m)=0.
      taglate(i,m)=0.
      trt(i,m)=0.
      sqsum(i,m)=0.
cvar  ssum(i,m)=0.
899   continue
      do 898 m=1,7
898   klink(i,m)=0
900   continue
c
      newchos=0
      do 903 m=1,4
```

```
      dswch(m)=0
      nswch(m)=0
903   continue
c
c
      do 131 j=1,jj
c
c     select only the vehicles starting after the start-up time
c     and leaves the system.
c
      if((stime(j).lt.starttm).or.(nottin(j).eq.0)) go to 131
      if(itag(j).eq.0) go to 131
c
c
c
c
c
c     if the congestion report of each day is specified ,
c     then these are printed on tape11.
c     the average values are calculated and printed on tape15
cd    conperd(j)=congesd(j)/(distans(j))
cd    conpern(j)=congesn(j)/(distans(j))
cd    conpert(j)=congest(j)/(ttilnow(j))
cd    write(15,1243)j,isec(j),congesn(j),congest(j),congesd(j)
cd    write(16,1243)j,isec(j),conpern(j),conpert(j),conperd(j)
1243  format(i10,14x,i2,6x,f10.3,8x,f10.3,8x,f10.3)
c
c     route-switch statistics.
c
      if(nsw(j).gt.0) then
      if(nsw(j).ge.4) km=4
      if(nsw(j).lt.4) km=nsw(j)
       do 971 m=1,km
       dswch(m)=dswch(m)+dsw(j,m)/distans(j)
       nswch(m)=nswch(m) + 1
971    continue
      endif
      nswches=nswches+nsw(j)
c
c
      js=isec(j)
      sdabs=abs(wst-atime(j))
      sdlate = 0.0
      sderly = 0.0
      ontime = 0.0
      if((wst-atime(j)).gt.0.0) sderly = wst-atime(j)
      if((wst-atime(j)).lt.0.0) sdlate = atime(j)-wst
      if(wst.eq.atime(j))       ontime = 1.0
      do 911 i=1,n
      if (js.eq.i) then
        klink(i,1) = klink(i,1) + 1
        trt(i,1) = trt(i,1) + ttilnow(j)
cvar     tsd(i,klink(i,1),1) = sdabs
        tag(i,1)=tag(i,1)+sdabs
        taglate(i,1) = taglate(i,1) + sdlate
        tagerly(i,1) = tagerly(i,1) + sderly
        if (info(j).eq.0) then
        klink(i,2) = klink(i,2) + 1
        if(sderly.gt.0.0) klink(i,4) = klink(i,4) + 1
        if(sdlate.gt.0.0) klink(i,5) = klink(i,5) + 1
        if(ontime.gt.0.0) klink(i,4) = klink(i,4) + 1
        trt(i,2) = trt(i,2) + ttilnow(j)
cvar     tsd(i,klink(i,2),2) = sdabs
        tag(i,2)=tag(i,2)+sdabs
        taglate(i,2) = taglate(i,2) + sdlate
        tagerly(i,2) = tagerly(i,2) + sderly
        endif
        if (info(j).ne.0) then
        klink(i,3) = klink(i,3) + 1
        if(sderly.gt.0.0) klink(i,6) = klink(i,6) + 1
        if(sdlate.gt.0.0) klink(i,7) = klink(i,7) + 1
        if(ontime.gt.0.0) klink(i,6) = klink(i,6) + 1
        trt(i,3) = trt(i,3) + ttilnow(j)
cvar     tsd(i,klink(i,3),3) = sdabs
        tag(i,3)=tag(i,3)+sdabs
        taglate(i,3) = taglate(i,3) + sdlate
        tagerly(i,3) = tagerly(i,3) + sderly
        endif
      endif
911   continue
131   continue
c
c
      do 910 m=1,3
910   vtotal(m)=0.0
      infno  = 0
      noino  = 0
      do 912 i=1,n
      noino = noino + klink(i,2)
      infno = infno + klink(i,3)
      do 912 m=1,3
      if(klink(i,m).ne.0) then
        vtr(i,m)=trt(i,m)/float(klink(i,m))
        vtotal(m) = vtotal(m) + trt(i,m)
      endif
912   continue
      vtothr1 = vtotal(1)/60.0
      vtothr2 = vtotal(2)/60.0
      vtothr3 = vtotal(3)/60.0
      vavg1  = vtotal(1)/float(infno+noino)
      if(noino.gt.0) vavg2  = vtotal(2)/float(noino)
      if(infno.gt.0) vavg3  = vtotal(3)/float(infno)
c
c     route-switch overall statistics.
c
      do 904 m=1,4
      if(nswch(m).gt.0) dswch(m)=dswch(m)/float(nswch(m))
904   continue
c
c
      do 913 j=1,jj
```

```
      do 913 i=1,n
      do 913 m=1,3
      if(klink(i,m).ne.0) then
      if(isec(j).eq.i) sqsumt(i,m) =
     *  sqsumt(i,m)+(ttinow(j)-vtr(i,m))*(ttinow(j)-vtr(i,m))
      endif
913   continue
c
c     system-wide statistics in tape4
c     output avg.trip times in tape9
c     output avg.abs.schedule delay in tape5
c     output avg.erly.schedule delay in tape7
c     output avg.late.schedule delay in tape8
c     output trip time std.deviation in tape10
c     output schedule delay std.deviation in tape6
c
      write(4,*) 'maximum simulation time = ',float(ntto)
      write(4,*) 'start-up time          = ',starttm
      write(4,*) 'end of time of interest = ',endtm
      write(4,*) 'total vehicles         = ',infno+noino
      write(4,*) 'with info = ',infno,'   without info = ',noino
      write(4,3234) vtothr1,vtothr2,vtothr3,vavg1,vavg2,vavg3
3234  format('total trip times (hrs) & avg.trip times (min) ',
     *' - overall, noinfo & info :- ',3f8.2,3f7.3///)
      write(4,*) 'route-switch statistics'
      write(4,*) 'total number of switches = ',nswches
      write(4,*) ' '
      write(4,*) 'numbers of 1st, 2nd, 3rd and 4th switches'
      write(4,*) 'and their avg. distance fractions'
      write(4,*) ' '
      write(4,3533) ((nswch(m),dswch(m)),m=1,4)
3533  format(i7,f10.4)
      do 914 i=1,n
      do 919 m=1,3
      if(klink(i,m).ne.0) then
      stdtrt(i,m)=sqrt(sqsumt(i,m)/float(klink(i,m)))
      vlag(i,m)=tag(i,m)/float(klink(i,m))
      vlage(i,m)=tagerly(i,m)/float(klink(i,m))
      vlagl(i,m)=taglate(i,m)/float(klink(i,m))
      do 915 kb=1,klink(i,m)
cvar   sa = abs(tsd(i,kb,m))-vlag(i,m)
cvar   ssum(i,m) = ssum(i,m) + sa*sa
915   continue
cvar  std(i,m) = sqrt(ssum(i,m)/float(klink(i,m)))
      endif
919   continue
      if(klink(i,4).ne.0) vlage(i,4)=tagerly(i,2)/float(klink(i,4))
      if(klink(i,5).ne.0) vlagl(i,4)=taglate(i,2)/float(klink(i,5))
      if(klink(i,6).ne.0) vlage(i,5)=tagerly(i,3)/float(klink(i,6))
      if(klink(i,7).ne.0) vlagl(i,5)=taglate(i,3)/float(klink(i,7))
914   continue
c
c     write headers for tape5 to tape10.
c
c     write(5,3525)
3525  format('average absolute schedule delay.   averaged
over'/
     *'all, no-info  and info vehicles.'/)
      write(6,3526)
3526  format('std.deviation of abs.schedule delay.  averaged
over'/
     *'all, no-info and  info vehicles.'/)
      write(7,3527)
3527  format('average early-side schedule delay.    averaged
over'/
     *'all, no-info, info, no-info & early, info & early vehicles'/
     *'the numbers are in perentheses.'/)
      write(8,3528)
3528  format('average late-side schedule delay.    averaged
over'/
     *'all, no-info, info, no-info & late, info & late vehicles'/
     *'the numbers are in perentheses.'/)
      write(9,3529)
3529  format('average trip time.     averaged over'/
     *'all, no-info  and info vehicles.'/)
      write(10,3530)
3530  format('std.deviation of trip times.    averaged over'/
     *'all, no-info  and info vehicles.'/)
c
      do 916 i=1,n
c     write(5,3531) idy,i,(vlag(i,m),m=1,3)
cvar  write(6,3531) idy,i,(std(i,m),m=1,3)
      write(7,3532) idy,i,(vlage(i,m),m=1,5),(klink(i,m),m=1,3)
     *,klink(i,4),klink(i,6)
      write(8,3532) idy,i,(vlagl(i,m),m=1,5),(klink(i,m),m=1,3)
     *,klink(i,5),klink(i,7)
      write(9,3531) idy,i,(vtr(i,m),m=1,3)
      write(10,3531) idy,i,(stdtrt(i,m),m=1,3)
3531  format(i2,i4,3f7.2)
3532  format(i2,i4,5f7.2,' (',5(i5,','),')')
916   continue
c
c
c
c     the next block calculates the fraction of particles
c     experiencing congestion in each  link.
c
c
c
cn    do 130 i=1,n
cn    confrac(i)=0.0
cn130 continue
cn    do 119 kk=1,jj
cn    if(congesn(kk).gt.0) then
c     n
confrac(isec(kk))=confrac(isec(kk))+(1/float(klink(isec(kk),1)))
cn    endif
cn119 continue
cn    write(37,125)
125   format('the fraction of particles generated in each link',

     *'/that experience congestion on the way.'/)
```

```
cn    do 127 i=1,n
cn    write(37,135)idy,j,confrac(i)
135   format(i3,i5,f7.2)
cn127 continue
c
c
c     the next statements calculate the fraction of time when
c     each link was under congestion.
c     print it in tape39
c
      write(39,126)
126   format('fraction of time that each link is under
congestion.'
     */)
      do 441 i =1,n
      do 442 j =1,ntt
      if(ck(i,j).gt.0.0) conct(i)=conct(i)+tii
      if(ck(i,j).ge.conglim(i)) secont(i)=secont(i)+tii
 442  continue
      if(conct(i).ne.0.0) cfrac(i)=secont(i)/conct(i)
      if(conct(i).eq.0.0) cfrac(i)=0.0
      write(39,135)idy,i,cfrac(i)
 441  continue
c
      write(61,570)
570   format('average length of link end queues and the
average',/
     *'length of the link entry queues.',//
     *'note: both the queues are averaged over the time steps',/
     *'  when there were queues, shown as the first number',/
     *'  in parentheses. the second number in parentheses',/
     *'  is the number of steps with vehicle out-flux or',/
     *'  queue for link end queues, and the number of time',/
     *'  steps with vehicle-entry or queue for the link',/
     *'  entry queues.'//)
c
c
c     link end queues.
c
c     totendq(i) has the total number of vehicles in the link
c     end queue, and nenqact(i) has the number of time steps

c     over which the link-end queues existed.
c
c     entry queue statistics.
c
c     the average length of queues are calculated and written

c     here. totalq(i) keeps the total sum of the queuelengths
c     at different times. nqactiv stores the total sum of the
c     time steps when there were partcles queuing at the link.
c
      do 571 i=1,n
      aventrq=0.0
      if(nqactiv(i).gt.0) aventrq=totalq(i)/float(nqactiv(i))
      avendq=0.0
      if(nenqact(i).gt.0.0) avendq=totendq(i)/float(nenqact(i))
      write(61,139)i,avendq,nenqact(i),nenac1(i),aventrq,
     * nqactiv(i),nqac1(i)
571   continue
139   format(i5,2(f8.2,' (',i4,',',i4,')'))
c
c
      if(idy.lt.lastday) go to 609
c
432   stop
      end
      subroutine partco(initcop,lt,n,nzones,kay,t,tii,idy,akf)
      dimension r(10),r1(10)
      dimension cmax(1781),c(1781),vmax(1781),p(1781),
     *gen(1781),vlg(1781),labelnk(1781)
      dimension xl(1781),npar(1781),nout(1781),iflag(1781)
      dimension stime(69000),isel(676)
      dimension atime(69000),isec(69000)
      common/fni12/xpar(69000),tleft(69000),tqwait(69000)
      common/fni1/cmax,vmax,p,eps,prob,vlg,xl,c,kk
      common/fni2/isec,stime,atime,illo,jj,minum,sd,ist,npar,nout
      common/fni4/iflag,gen,noutflx(1781),entry(1781)

common/fni3/mtxj(1781,335),mtqj(1781,128),mxj(335),myj(335)
      common/fni6/numsec(1),isect(1,1781)
      common/fni7/ratio(1,1781,3),nupuls(1,1781)
      common/fni10/s(1781),v(1781),nlanes(1781),nmov(1781)
      common/fni13/time1(1,1781,3)
      common/fni14/time2(1,1781,3)
      common/fni15/ndque(1781),anqr(1781),expquet(1781)
     *       ,nqaccum(1781),ndqr(1781,30)
      common/fni16/totalql(1781),totendq(1781),nqactiv(1781),
     *nenqact(1781),ntryq(1781),nenac1(1781),nenaflg(1781),
     *nqac1(1781)
c        d
common/fni8/congesn(69000),congesd(69000),congest(69000),
cd   *condist(69000),cflag(69000)
      common/fni5/conglim(1781),distans(69000),qflag(69000)
     * ,nsw(69000),dsw(69000,4)
      common/jay1/link(1781,8),klink(1781,7),statmpt(1781)
     *       ,inlink(1781,8),intooi(1781,676,3),intoo(1781)
      common/jay2/ttinow(69000),jdest(69000),jpath(69000,90)
     *       ,ribf(69000),icurrnt(69000),nottin(69000)

common/jay3/info(69000),itag(69000),jsw(69000),nexlink(690
00)

common/jay4/startm,endtm,numcars,iseed,fracinf,ribfa,bound
      common/jay5/influx(7,1781),limflux(7,1781),cayvee(7)

common/jn1/ifwdarc(1779,2),npoint(661),idests(32),nreach(661)

common/jn2/iwork1(1779),iwork2(1779),iunod(1779),idnod(17
79)
      common/jzone/izlins(36),izlin(36,401),zdem(36,36,10)
```

```
     ,izone(661),ioz(36),idz(36),ipz(36),npz(801),zfdem(36,36,10)
    * ,jidem(1781),ztdem(36,10),int,totimz(36)
    * ,expgenz(36),expgen(1781)
c
c
c    data statements for certain key simulation parameters.
c
c    ribfa is the average time-indifference-band for route
c    change as a fraction of the remaining trip time. It has
c    a triangular distribution over this mean value and so
c    is different for different drivers.
c
c    fracinf is the fraction of drivers getting information.
c
c    bound is the lower limit of the indifference band,
c    which is the minimum trip time advantage needed for a
c    route change.
c
c    nqs is the number of time steps over which the queue
c    accumulation is done to calculate an expected average
c    queue dispersion time, which is used in route trip time
c    prediction.
c
c
c    set entrymx to a smaller figure (say, 60 to 80 veh/min),
c    for a limit on the entry queue discharge.
c    set iqwait=1, if entry-queue waiting time is to be added
c    to the trip-time, and the start-time of the vhicle to be
c    the time of joining the queue.
c    set iflux=1, for flux limits from link to link.
c
     data ribfa/0.2/
     data fracinf/0.50/
     data bound/0.25/
     data nqs/30/
     data entrymx/40.0/
     data iqwait/1/
     data iflux/1/
c
     nr=10
     tend=t+tii
c
c
c    loop over the links and simulate the traffic.
c
c
     do 7 i=1,n
     nout(i)=0
     nmov(i)=0
     limentr=entry(i)*tii
c
c    add the new incoming volume to the waiting volume.
c
     gen(i) = gen(i) + vig(i)*1.5
c
c    nqac1 has the number of time steps with generation
c    or entry queue.
c
     if(gen(i).gt.0) nqac1(i) = nqac1(i) + 1
c
c    calculate the available capacity in the link.
c
     genm = (cmax(i)-c(i))*xl(i)
c
c    if concentration in the link is more than or equal to
c    the jam concentration at the end of the previous time
c    step, then the iflag is 1.
c
     if(iflag(i).ne.1) then
c
c
c
      if(c(i).ge.conglim(i)) limentr = ifix(entry(i)*tii/2.0)
      if(gen(i).gt.genm)  mnum = genm/mtnum
      if(gen(i).le.genm)  mnum = gen(i)/mtnum
      if(mnum.gt.limentr) mnum = limentr
      if(mnum.gt.0)  then
       if(ntryq(i).gt.0) then
        if(mnum.ge.ntryq(i)) k=ntryq(i)
        if(mnum.lt.ntryq(i)) k=mnum
        do 71 j=1,k
         npar(i)=npar(i)+1
         mtq(i,npar(i)) = mtqj(i,j)
         if(iqwait.ne.1) then
          stime(mtqj(i,j)) = t
          ttinow(mtqj(i,j)) = 0.0
         endif
         mtqj(i,j) = 0
71       continue
       endif
       if(mnum.gt.ntryq(i)) then
        if(ntryq(i).gt.0) k=mnum-ntryq(i)
        if(ntryq(i).lt.1) k=mnum
        do 72 j=jj+1,jj+k
         xpar(j)=s(i)/2.0
         npar(i)=npar(i)+1
         mtq(i,npar(i))=j
         call mun(nr,r)
         call mtri(nr,r1)
         if(r(5).lt.fracinf) then
          info(j)=1
          ribf(j)=ribfa + (r1(10)-0.5)*(ribfa*0.5)
         else
          info(j)=0
         endif
         if(t.lt.startm) info(j)=0
         if((t.ge.startm).and.(t.le.endtm)) itag(j)=1
```

No Page 234 in Original Document


```
52      continue
        jj = jj + new1 - ntryq(i)
        ntryq(i)= new1
        nqactiv(i)=nqactiv(i)+1
        totalql(i)=totalql(i)+ntryq(i)
       endif
c
      endif
c
c
c   initialize the influx and limflux arrays.
c
      do 53 j=1,inlink(i,8)
        influx(j,i) = 0
        limflux(j,i) = 9999
53    continue
c
c   if queue-dispersion limits are not to be applied for link
c   to link movements, skip the next block.
c
      if(iflux.ne.1) go to 51
c
      if(inlink(i,8).gt.0) then
      iqdisp = 3000.0*((tii/60.0)*(xl(i)/s(i)))/minum
      do 20 j=1,inlink(i,8)
      if(ndque(inlink(i,j+1)).gt.iqdisp) limflux(j,i) = iqdisp
20    continue
      endif
      if(c(i).gt.120) write(6,12139) t,i,c(i),npar(i),(limflux(j,i),
     * j=1,4),vlg(i),ndque(i),ntryq(i)
c     if(iunod(i).eq.603) write(6,12139) t,i,c(i),npar(i),(limflux(j,i),
c    * j=1,4),vlg(i),ndque(i),ntryq(i)
12139 format(f5.1,i5,f6.1,i4,4i5,f6.1,i4,i4)
c
c   go over each vehicle and see if they reach the end.
c   otherwise, move them.
c
c
51    do 8 kj=1,335
      j=mtxj(i,kj)
      if(j.eq.0) go to 8
      if((t.gt.44.85).and.(itag(j).eq.1)) then
       write(6,11812) i,j,isec(j),jdest(j),info(j),c(i),v(i)
11812 format('i,j,isec,des,inf,c,v=',i4,i6,i5,i3,i2,f6.1,f6.3)
      write(6,11912) (ipath(j,km),km=1,60)
11912 format(20i4)
      endif
      xpos=xpar(j)-v(i)*tii
      tocross = xpar(j)/v(i)
      tleft(j) = tii - tocross
c     if((idnod(i).eq.329).or.(idnod(i).eq.339)) then
c      write(6,17719) i,iunod(i),idnod(i),c(i),v(i),j,xpar(j),xpos
17719 format('i,iu,id,c,v,j,x,xp= ',3i4,f6.1,f6.3,i9,f6.3,f6.3)
c     endif
c
c   the following statements are for keeping track of
c   congestion.
c   cflag(j) has the number of minutes that particle j has
c   been experiencing congestion currently at a stretch.
c   once cflag(j) is above 3 , that will be taken as a
c   perceivable period of congestion and the congest(j)
c   which stores the total congestion time of particle j is
c   incremented accordingly.
c   congesn(j) which stores the number of congested periods
c   percieved by particle j is also incremented along with
c   congest(j). incrementing of congest(j) and congesn(j)
c   are done only after the concentration falls below
c   the congestion criterion.
c   condist array is an array exactly equivalent to cflag
c   array, but having the cumulative distance travelled in
c   the current stretch of congestion.
c
cd    congt = tii
cd    if(xpos.le.0.0) congt = tocross
cd    if(c(i).ge.conglim(i)) then
cd     cflag(j)=cflag(j)+congt
cd     condist(j)=condist(j)+v(i)*congt
cd    endif
cd    if(c(i).lt.conglim(i)) then
cd     if(cflag(j).ge.3.0) then
cd      congesn(j)=congesn(j)+1.0
cd      congest(j)=congest(j)+cflag(j)
cd      congesd(j)=congesd(j)+condist(j)
cd     endif
cd     cflag(j)=0.0
cd     condist(j)=0.0
cd    endif
      if(xpos.gt.0) go to 61
c
c   the vehicle has reached the end of the link.
c   add the portion of the time step used to reach
c   the link-end to the ttlnow(j) values.
c
      ttlnow(j) = ttlnow(j) + tocross
c
c   if the vehicle has reached the destination node, take it
c   out and fix its arrival time.
c
      if(idnod(i).eq.idests(jdest(j))) then
       nreach(idnod(i))=nreach(idnod(i))+1
       npar(i)=npar(i)-1
       nout(i)=nout(i)+1
       mtxj(i,kj)=0
       distans(j)=distans(j) + s(i)
       atime(j) = t + tocross
       notin(j)=1
       if(itag(j).eq.1) numcars = numcars - 1
       go to 8
      endif
c
c   call the next-link selection routine, which may be
```

```
c     using a route selection routine also..
c
      call getlink(t,lfl,j,i,kay)
c
c
      nl=nextlink(j)
      if(nl.lt.1) then
      write(4,12117) i,j,isec(j),info(j),jdest(j),(ipath(j,k),k=1,90)
12117 format('i,j,isec,inf,jdest,path = ',i4,i5,i5,i3,i5/6(15i5/))
       lfl=1
       go to 113
      endif
      intoo(nl)=intoo(nl)+1
      if(intoo(nl).gt.676) then
        write(4,*) 'gt. 676 vehicles in upstream demand array for
link '
     *,nl
       lfl=1
       go to 113
      endif
      intooi(nl,intoo(nl),1)=j
      intooi(nl,intoo(nl),2)=i
      intooi(nl,intoo(nl),3)=kj

      go to 8
c
c     the next two lines are executed in an 'uneventful' time
c     interval when the vehicle just moves from one position
c     to another within the same link.
c
61    xpar(j)=xpos
      ttinow(j) = ttinow(j) + tii
8     continue
7     continue
c
c
c     initialize the out-flux from each link to the downstream
c     links as zero.  this variable will be updated during
c     vehicle movement and be used for link-end queue service

c     rate calculations.
c
      do 13 i=1,n
13    noutlx(i)=0
c
c
c     loop over all the links once more to move vehicles
c     across,  based on the available capacity in the new
c     links, existence of a queue, time of the vehicle's
c     arrival at the end, and sometimes a flow allocation
c     depending on the number of particles on the inflow links.
c
      do 9 i=1,n
c
c     check if the link has incident links.
c
      if(inlink(i,8).eq.0) go to 9
c
c     set up the incident link order number of different links
c     coming into this link.
c
      do 2 k=2,7
      if(inlink(i,k).gt.0) labelnk(inlink(i,k)) = k-1
2     continue
c
c
      do 114 l=1,intoo(i)
114   isel(l)=0
c
      ncan=cmax(i)*xl(i)-npar(i)-nmov(i)-nout(i)
      nc=ncan
      if(ncan.gt.intoo(i)) nc=intoo(i)
      do 11 nb=1,nc
c
c     find the vehicle with the earliest link-end-arrival
c     time among all the vehicles remaining to be moved in.

c
      tma = -100.0
      do 10 k=1,intoo(i)
      if(isel(k).eq.0) then
      tl=tqwait(intooi(i,k,1))
      if(tl.eq.0) tl=tleft(intooi(i,k,1))
      if(tl.gt.tma) then
       tma = tl
       mk = k
      endif
      endif
10    continue
      in=intooi(i,mk,2)
      j =intooi(i,mk,1)
      kj=intooi(i,mk,3)
      isel(mk)=1
c
      if(influx(labelnk(in),i).ge.limflux(labelnk(in),i)) then
c
c     the vehicle is unable to move in due to the flux limit
c     constraint.
c     we add 1 to the link end queue.
c     the xpar(j) is fixed as zero.
c     add the remaining time to the clag array.  as it is
c     not moving, the condist array is not changed.
c     note that tleft is less than tii during the time step
c     when it reaches the end and is tii for the subsequent
c     time steps there, as is clear from the formula for
c     tleft before.
c
      if(qflag(j).lt.0.5) then
       ndque(in)=ndque(in)+1
       qflag(j)=1.0
       xpar(j)=0.0
      endif
```

```
cd    cflag(j)=cflag(j)+tleft(j)
      tqwait(j)=tqwait(j)+tleft(j)
      ttilnow(j) = ttilnow(j)+tleft(j)
c
      else
c
c     move the vehicle into link (i) from link (in).
c     also, we increment the influx into link (i) from link
c     (in) during this time step by one.
c     increment the out-flux from (in) too.
c
      influx(labelnk(in),i) = influx(labelnk(in),i) + 1
      noutflx(in) = noutflx(in) + 1
      if(qflag(j).gt.0.5) then
        qflag(j)=0.0
        ndque(in)=ndque(in)-1
        tqwait(j)=0.0
      endif
c
c     fix the particle position from link-end.
c     add the time left to its time-till-now array.
c
      xper(j)=s(i)-v(i)*tleft(j)
c     do not let the particle move past this new link also during
this
c     time step. so, if position is negative, keep it as zero.

      if(xper(j).lt.0.0) xpar(j)=0.0
      ttilnow(j) = ttilnow(j) + tleft(j)
c
c     collect the statistics on the congestion experience of
c     individual particles in the new link
c
cd    if(c(i).ge.conglim(i)) then
cd      cflag(j)=cflag(j)+tleft(j)
cd      condist(j)=condist(j)+v(i)*tleft(j)
cd    endif
cd    if(c(i).lt.conglim(i)) then
cd      if(cflag(j).ge.3.0) then
cd        congesn(j)=congesn(j)+1.0
cd        congest(j)=congest(j)+cflag(j)
cd        congesd(j)=congesd(j)+condist(j)
cd      endif
cd      cflag(j)=0.0
cd      condist(j)=0.0
cd    endif
c
c     the particle is removed from the previous links array.
c
      npar(in)=npar(in)-1
      nmov(in)=nmov(in)+1
      mtxj(in,kj)=0
      if(nenaflg(in).eq.0) nenaflg(in)=1
      distans(j)=distans(j)+s(in)
c
c     check if the vehicle is switching routes now.
c     if so, set the switch-number and switch-distance arrays
c
      if(isw(j).eq.1) then
       nsw(j)=nsw(j)+1
       if(nsw(j).lt.5) dsw(j,nsw(j)) = distans(j)
      endif
c
c     add the vehicle to this link's mtxj array.
c
      npar(i)=npar(i)+1
      mtxj(i,(nout(i)+npar(i)+nmov(i)))=j
c
c     increment the icurmt() value, to keep its position
c     within the jpath array.
c
      icurmt(j)=icurmt(j)+1
c
      endif
c
11    continue
c
c     if the available capacity is less than the demand for
c     movement into the link, then keep the remaining vehicles
c     in the previous links.
c
c     this means their congestion arrays need to be modified.

c
      if(intoo(i).gt.nc) then
      do 12 k=1,intoo(i)
      if(isel(k).lt.1) then
      j = intooi(i,k,1)
      in = intooi(i,k,2)
      if(qflag(j).lt.0.5) then
       ndque(in)=ndque(in)+1
       qflag(j)=1.0
       xpar(j)=0.0
      endif
cd    cflag(j)=cflag(j)+tleft(j)
      tqwait(j)=tqwait(j)+tleft(j)
      ttilnow(j) = ttilnow(j)+tleft(j)
      endif
12    continue
      endif
c
9     continue
c
c
c     update the active-link-end-movement-time-step, if any
c     vehicle moved out.
c
      do 14 i=1,n
14    if(nenaflg(i).eq.1) nenact1(i)=nenact1(i)+1
c
c
c     compress the mtxj and mtqj arrays and bring vehicle
```

```
c     numbers to the top of the array, now that there are
c     zeros in the arrays to reflect vehicles which moved out.
c
      do 21 i=1,n
      if(npar(i).eq.0) go to 31
      lj=0
      do 22 kj=1,335
      if(mtxj(i,kj).eq.0) go to 22
      lj=lj+1
      mxj(lj)=kj
22    continue
      do 23 kj=1,lj
23    myj(kj)=mtxj(i,mxj(kj))
      do 24 kj=1,lj
24    mtxj(i,kj)=myj(kj)
      do 25 kj=lj+1,335
25    mtxj(i,kj)=0
31    lj=0
      do 26 kj=1,128
      if(mtqj(i,kj).eq.0) go to 26
      lj=lj+1
      mxj(lj)=kj
26    continue
      if(lj.eq.0) go to 21
      do 27 kj=1,lj
27    myj(kj)=mtqj(i,mxj(kj))
      do 28 kj=1,lj
28    mtqj(i,kj)=myj(kj)
      do 29 kj=lj+1,128
29    mtqj(i,kj)=0
21    continue
c
c     calculate the concentration and velocity at the end of
c     the interval.
c
      do 40 i=1,n
      iflag(i)=0
      c(i)=(npar(i)*minum)/xl(i)
      if(c(i).ge.cmax(i)) iflag(i)=1
      if(c(i).gt.cmax(i)) c(i)=cmax(i)
      if(iflag(i).eq.1) v(i)=eps
      if(iflag(i).ne.1)
     *v(i)=(vmax(i)-eps)*((1.0-c(i)/cmax(i))**p(i))+eps
40    continue
c
c     estimate the time needed to disperse the link end queue

c     based on the average queue length for a few past time

c     steps, and a certain queue dispersion service rate.
c
c     ndqr = end_queue_dispersion_rate for last few time steps
c     anqr = average_end_queue_rate for last few time steps.

c
      do 50 i=1,n
      if(ndque(i).gt.0) then
        totendq(i)=totendq(i)+ndque(i)
        nenqact(i)=nenqact(i)+1
      endif
      if(ndque(i).gt.(2.0*(xl(i)/s(i)))) then
        nqaccum(i)=nqaccum(i)+1
        if(nqaccum(i).le.nqs) then
          ndqr(i,nqaccum(i))=noutfix(i)
          anqr(i)=(anqr(i)*(nqaccum(i)-1)+noutfix(i))/nqaccum(i)
        endif
        if(nqaccum(i).gt.nqs) then
          do 81 k=1,nqs-1
81          ndqr(i,k)=ndqr(i,k+1)
          ndqr(i,nqs)=noutfix(i)
          anqr(i)=anqr(i)+(ndqr(i,nqs)-ndqr(i,1))/float(nqs)
          nqaccum(i)=nqs
        endif
      else
        do 80 k=1,nqs
80        ndqr(i,k)=0.0
        anqr(i)=99999999.
        nqaccum(i)=0
      endif
      min = (tii*1500./60.)*(xl(i)/s(i))
      anqr(i) = max(anqr(i),min)
      expquet(i) = ndque(i)*tii/anqr(i)
50    continue
c
c     also calculate the expected travel times from the
c     beginning point of each link and store them in statmpt
c     array (static map times).
c     for cross-over links we store their own trip time only.
c
      do 41 i=1,n
      statmpt(i)=s(i)/v(i)+expquet(i)
41    continue
c
113   return
      end
      subroutine change(jjj,iiii,idy,xl,i)
      common/ini7/ratio(1,1781,3),nupuls(1,1781)
      common/ini13/time1(1,1781,3)
      common/ini14/time2(1,1781,3)
      dimension xl(1781)
      xl(i)=xl(i)*ratio(idy,jjj,iiii)
      return
      end
      subroutine restor(jjj,iiii,idy,xl,i)
      common/ini7/ratio(1,1781,3),nupuls(1,1781)
      common/ini13/time1(1,1781,3)
      common/ini14/time2(1,1781,3)
      dimension xl(1781)
      xl(i)=xl(i)/ratio(idy,jjj,iiii)
      return
      end
      subroutine getlink(t,ttt,j,i,key)
```

```
      common/jay1/link(1781,8),klink(1781,7),statmpt(1781)
     *        ,inlink(1781,8),intooi(1781,676,3),intoo(1781)
      common/jay2/tiinow(69000),jdest(69000),jpath(69000,90)
     *        ,ribf(69000),icurmt(69000),nottin(69000)

common/jay3/info(69000),itag(69000),jsw(69000),nexlink(690
00)

common/jay4/starttm,endtm,numcars,iseed,fracinf,ribfa,bound

common/jn1/ifwdarc(1779,2),npoint(661),idests(32),nreach(661)

common/jn2/iwork1(1779),iwork2(1779),iunod(1779),idnod(17
79)
      common/jn3/lastnod(32,661,12,2),dist(32,661,12),
     *        lastno1(32,661,12,2),lastk(32,661),kclos(32,661)
      common/jzone/izlins(36),izlin(36,401),zdem(36,36,10)
     *
,izone(661),ioz(36),idz(36),ipz(36),npz(801),zfdem(36,36,10)
     * ,ildem(1781),ztdem(36,10),int,totimz(36)
     * ,expgenz(36),expgen(1781)
c
c     initialize the route-switch indicator.
      jsw(j)=0
c
c     first, the case of the no-information vehicles.
c     just use the route stored when they started the trip.
      if(info(j).eq.0) then
c      copy the next location from the jpath list
       nexnod=jpath(j,icurmt(j)+1)
       do 2 k=npoint(nexnod),npoint(nexnod+1)-1
       if(idnod(i).eq.ifwdarc(k,1)) nexlink(j)=ifwdarc(k,2)
2       continue
      else
c      calculate the trip time on the current path.
       tmthis=0.0
       do 3 k=icurmt(j)+1,90
       nexnod=jpath(j,k)
       do 4 l=npoint(nexnod),npoint(nexnod+1)-1
       if(jpath(j,k-1).eq.ifwdarc(l,1))
     *  tmthis=tmthis+statmpt(ifwdarc(l,2))
4      continue
       if(nexnod.eq.idests(jdest(j))) go to 5
3      continue
c      find out which of the k-shortest paths is currently
c      the best path.
5      nodcur=jpath(j,icurmt(j))
       best=99999.0
       do 10 l=1,kay
       if(dist(jdest(j),jpath(j,icurmt(j)),l).lt.best) then
        best=dist(jdest(j),jpath(j,icurmt(j)),l)
        ibest=l
       endif
10     continue
c      if alternative route is selected to switch to, then
c      copy the jpath() with that route. otherwise, just pick
c      the link according to the current jpath.
       if((best.lt.tmthis*(1-ribf(j))).and.(best.lt.tmthis-bound))
     * then
       jsw(j)=1
       jpath(j,1)=jpath(j,icurmt(j))
       do 20 k=2,90
       jpath(j,k)=lastnod(jdest(j),jpath(j,k-1),ibest,1)
       ibest=lastnod(jdest(j),jpath(j,k-1),ibest,2)
       if(jpath(j,k).eq.idests(jdest(j))) go to 21
20      continue
21      icurmt(j)=1
       do 6 k=npoint(jpath(j,2)),npoint(jpath(j,2)+1)-1
       if(idnod(i).eq.ifwdarc(k,1))
     *  nexlink(j)=ifwdarc(k,2)
6      continue
      else
       do 7 k=npoint(jpath(j,icurmt(j)+1)),
     *      npoint(jpath(j,icurmt(j)+1)+1)-1
       if(idnod(i).eq.ifwdarc(k,1)) nexlink(j)=ifwdarc(k,2)
7      continue
      endif
      endif
115   return
      end
      subroutine beginr(initcop,j,i,kay)
      dimension r(10)
      common/jay1/link(1781,8),klink(1781,7),statmpt(1781)
     *        ,inlink(1781,8),intooi(1781,676,3),intoo(1781)
      common/jay2/tiinow(69000),jdest(69000),jpath(69000,90)
     *        ,ribf(69000),icurmt(69000),nottin(69000)

common/jay3/info(69000),itag(69000),jsw(69000),nexlink(690
00)

common/jay4/starttm,endtm,numcars,iseed,fracinf,ribfa,bound
      common/jay5/influx(7,1781),limflux(7,1781),cayvee(7)

common/jn1/ifwdarc(1779,2),npoint(661),idests(32),nreach(661)

common/jn2/iwork1(1779),iwork2(1779),iunod(1779),idnod(17
79)
      common/jn3/lastnod(32,661,12,2),dist(32,661,12),
     *        lastno1(32,661,12,2),lastk(32,661),kclos(32,661)
      common/jn4/hplist1(32,6610),iplist1(32,6610),
     *        nhpoint(32,661,12),iplist2(32,6610)
      common/jn5/nkorder(32,661,12),lastord(32,661)
c
c     if the initial path is to be the best path after start-up
c     fix ipinit as 1. if they are to be any random path from
c     the best 10, fix it as 0.
c
      data ipinit/0/
c
      nr=10
      jpath(j,1)=idnod(i)
c
```

```
c    if the vehicles are send to the shortest of the paths
c    at the end of start-up time, then...
c
     if(ipinit.eq.1) then
      ibest=1
c
c    otherwise, pick one route at random..
c
     else
      call mun(nr,r)
      ipaths=kclos(jdest(j),jpath(j,1))
      ibest=ifix(r(5)*ipaths)+1
      if(ibest.gt.ipaths)ibest=ipaths
     endif
     do 20 k=2,90
     if(initcop.eq.0) then
     jpath(j,k)=lastnod(jdest(j),jpath(j,k-1),ibest,1)
     ibest=lastnod(jdest(j),jpath(j,k-1),ibest,2)
     else
     jpath(j,k)=lastno1(jdest(j),jpath(j,k-1),ibest,1)
     ibest=lastno1(jdest(j),jpath(j,k-1),ibest,2)
     endif
     if(jpath(j,k).eq.idests(jdest(j))) go to 21
20    continue
21    icurrnt(j)=1
     return
     end
     subroutine kshort(narcs,nnodes,ndests,kay,n,t)
     common/jay1/link(1781,8),klink(1781,7),statmpt(1781)
    *          ,inlink(1781,8),intooi(1781,676,3),intoo(1781)

common/jn1/ifwdarc(1779,2),npoint(661),idests(32),nreach(661)

common/jn2/iwork1(1779),iwork2(1779),iunod(1779),idnod(17
79)
     common/jn3/lastnod(32,661,12,2),dist(32,661,12),
    *          lastno1(32,661,12,2),lastk(32,661),kclos(32,661)
     common/jn4/hplist1(32,6610),iplist1(32,6610),
    *          nhpoint(32,661,12),iplist2(32,6610)
     common/jn5/nkorder(32,661,12),lastord(32,661)
     dimension nodup(3)
c    data (idests(i),i=1,10)/814,218,819,245,834,845,
c   *                  829,265,805,345/
c
c    the dimensions assume 660 nodes, 1781 arcs,

c    32 destinations and 10 shortest-paths..
c
     do 2010 i=1,660
     if(npoint(i).eq.0) go to 2010
     inum = npoint(i+1)-npoint(i)
     if(inum.gt.1) then
     do 2030 l=1,inum-1
     arcmin = statmpt(ifwdarc(npoint(i)+l-1,2))
     mflag  = 0
     do 2020 j=npoint(i)+l,npoint(i)+inum-1
       if(statmpt(ifwdarc(j,2)).lt.arcmin) then
         arcmin = statmpt(ifwdarc(j,2))
         ipos   = j
         mflag  = 1
       endif
2020  continue
     if(mflag.ne.0) then
c    swap the numbers.. smaller number backward..
     iarcmin=ifwdarc(ipos,1)
     linkmin=ifwdarc(ipos,2)
     ifwdarc(ipos,1)=ifwdarc(npoint(i)+l-1,1)
     ifwdarc(ipos,2)=ifwdarc(npoint(i)+l-1,2)
     ifwdarc(npoint(i)+l-1,1)=iarcmin
     ifwdarc(npoint(i)+l-1,2)=linkmin
     endif
2030  continue
     endif
2010  continue
c
1030  do 1040 i=1,ndests
     do 1035 k=1,kay
1035  lastord(i,k)=0
     do 1040 j=1,660
     lastk(i,j)=0
     kclos(i,j)=0
     do 1040 k=1,kay
      lastnod(i,j,k,1)=0
      lastnod(i,j,k,2)=0
      dist(i,j,k)=99999.0
1040  continue
c
c
c    fill up the heap-lists with large number
     do 1060 i=1,ndests
     do 1060 j=1,660*kay
1060  hplist1(i,j)=88888.0
c
c
c    fix up the first heap.. the forward-star of the
c    destination to start with.
c
     do 3000 ides=1,ndests
     idest=idests(ides)
     inum=npoint(idest+1)-npoint(idest)
     do 1070 i=1,inum
      hplist1(ides,i)=statmpt(ifwdarc(npoint(idest)+i-1,2))
      nhpoint(ides,ifwdarc(npoint(idest)+i-1,1),1) = i
      lastnod(ides,ifwdarc(npoint(idest)+i-1,1),1,1) = idest
      lastnod(ides,ifwdarc(npoint(idest)+i-1,1),1,2) = 1
      lastk(ides,ifwdarc(npoint(idest)+i-1,1)) =
    * lastk(ides,ifwdarc(npoint(idest)+i-1,1)) + 1
      dist(ides,ifwdarc(npoint(idest)+i-1,1),1) =
    *     hplist1(ides,i)
      iplist1(ides,i)=ifwdarc(npoint(idest)+i-1,1)
      iplist2(ides,i)=1
1070  continue
```

```
c
1210 nnext=inum+1
c
c    delete the minimum from the heap.
1330 if(hplist1(ides,1).gt.88887.0) go to 1340
     nodclos=iplist1(ides,1)
     kclos(ides,nodclos)=kclos(ides,nodclos)+1
     lastord(ides,kclos(ides,nodclos))=
    *lastord(ides,kclos(ides,nodclos))+1
     nkorder(ides,lastord(ides,kclos(ides,nodclos)),
    *kclos(ides,nodclos)) = nodclos
c
c    store the string of 3 nodes for checking loops.
c
     knode=lastnod(ides,nodclos,kclos(ides,nodclos),1)
     kpath=lastnod(ides,nodclos,kclos(ides,nodclos),2)
     do 1228 nj=1,3
     nodup(nj)=knode
     if(nj.lt.3) then
     if(knode.eq.idests(ides)) then
      do 1229 nnj=nj+1,3
1229  nodup(nnj)=0
      go to 1221
     endif
     endif
     knode1=knode
     knode=lastnod(ides,knode1,kpath,1)
     kpath=lastnod(ides,knode1,kpath,2)
1228 continue
c
c    re-form the heap. put a large-number at the heap-top,
c    compare with the two children, swap positions with the
c    smaller one, look at the new children, and thus push
c    the large-number down.
c
1221 hplist1(ides,1)=88888.0
     large=1
     if(nnext.le.1) go to 1225
     ireform=0
     bigno=88888.0
1220 minchd=large*2
     if(hplist1(ides,large*2).eq.hplist1(ides,large*2+1)) then
      if(iplist2(ides,large*2).gt.iplist2(ides,large*2+1))
    * minchd=large*2+1
     endif
     if(hplist1(ides,large*2).gt.hplist1(ides,large*2+1))
    *minchd=large*2+1
     if(bigno.gt.hplist1(ides,minchd)) then
       hplist1(ides,large)=hplist1(ides,minchd)
       iplist1(ides,large)=iplist1(ides,minchd)
       iplist2(ides,large)=iplist2(ides,minchd)
       nhpoint(ides,iplist1(ides,minchd),iplist2(ides,minchd))
    *  = large
       hplist1(ides,minchd)=88888.0
       large=minchd
     else
       ireform=1
     endif
     if(ireform.lt.1) go to 1220
1225 continue
c
c    look at the forward star nodes of the just-deleted node,
c    add the arc costs to the distance to the deleted node,
c    compare with the current cost to those nodes, and decide
c    if they reduce. if they do, compare with its current
c    parent node in the heap, and swap positions, till it
c    finds a parent that is not bigger than it.
c
     if((npoint(nodclos).lt.1).or.(npoint(nodclos+1).eq.
    *  npoint(nodclos))) go to 1380
     do 1230 i=npoint(nodclos),npoint(nodclos+1)-1
     newnode=ifwdarc(i,1)
     if(kclos(ides,newnode).eq.kay) go to 1230
c
c    check up the node string to prevent short loops.
c
     do 1231 nj=1,3
     if(newnode.eq.nodup(nj)) go to 1230
1231 continue
c
c
     if(lastk(ides,newnode).lt.kay) then
      if(lastk(ides,newnode).eq.kclos(ides,newnode)) then
       nl=lastk(ides,newnode)+1
       l=large
       lastk(ides,newnode)=nl
      else
       nl=kclos(ides,newnode)+1
1250   if(dist(ides,nodclos,kclos(ides,nodclos)) +
    *  statmpt(ifwdarc(i,2)) .lt. dist(ides,newnode,nl)) then
        do 1240 j=lastk(ides,newnode),nl,-1
         iplist2(ides,nhpoint(ides,newnode,j)) =
    *    iplist2(ides,nhpoint(ides,newnode,j)) + 1
         nhpoint(ides,newnode,j+1)=nhpoint(ides,newnode,j)
         dist(ides,newnode,j+1)=dist(ides,newnode,j)
         lastnod(ides,newnode,j+1,1)=lastnod(ides,newnode,j,1)
         lastnod(ides,newnode,j+1,2)=lastnod(ides,newnode,j,2)
1240    continue
        lastk(ides,newnode)=lastk(ides,newnode)+1
        l=large
        go to 1260
       else
        if(nl.eq.lastk(ides,newnode)) then
         nl=nl+1
         lastk(ides,newnode)=nl
         l=large
         go to 1260
        endif
        nl=nl+1
        go to 1250
       endif
      endif
```

```
      else
       if(lastk(ides,newnode).eq.kclos(ides,newnode)) then
        go to 1270
       else
        nl=kclos(ides,newnode)+1
1255    if(dist(ides,nodclos,kclos(ides,nodclos)) +
     *  statmpt(ifwdarc(i,2)) .lt. dist(ides,newnode,nl)) then
        l=nhpoint(ides,newnode,kay)
        if(nl.lt.kay) then
        do 1245 j=kay-1,nl,-1
         iplist2(ides,nhpoint(ides,newnode,j)) =
     *   iplist2(ides,nhpoint(ides,newnode,j)) + 1
         nhpoint(ides,newnode,j+1)=nhpoint(ides,newnode,j)
         dist(ides,newnode,j+1)=dist(ides,newnode,j)
         lastnod(ides,newnode,j+1,1)=lastnod(ides,newnode,j,1)
         lastnod(ides,newnode,j+1,2)=lastnod(ides,newnode,j,2)
1245    continue
        endif
        go to 1260
       else
        nl=nl+1
        if(nl.le.kay) go to 1255
        go to 1270
       endif
      endif
      endif
c
1260  if(l.eq.0) then
       l=nnext
       nnext=nnext+1
      else
       large=0
      endif
      hplist1(ides,l)=statmpt(ifwdarc(i,2))+
     * dist(ides,nodclos,kclos(ides,nodclos))
      iplist1(ides,l)=newnode
      iplist2(ides,l)=nl
      nhpoint(ides,newnode,nl)=l
      lastnod(ides,newnode,nl,1)=nodclos
      lastnod(ides,newnode,nl,2)=kclos(ides,nodclos)
      dist(ides,newnode,nl)=hplist1(ides,l)
c     compare the parent, exchange and move up the heap,if
needed**
      jk=l
       if(jk.eq.1) go to 1290
1280    if(hplist1(ides,jk).ge.hplist1(ides,jk/2)) go to 1290
         a=hplist1(ides,jk)
         ib=iplist1(ides,jk)
         ic=iplist2(ides,jk)
         hplist1(ides,jk)=hplist1(ides,jk/2)
         iplist1(ides,jk)=iplist1(ides,jk/2)
         iplist2(ides,jk)=iplist2(ides,jk/2)
         if(hplist1(ides,jk).lt.88880)
     *   nhpoint(ides,iplist1(ides,jk),iplist2(ides,jk))=jk
         hplist1(ides,jk/2)=a
         iplist1(ides,jk/2)=ib
         iplist2(ides,jk/2)=ic
         if(a.lt.88880.0) nhpoint(ides,ib,ic)=jk/2
         jk=jk/2
        if(jk.gt.1) go to 1280
1290    continue
c
c
1270  continue
c
c
c
1230  continue
1380  go to 1330
1340  if(ides.lt.0) then
       do 1360 i=1,660
       write(4,73) i,((dist(ides,i,j),(lastnod(ides,i,j,k),k=1,2)),
     * j=1,kay)
73    format(1x,i3,10(f5.1,i4,i3))
1360  continue
       endif
3000  continue
1370  if(t.lt.0.05) then
       do 3010 i=1,ndests
       do 3010 j=1,660
       dist(i,j,11)=dist(i,j,1)
       lastnod(i,j,11,1)=lastnod(i,j,1,1)
       lastnod(i,j,11,2)=lastnod(i,j,1,2)
3010  continue
       endif
       return
       end
       subroutine addchain(l,linkmax,narcs,n)
       common/ini10/s(1781),v(1781),nlanes(1781),nmov(1781)
       common/jay1/link(1781,8),klink(1781,7),statmpt(1781)
     *          ,inlink(1781,8),intooi(1781,676,3),intoo(1781)

common/jn1/ifwdarc(1779,2),npoint(661),idests(32),nreach(661)

common/jn2/iwork1(1779),iwork2(1779),junod(1779),idnod(17
79)
c
c      find the current trip time on each arc, by adding
c      up the times on the link-chain of that arc.
c
c      the following block loops over the arcs and adds up one

c      link along the chain every time.  written in this form
c      for vector processing of the inner loops.
c
c      start with the trip time on the first link of each chain.
c
       do 251 ia=1,narcs
251   iwork2(ia)=ifwdarc(ia,2)
c
       do 254 kq=1,linkmax
c
```

```
      do 250 ia=1,narcs
250     iwork1(ia)=iwork2(ia)
c
      do 253 ia=1,narcs
253     statmpt(ia)=statmpt(ia)+c(iwork1(ia))/v(iwork1(ia))
c
      do 252 ia=1,narcs
      iwork2(ia)=n+1
      if(link(iwork1(ia),8).lt.2)iwork2(ia)=link(iwork1(ia),2)
252     continue
c
254   continue
c
c
      return
      end
      subroutine routstm(narcs,nnodes,ndests,kay,n)
      common/jay1/link(1781,8),klink(1781,7),statmpt(1781)
     *          ,inlink(1781,8),intooi(1781,676,3),intoo(1781)
      common/jn1/ifwdarc(1779,2),npoint(661),idests(32),nreach(661)
      common/jn3/lastnod(32,661,12,2),dist(32,661,12),
     *       lastno1(32,661,12,2),lastk(32,661),kclos(32,661)
      common/jn5/nkorder(32,661,12),lastord(32,661)
      do 4000 ides=1,ndests
      idest=idests(ides)
      do 4010 k=1,kay
      do 4010 j=lastord(ides,k),1,-1
      nodcur = nkorder(ides,j,k)
      if(dist(ides,nodcur,k).lt.0.) go to 4010
      nodcur1= nodcur
      dist(ides,nodcur1,k) = 0.0
4015  nodpre = lastnod(ides,nodcur,k,1)
      nodprek= lastnod(ides,nodcur,k,2)
      if(nodpre.eq.idest) then
      do 4020 ip=npoint(nodpre),npoint(nodpre+1)-1
       if(ifwdarc(ip,1).eq.nodcur) then
        dist(ides,nodcur1,k)=dist(ides,nodcur1,k) +
     *                 statmpt(ifwdarc(ip,2))
       go to 4060
      endif
4020  continue
      endif
      if(dist(ides,nodpre,nodprek).lt.0.) then
      do 4030 ip=npoint(nodpre),npoint(nodpre+1)-1
       if(ifwdarc(ip,1).eq.nodcur) then
        dist(ides,nodcur1,k)=dist(ides,nodcur1,k) +
     *                 statmpt(ifwdarc(ip,2)) -
     *                 dist(ides,nodpre,nodprek)
       go to 4060
       endif
4030  continue
      endif
      do 4040 ip=npoint(nodpre),npoint(nodpre+1)-1
       if(ifwdarc(ip,1).eq.nodcur) then
        dist(ides,nodcur1,k)=dist(ides,nodcur1,k) +
     *                 statmpt(ifwdarc(ip,2))
       nodcur=nodpre
       go to 4015
      endif
4040  continue
4060  dist(ides,nodcur1,k)=-dist(ides,nodcur1,k)
      nodcur=nodcur1
4065  nodpre=lastnod(ides,nodcur,k,1)
      nodprek=lastnod(ides,nodcur,k,2)
      if(nodpre.ne.idest) then
      if(dist(ides,nodpre,nodprek).ge.0.) then
       do 4070 ip=npoint(nodpre),npoint(nodpre+1)-1
       if(ifwdarc(ip,1).eq.nodcur) then
        dist(ides,nodpre,k)=dist(ides,nodcur,k) +
     *                 statmpt(ifwdarc(ip,2))
       nodcur=nodpre
       go to 4065
       endif
4070  continue
      endif
      endif
4010  continue
4000  continue
      do 3000 i=1,ndests
      do 3000 k=1,kay
      do 3000 j=1,660
      dist(i,j,k)=abs(dist(i,j,k))
3000  continue
      return
      end
```

APPENDIX - B (Input data : Austin network)

36 635 1776 1776 32 10
201 202 203 204 205 206 207 208 360 356
358 359 362 363 357 371 372 386 375 376
385 374 377 384 373 378 379 383 380 381
382 404 405 406 407 408
571 576 581 586 0 0 0 0 555 550
456 460 448 529 403 259 109 497 371 7
380 319 37 381 39 453 474 230 111 103
40 41 28 31 35 5
1 408
2 408
3 408
4 408
5 408
6 408
8 408
9 408
10 408
11 406
12 406
13 406
14 408
15 408
16 408
17 408
18 408
19 406
20 406
21 406
22 406
23 407
24 407
25 404
26 404
27 405
28 405
29 405
30 405
31 406
32 406
33 406
34 407
35 407
36 407
38 371
41 404
42 404
43 405
44 405
45 405
46 405
47 406
48 406
49 406
50 407
51 407
52 407
53 404
54 404
55 404
56 405
57 405
58 405
59 405
60 406
61 406
62 406
63 407
64 407
65 407
66 371
67 371
68 371
69 372
70 371
71 371
72 371
73 371
74 371
75 371
76 371
77 372
78 373
79 373
80 373
81 373
82 380
83 380
84 380
85 380
86 381
87 381
88 381
89 382
90 382
91 382
92 371
93 371
94 371
95 371
96 371
97 371
98 371
99 371

100	371
101	371
102	371
104	371
105	371
106	371
107	371
108	371
109	372
110	372
112	373
113	373
114	373
115	373
116	380
117	380
118	380
119	380
120	381
121	381
122	381
123	382
124	382
125	382
126	371
127	371
128	371
129	371
130	371
131	371
132	371
133	371
134	371
135	371
136	371
137	371
138	371
139	371
140	371
141	371
142	371
143	371
144	373
145	373
146	373
147	373
148	380
149	380
150	380
151	380
152	381
153	381
154	381
155	382
156	382
157	382
158	373
159	373
160	373
161	373
162	380
163	380
164	380
165	380
166	381
167	381
168	381
169	382
170	382
171	382
172	371
173	371
174	371
175	371
176	371
177	371
178	371
179	371
180	371
181	371
182	371
183	371
184	371
185	372
186	372
187	373
188	373
189	373
190	373
191	378
192	378
193	378
194	378
195	379
196	379
197	379
198	383
199	383
200	383
201	371
202	371
203	371
204	371
205	371
206	371
207	371
208	371
209	371
210	371
211	371
212	371
213	371

214	371
215	371
216	372
217	371
218	373
219	373
220	373
221	373
222	378
223	378
224	378
225	378
226	379
227	379
228	379
229	383
230	383
231	383
232	371
233	371
234	373
235	373
236	373
237	373
238	378
239	378
240	378
241	378
242	379
243	379
244	379
245	383
246	383
247	371
248	357
249	371
250	357
251	371
252	371
253	371
254	371
255	371
256	371
257	371
258	371
259	371
260	371
261	371
262	371
263	357
264	371
265	372
266	373
267	373
268	373
269	373
270	373
271	378
272	378
273	378
274	375
275	378
276	377
277	377
278	377
279	379
280	379
281	383
282	383
283	377
284	377
285	371
286	371
287	371
288	371
289	374
290	374
291	374
292	374
293	374
294	377
295	377
296	377
297	377
298	371
299	371
300	371
301	371
302	371
303	371
304	371
305	371
306	371
307	371
308	371
309	371
310	371
311	371
312	371
313	371
314	371
315	371
316	371
317	374
318	374
319	374
320	374
321	377
322	377
323	377
324	377
325	377

326 377
327 377
328 371
329 357
330 383
331 371
332 371
333 371
334 371
335 371
336 371
337 371
338 371
339 371
340 371
341 374
342 374
343 374
344 374
345 377
346 377
347 377
348 377
349 377
350 377
351 371
352 357
353 357
354 357
355 357
356 357
357 357
358 375
359 375
360 375
361 375
362 376
363 376
364 376
365 376
366 376
367 376
368 385
369 375
370 375
371 375
372 375
373 373
374 376
375 376
376 376
377 376
378 376
379 376
380 385
381 384
382 384
383 357
384 375
385 375
386 375
387 375
388 375
389 375
390 376
391 376
392 376
393 376
394 376
395 376
396 385
397 357
398 357
399 357
400 357
401 357
402 357
403 357
404 375
405 357
406 357
407 357
408 357
409 362
410 357
411 357
412 357
413 357
414 358
415 375
416 375
417 375
418 375
419 375
420 375
421 375
422 375
423 375
424 375
425 375
426 376
427 376
428 376
429 376
430 376
431 376
432 362
433 376
434 376
435 376
436 385
437 358

438	358
439	359
440	359
441	359
442	362
443	362
444	362
445	362
446	362
447	362
449	362
450	362
451	375
452	371
454	358
455	358
456	358
457	358
458	358
459	359
460	359
461	359
462	362
463	357
464	357
465	357
466	357
467	357
468	357
469	357
470	357
471	357
472	357
473	357
477	357
478	357
479	358
480	358
481	358
482	358
483	358
484	358
485	358
486	358
487	358
488	358
489	358
490	359
491	359
492	359
493	362
494	362
495	363
496	386
497	386
498	360
499	359
500	357
501	357
502	357
503	362
505	357
506	356
507	357
509	357
510	358
511	358
512	358
513	358
514	358
515	358
516	359
517	360
518	359
519	360
520	360
521	360
522	362
525	362
526	362
527	363
528	363
529	363
530	363
531	362
532	356
533	356
534	356
535	356
536	356
537	356
538	356
539	360
540	360
541	360
542	360
543	360
544	362
545	356
546	356
547	356
548	356
549	356
550	356
551	356
552	356
553	356
554	360
555	360
556	360
557	360
558	360

559 362
560 362
561 362
562 362
563 362
564 363
565 363
566 363
567 363
568 363
602 207
603 207
604 207
605 207
609 207
610 207
611 207
612 207
613 207
619 207
620 207
621 207
622 207
623 207
630 207
631 207
632 207
633 207
634 207
654 207
655 206
653 206
652 206
651 206
650 206
649 206
647 206
646 206
645 206
644 206
642 206
641 206
640 206
639 206
638 206
637 206
636 205
635 205
629 205
628 205
627 205
626 205
625 205
624 205
618 205
617 205
616 205
614 208
615 208
608 208
607 208
656 208
606 208
571 201
572 201
573 201
574 201
575 201
576 202
577 202
578 202
579 202
580 202
581 203
582 203
583 203
584 203
585 203
586 204
587 204
588 204
589 204
590 204
111 380
103 381
40 382
39 373
37 377
7 376
448 362
453 378
474 379
1 5 1650 2 2 0.583
5 1 1650 2 2 0.583
3 4 300 2 2 0.583
4 3 300 2 2 0.583
4 5 300 2 2 0.583
5 4 300 2 2 0.583
5 6 600 2 2 0.583
6 5 600 2 2 0.583
8 9 900 2 2 0.583
9 8 900 2 2 0.583
9 10 500 2 2 0.583
10 9 500 2 2 0.583
12 13 350 2 2 0.583
13 12 350 2 2 0.583
14 15 350 2 2 0.583
15 16 350 2 2 0.583
16 17 350 2 2 0.583
17 18 375 2 2 0.583
18 19 375 2 2 0.583
19 20 350 2 2 0.583

20	21	350	2	2	0.583
21	22	350	2	2	0.583
22	23	350	2	2	0.583
23	24	350	2	2	0.583
11	12	350	2	2	0.583
12	11	350	2	2	0.583
26	25	375	2	2	0.583
27	26	375	2	2	0.583
28	27	375	2	2	0.583
29	28	380	2	2	0.583
30	29	410	2	2	0.583
31	30	405	2	2	0.583
32	31	375	2	2	0.583
33	32	375	2	2	0.583
34	33	375	2	2	0.583
35	34	375	2	2	0.583
36	35	375	2	2	0.583
41	42	375	2	2	0.583
42	41	375	2	2	0.583
42	43	375	2	2	0.583
43	42	375	2	2	0.583
43	44	375	2	2	0.583
44	43	375	2	2	0.583
44	45	380	2	2	0.583
45	44	380	2	2	0.583
45	46	410	2	2	0.583
46	45	410	2	2	0.583
46	47	405	2	2	0.583
47	46	405	2	2	0.583
47	48	375	2	2	0.583
48	47	375	2	2	0.583
48	49	375	2	2	0.583
49	48	375	2	2	0.583
49	50	375	2	2	0.583
50	49	375	2	2	0.583
50	51	375	2	2	0.583
51	50	375	2	2	0.583
51	52	375	2	2	0.583
52	51	375	2	2	0.583
53	54	375	2	2	0.583
54	53	375	2	2	0.583
54	55	375	2	2	0.583
55	54	375	2	2	0.583
55	56	375	2	2	0.583
56	55	375	2	2	0.583
56	57	375	2	2	0.583
57	56	375	2	2	0.583
57	58	380	2	2	0.583
58	57	380	2	2	0.583
58	59	410	2	2	0.583
59	58	410	2	2	0.583
59	60	405	2	2	0.583
60	59	405	2	2	0.583
60	61	375	2	2	0.583
61	60	375	2	2	0.583
61	62	375	2	2	0.583
62	61	375	2	2	0.583
62	63	375	2	2	0.583
63	62	375	2	2	0.583
63	64	375	2	2	0.583
64	63	375	2	2	0.583
64	65	375	2	2	0.583
65	64	375	2	2	0.583
66	67	700	2	3	0.750
67	68	240	2	3	0.750
68	70	400	2	3	0.750
70	71	200	2	3	0.750
71	73	960	2	3	0.750
73	75	780	2	3	0.750
75	76	550	2	3	0.750
76	77	500	2	3	0.750
77	78	695	2	3	0.750
78	79	425	2	3	0.750
79	80	375	2	3	0.750
80	81	375	2	3	0.750
81	82	375	2	3	0.750
82	83	375	2	3	0.750
83	84	380	2	3	0.750
84	85	410	2	3	0.750
85	86	405	2	3	0.750
86	87	375	2	3	0.750
87	88	375	2	3	0.750
88	89	375	2	3	0.750
89	90	375	2	3	0.750
90	91	375	2	3	0.750
93	92	375	2	2	0.583
94	93	650	2	2	0.583
97	94	225	2	2	0.583
96	95	240	2	2	0.583
98	97	200	2	2	0.583
99	98	200	2	2	0.583
100	99	50	2	2	0.583
101	100	180	2	2	0.583
102	101	290	2	2	0.583
104	102	600	2	2	0.583
105	104	150	2	2	0.583
106	105	600	2	2	0.583
107	106	110	2	2	0.583
108	107	380	2	2	0.583
109	108	350	2	2	0.583
110	109	175	2	2	0.583
112	110	600	2	2	0.583
113	112	430	2	2	0.583
114	113	375	2	2	0.583
115	114	375	2	2	0.583
116	115	375	2	2	0.583
117	116	375	2	2	0.583
118	117	380	2	2	0.583
119	118	410	2	2	0.583
120	119	405	2	2	0.583
121	120	375	2	2	0.583
122	121	375	2	2	0.583

123	122	375	2 2	0.583
124	123	375	2 2	0.583
125	124	375	2 2	0.583
95	96	240	2 2	0.583
144	145	425	2 2	0.583
145	146	375	2 2	0.583
146	147	375	2 2	0.583
147	148	375	2 2	0.583
148	149	375	2 2	0.583
149	150	380	2 2	0.583
150	151	410	2 2	0.583
151	152	405	2 2	0.583
152	153	375	2 2	0.583
153	154	375	2 2	0.583
154	155	375	2 2	0.583
155	156	375	2 2	0.583
156	157	375	2 2	0.583
145	144	425	2 2	0.583
146	145	375	2 2	0.583
147	146	375	2 2	0.583
159	158	425	2 2	0.583
160	159	375	2 2	0.583
161	160	375	2 2	0.583
162	161	375	2 2	0.583
163	162	375	2 2	0.583
164	163	380	2 2	0.583
165	164	410	2 2	0.583
166	165	405	2 2	0.583
167	166	375	2 2	0.583
168	167	375	2 2	0.583
169	168	375	2 2	0.583
170	169	375	2 2	0.583
171	170	375	2 2	0.583
128	129	750	2 2	0.583
129	128	750	2 2	0.583
128	127	425	2 2	0.583
127	128	425	2 2	0.583
127	129	400	2 2	0.583
129	127	400	2 2	0.583
129	92	350	2 2	0.583
92	129	350	2 2	0.583
128	173	225	2 2	0.583
173	128	225	2 2	0.583
66	93	335	2 2	0.583
93	66	335	2 2	0.583
93	131	350	2 2	0.583
131	93	350	2 2	0.583
131	72	700	2 2	0.583
72	131	700	2 2	0.583
72	74	325	2 2	0.583
74	72	325	2 2	0.583
74	130	600	2 2	0.583
130	74	600	2 2	0.583
130	131	325	2 2	0.583
131	130	325	2 2	0.583
133	134	200	2 2	0.583
134	133	200	2 2	0.583
138	135	200	2 2	0.583
135	138	200	2 2	0.583
137	136	200	2 2	0.583
136	137	200	2 2	0.583
180	140	250	2 2	0.583
140	180	250	2 2	0.583
140	141	275	2 2	0.583
141	140	275	2 2	0.583
142	143	725	2 2	0.583
143	142	725	2 2	0.583
175	176	375	2 2	0.583
176	175	375	2 2	0.583
176	177	190	2 2	0.583
177	176	190	2 2	0.583
177	178	220	2 2	0.583
178	177	220	2 2	0.583
178	179	375	2 2	0.583
179	178	375	2 2	0.583
179	180	200	2 2	0.583
180	179	200	2 2	0.583
180	181	250	2 2	0.583
181	180	250	2 2	0.583
181	182	180	2 2	0.583
182	181	180	2 2	0.583
183	184	720	2 2	0.583
184	183	720	2 2	0.583
184	185	370	2 2	0.583
185	184	370	2 2	0.583
185	186	370	2 3	0.750
186	187	710	2 3	0.750
187	188	425	2 3	0.750
188	189	375	2 3	0.750
189	190	375	2 3	0.750
190	191	375	2 3	0.750
191	192	375	2 3	0.750
192	193	380	2 3	0.750
193	194	410	2 3	0.750
194	195	405	2 3	0.750
195	196	375	2 3	0.750
196	197	375	2 3	0.750
197	198	375	2 3	0.750
198	199	375	2 3	0.750
199	200	375	2 3	0.750
172	173	290	2 2	0.583
173	172	290	2 2	0.583
173	126	110	2 2	0.583
126	173	110	2 2	0.583
172	201	520	2 2	0.583
201	172	520	2 2	0.583
126	202	520	2 2	0.583
202	126	520	2 2	0.583
201	202	400	2 2	0.583
202	201	400	2 2	0.583
202	203	140	2 2	0.583
203	202	140	2 2	0.583

203	204	700 2	2	0.583
204	203	700 2	2	0.583
204	232	630 2	2	0.583
232	204	630 2	2	0.583
206	207	205 2	2	0.583
207	206	205 2	2	0.583
207	208	315 2	2	0.583
208	207	315 2	2	0.583
208	209	300 2	2	0.583
209	208	300 2	2	0.583
210	209	265 2	2	0.583
209	210	265 2	2	0.583
210	211	310 2	2	0.583
211	210	310 2	2	0.583
211	212	170 2	2	0.583
212	211	170 2	2	0.583
212	213	300 2	2	0.583
213	212	300 2	2	0.583
214	217	370 2	2	0.583
217	214	370 2	2	0.583
219	218	425 2	3	0.750
218	217	1020 2	3	0.750
220	219	375 2	3	0.750
221	220	375 2	3	0.750
222	221	375 2	3	0.750
223	222	375 2	3	0.750
224	223	380 2	3	0.750
225	224	410 2	3	0.750
226	225	405 2	3	0.750
227	226	375 2	3	0.750
228	227	375 2	3	0.750
229	228	375 2	3	0.750
230	229	375 2	3	0.750
231	230	375 2	3	0.750
251	252	690 2	2	0.583
252	251	690 2	2	0.583
233	253	655 2	2	0.583
253	233	655 2	2	0.583
254	255	200 2	2	0.583
255	254	200 2	2	0.583
255	256	140 2	2	0.583
256	255	140 2	2	0.583
256	257	170 2	2	0.583
257	256	170 2	2	0.583
257	258	95 2	2	0.583
258	257	95 2	2	0.583
258	259	240 2	2	0.583
259	258	240 2	2	0.583
253	254	175 2	2	0.583
254	253	175 2	2	0.583
259	260	305 2	2	0.583
260	259	305 2	2	0.583
260	261	435 2	2	0.583
261	260	435 2	2	0.583
261	262	290 2	2	0.583
262	261	290 2	2	0.583
262	264	360 2	2	0.583
264	262	360 2	2	0.583
264	265	325 2	2	0.583
265	264	325 2	2	0.583
265	266	550 2	2	0.583
266	265	550 2	2	0.583
266	267	375 2	2	0.583
267	266	375 2	2	0.583
267	268	425 2	2	0.583
268	267	425 2	2	0.583
268	269	375 2	2	0.583
269	268	375 2	2	0.583
269	270	375 2	2	0.583
270	269	375 2	2	0.583
270	271	375 2	2	0.583
271	270	375 2	2	0.583
271	272	375 2	2	0.583
272	271	375 2	2	0.583
272	273	120 2	2	0.583
273	272	120 2	2	0.583
266	234	1050 2	2	0.583
234	266	1050 2	2	0.583
215	216	355 2	2	0.583
216	215	355 2	2	0.583
276	273	430 2	2	0.583
234	235	425 2	2	0.583
235	234	425 2	2	0.583
235	236	375 2	2	0.583
236	235	375 2	2	0.583
236	237	375 2	2	0.583
237	236	375 2	2	0.583
237	238	375 2	2	0.583
238	237	375 2	2	0.583
238	239	375 2	2	0.583
239	238	375 2	2	0.583
239	275	245 2	2	0.583
275	239	245 2	2	0.583
275	240	135 2	2	0.583
240	275	135 2	2	0.583
240	241	410 2	2	0.583
241	240	410 2	2	0.583
241	242	405 2	2	0.583
242	241	405 2	2	0.583
242	243	375 2	2	0.583
243	242	375 2	2	0.583
243	244	375 2	2	0.583
244	243	375 2	2	0.583
244	245	750 2	2	0.583
245	244	750 2	2	0.583
245	246	375 2	2	0.583
246	245	375 2	2	0.583
285	286	390 2	2	0.583
286	285	390 2	2	0.583
286	287	275 2	2	0.583
287	286	275 2	2	0.583
287	288	655 2	2	0.583

288	287	655	2	2	0.583
309	305	375	2	2	0.583
305	309	375	2	2	0.583
305	306	130	2	2	0.583
306	305	130	2	2	0.583
306	247	125	2	2	0.583
247	306	125	2	2	0.583
247	307	340	2	2	0.583
307	247	340	2	2	0.583
307	308	315	2	2	0.583
308	307	315	2	2	0.583
308	310	440	2	2	0.583
310	308	440	2	2	0.583
289	290	420	2	2	0.583
290	289	420	2	2	0.583
291	292	365	2	2	0.583
292	291	365	2	2	0.583
292	293	370	2	2	0.583
293	292	370	2	2	0.583
293	294	375	2	2	0.583
294	293	375	2	2	0.583
294	295	375	2	2	0.583
295	294	375	2	2	0.583
295	283	220	2	2	0.583
283	295	220	2	2	0.583
249	298	665	2	2	0.583
298	249	665	2	2	0.583
298	299	340	2	2	0.583
299	300	290	2	2	0.583
299	298	240	2	2	0.583
300	299	290	2	2	0.583
300	301	440	2	2	0.583
301	300	440	2	2	0.583
299	302	580	2	2	0.583
302	299	580	2	2	0.583
301	304	375	2	2	0.583
304	301	375	2	2	0.583
303	304	475	2	2	0.583
304	303	475	2	2	0.583
304	328	190	2	2	0.583
328	304	190	2	2	0.583
339	311	380	2	2	0.583
311	339	380	2	2	0.583
311	312	100	2	2	0.583
312	313	100	2	2	0.583
313	311	200	2	2	0.583
313	351	150	2	2	0.583
351	313	150	2	2	0.583
351	314	120	2	2	0.583
314	351	120	2	2	0.583
314	315	520	2	2	0.583
315	314	520	2	2	0.583
316	266	920	2	2	0.583
266	316	920	2	2	0.583
317	318	420	2	2	0.583
318	317	420	2	2	0.583
318	319	375	2	2	0.583
319	318	375	2	2	0.583
319	320	375	2	2	0.583
320	319	375	2	2	0.583
320	321	375	2	2	0.583
321	320	375	2	2	0.583
321	322	375	2	2	0.583
322	321	375	2	2	0.583
322	323	385	2	2	0.583
323	322	385	2	2	0.583
323	324	410	2	2	0.583
324	323	410	2	2	0.583
324	325	405	2	2	0.583
325	324	405	2	2	0.583
325	326	375	2	2	0.583
326	325	375	2	2	0.583
326	327	375	2	2	0.583
327	326	375	2	2	0.583
284	296	260	2	2	0.583
296	284	260	2	2	0.583
296	297	375	2	2	0.583
297	296	375	2	2	0.583
278	277	365	2	2	0.583
279	278	260	2	2	0.583
278	279	260	2	2	0.583
279	280	375	2	2	0.583
280	279	375	2	2	0.583
280	281	750	2	2	0.583
281	280	750	2	2	0.583
281	282	375	2	2	0.583
282	281	375	2	2	0.583
331	332	220	2	4	0.917
332	331	220	2	4	0.917
332	333	675	2	4	0.917
333	332	675	2	4	0.917
333	334	500	2	4	0.917
334	333	500	2	4	0.917
334	335	410	2	4	0.917
335	334	410	2	4	0.917
335	336	300	2	4	0.917
336	335	300	2	4	0.917
336	337	380	2	4	0.917
337	336	380	2	4	0.917
337	338	475	2	4	0.917
338	337	475	2	4	0.917
338	339	600	2	4	0.917
339	329	305	2	4	0.917
339	338	600	2	4	0.917
329	339	300	2	4	0.917
329	340	305	2	4	0.917
340	329	305	2	4	0.917
340	451	1025	2	4	0.917
451	340	1025	2	4	0.917
451	341	975	2	4	0.917
341	451	975	2	4	0.917
341	342	425	2	4	0.917

342	341	425	2	4	0.917
342	343	375	2	4	0.917
343	342	375	2	4	0.917
343	344	375	2	4	0.917
344	343	375	2	4	0.917
344	345	375	2	4	0.917
345	344	375	2	4	0.917
345	346	375	2	4	0.917
346	345	375	2	4	0.917
346	347	390	2	4	0.917
347	346	390	2	4	0.917
347	348	405	2	4	0.917
348	347	405	2	4	0.917
348	349	405	2	4	0.917
349	348	405	2	4	0.917
349	350	375	2	4	0.917
350	349	375	2	4	0.917
350	382	375	2	4	0.917
382	350	375	2	4	0.917
382	381	380	2	4	0.917
381	382	380	2	4	0.917
358	404	400	2	2	0.583
359	358	400	2	2	0.583
360	359	375	2	2	0.583
361	360	375	2	2	0.583
362	361	375	2	2	0.583
363	362	375	2	2	0.583
364	363	390	2	2	0.583
365	364	405	2	2	0.583
366	365	405	2	2	0.583
367	366	375	2	2	0.583
368	367	375	2	2	0.583
274	369	400	2	2	0.583
369	370	400	2	2	0.583
370	371	400	2	2	0.583
371	372	375	2	2	0.583
372	373	375	2	2	0.583
373	374	375	2	2	0.583
374	375	375	2	2	0.583
375	376	390	2	2	0.583
376	377	405	2	2	0.583
377	378	405	2	2	0.583
378	379	375	2	2	0.583
379	380	375	2	2	0.583
384	385	400	2	2	0.583
385	384	400	2	2	0.583
387	386	400	2	2	0.583
388	387	375	2	2	0.583
389	388	375	2	2	0.583
390	389	375	2	2	0.583
391	390	375	2	2	0.583
392	391	390	2	2	0.583
393	392	405	2	2	0.583
394	393	405	2	2	0.583
395	394	375	2	2	0.583
396	395	375	2	2	0.583

352	353	625	2	2	0.583
353	352	625	2	2	0.583
400	402	600	2	2	0.583
402	400	600	2	2	0.583
356	411	370	2	2	0.583
411	356	370	2	2	0.583
413	414	750	2	4	0.917
414	413	750	2	4	0.917
414	415	200	2	4	0.917
415	414	200	2	4	0.917
415	416	380	2	4	0.917
416	415	380	2	4	0.917
416	417	375	2	4	0.917
417	416	375	2	4	0.917
417	418	175	2	4	0.917
418	417	175	2	4	0.917
418	419	230	2	4	0.917
419	418	230	2	4	0.917
419	420	260	2	4	0.917
420	419	260	2	4	0.917
420	421	130	2	4	0.917
421	420	130	2	4	0.917
421	422	200	2	4	0.917
422	421	200	2	4	0.917
422	423	135	2	4	0.917
423	422	135	2	4	0.917
423	424	185	2	4	0.917
424	423	185	2	4	0.917
424	425	160	2	4	0.917
425	424	160	2	4	0.917
425	426	195	2	4	0.917
426	425	195	2	4	0.917
426	427	180	2	4	0.917
427	426	180	2	4	0.917
427	428	180	2	4	0.917
428	427	180	2	4	0.917
428	429	180	2	4	0.917
429	428	180	2	4	0.917
429	430	160	2	4	0.917
430	429	160	2	4	0.917
430	431	230	2	4	0.917
431	430	230	2	4	0.917
431	432	395	2	4	0.917
432	431	395	2	4	0.917
432	433	395	2	4	0.917
433	432	395	2	4	0.917
433	434	375	2	4	0.917
434	433	375	2	4	0.917
434	435	375	2	4	0.917
435	434	375	2	4	0.917
435	436	720	2	4	0.917
436	435	720	2	4	0.917
444	445	340	2	2	0.583
445	444	340	2	2	0.583
445	446	340	2	2	0.583
446	445	340	2	2	0.583

437	438	500 2 2	0.583	
438	437	500 2 2	0.583	
438	439	340 2 2	0.583	
439	438	340 2 2	0.583	
439	440	340 2 2	0.583	
440	439	340 2 2	0.583	
440	441	350 2 2	0.583	
441	440	350 2 2	0.583	
441	442	340 2 2	0.583	
442	441	340 2 2	0.583	
442	443	340 2 2	0.583	
443	442	340 2 2	0.583	
443	447	850 2 2	0.583	
447	443	850 2 2	0.583	
447	449	755 2 2	0.583	
449	447	755 2 2	0.583	
449	450	375 2 2	0.583	
450	449	375 2 2	0.583	
397	398	200 2 2	0.583	
398	397	200 2 2	0.583	
398	399	240 2 2	0.583	
399	398	240 2 2	0.583	
399	401	500 2 2	0.583	
401	399	500 2 2	0.583	
401	403	265 2 2	0.583	
403	401	265 2 2	0.583	
403	406	475 2 2	0.583	
406	403	475 2 2	0.583	
406	405	150 2 2	0.583	
405	406	150 2 2	0.583	
464	465	430 2 2	0.583	
465	464	430 2 2	0.583	
466	248	910 2 2	0.583	
248	466	910 2 2	0.583	
383	468	300 2 2	0.583	
468	383	300 2 2	0.583	
468	469	300 2 2	0.583	
469	468	300 2 2	0.583	
469	470	690 2 2	0.583	
470	469	690 2 2	0.583	
471	472	720 2 2	0.583	
472	471	720 2 2	0.583	
467	407	370 2 2	0.583	
407	408	220 2 2	0.583	
408	407	220 2 2	0.583	
473	467	480 2 2	0.583	
473	477	500 2 2	0.583	
477	473	500 2 2	0.583	
477	507	240 2 2	0.583	
507	477	240 2 2	0.583	
454	455	380 2 2	0.583	
455	454	380 2 2	0.583	
455	456	450 2 2	0.583	
456	455	450 2 2	0.583	
456	457	530 2 2	0.583	
457	456	530 2 2	0.583	
457	458	490 2 2	0.583	
458	457	490 2 2	0.583	
458	459	320 2 2	0.583	
459	458	320 2 2	0.583	
459	460	310 2 2	0.583	
460	461	310 2 2	0.583	
461	460	310 2 2	0.583	
461	409	1100 2 2	0.583	
409	461	1100 2 2	0.583	
409	462	430 2 2	0.583	
462	409	430 2 2	0.583	
463	454	190 2 2	0.583	
454	463	190 2 2	0.583	
460	459	310 2 2	0.583	
479	480	690 2 2	0.583	
480	479	690 2 2	0.583	
484	485	480 2 2	0.583	
485	484	480 2 2	0.583	
486	487	530 2 2	0.583	
487	486	530 2 2	0.583	
487	488	530 2 2	0.583	
488	487	530 2 2	0.583	
482	483	530 2 2	0.583	
483	482	530 2 2	0.583	
481	482	530 2 2	0.583	
482	481	530 2 2	0.583	
489	490	330 2 2	0.583	
490	489	330 2 2	0.583	
490	491	330 2 2	0.583	
491	490	330 2 2	0.583	
491	492	330 2 2	0.583	
492	491	330 2 2	0.583	
500	501	280 2 2	0.583	
501	500	280 2 2	0.583	
501	502	375 2 2	0.583	
502	501	375 2 2	0.583	
502	505	400 2 2	0.583	
505	502	400 2 2	0.583	
505	507	900 2 2	0.583	
507	505	900 2 2	0.583	
507	509	480 2 2	0.583	
509	507	480 2 2	0.583	
509	510	200 2 3	0.750	
510	509	200 2 3	0.750	
510	511	190 2 3	0.750	
511	510	190 2 3	0.750	
511	512	560 2 3	0.750	
512	511	560 2 3	0.750	
512	506	500 2 3	0.750	
506	512	500 2 3	0.750	
506	513	340 2 3	0.750	
513	506	340 2 3	0.750	
513	514	190 2 3	0.750	
514	513	190 2 3	0.750	
514	515	740 2 3	0.750	
515	514	740 2 3	0.750	

515	516	240	2	3	0.750
516	515	240	2	3	0.750
516	498	100	2	3	0.750
498	516	100	2	3	0.750
498	517	120	2	3	0.750
517	498	120	2	3	0.750
517	499	155	2	3	0.750
499	517	155	2	3	0.750
499	518	75	2	3	0.750
518	499	75	2	3	0.750
518	521	275	2	3	0.750
521	518	275	2	3	0.750
521	522	400	2	2	0.583
522	521	400	2	2	0.583
525	522	400	2	2	0.583
522	525	400	2	2	0.583
525	526	765	2	2	0.583
526	525	765	2	2	0.583
526	527	1140	2	2	0.583
527	526	1140	2	2	0.583
493	531	265	2	2	0.583
531	522	290	2	2	0.583
494	495	350	2	2	0.583
495	494	350	2	2	0.583
495	496	640	2	2	0.583
496	495	640	2	2	0.583
530	528	620	2	2	0.583
528	495	730	2	2	0.583
528	529	450	2	2	0.583
529	528	450	2	2	0.583
525	493	240	2	2	0.583
493	525	240	2	2	0.583
493	409	1050	2	2	0.583
543	544	380	2	2	0.583
544	543	380	2	2	0.583
519	520	245	2	2	0.583
520	519	245	2	2	0.583
545	546	495	2	2	0.583
546	545	465	2	2	0.583
547	548	745	2	2	0.583
548	547	745	2	2	0.583
533	534	400	2	2	0.583
534	533	400	2	2	0.583
534	535	625	2	2	0.583
535	534	625	2	2	0.583
535	536	500	2	2	0.583
536	535	500	2	2	0.583
536	537	330	2	2	0.583
537	536	330	2	2	0.583
537	538	740	2	2	0.583
538	537	740	2	2	0.583
538	539	245	2	2	0.583
539	538	245	2	2	0.583
539	540	220	2	2	0.583
540	539	220	2	2	0.583
540	541	245	2	2	0.583
541	540	245	2	2	0.583
541	542	245	2	2	0.583
542	541	245	2	2	0.583
532	549	500	2	2	0.583
549	532	500	2	2	0.583
549	550	330	2	2	0.583
550	549	330	2	2	0.583
550	551	190	2	2	0.583
551	550	190	2	2	0.583
551	552	180	2	2	0.583
552	551	180	2	2	0.583
552	553	370	2	2	0.583
553	552	370	2	2	0.583
553	554	245	2	2	0.583
554	553	245	2	2	0.583
554	555	220	2	2	0.583
555	554	220	2	2	0.583
555	556	245	2	2	0.583
556	555	245	2	2	0.583
556	557	245	2	2	0.583
557	556	245	2	2	0.583
558	559	380	2	3	0.750
559	558	380	2	3	0.750
559	560	350	2	3	0.750
560	559	350	2	3	0.750
560	561	360	2	3	0.750
561	560	360	2	3	0.750
561	562	375	2	3	0.750
562	561	375	2	3	0.750
562	563	750	2	3	0.750
563	562	750	2	3	0.750
563	564	650	2	3	0.750
564	563	650	2	3	0.750
564	565	750	2	3	0.750
565	564	750	2	3	0.750
565	566	225	2	3	0.750
566	565	225	2	3	0.750
566	567	260	2	3	0.750
567	566	260	2	3	0.750
567	568	950	2	3	0.750
568	567	950	2	3	0.750
1	2	450	2	2	0.583
2	1	450	2	2	0.583
2	6	335	2	2	0.583
6	2	335	2	2	0.583
5	8	940	2	2	0.583
8	5	940	2	2	0.583
8	10	280	2	2	0.583
10	8	280	2	2	0.583
24	36	370	2	2	0.583
36	24	370	2	2	0.583
36	52	380	2	2	0.583
52	36	380	2	2	0.583
52	65	375	2	2	0.583
65	52	375	2	2	0.583
65	91	350	2	2	0.583

91	65	350	2	2	0.583
91	125	375	2	2	0.583
125	91	375	2	2	0.583
125	157	360	2	2	0.583
157	125	360	2	2	0.583
157	171	375	2	2	0.583
171	157	375	2	2	0.583
200	231	375	2	2	0.583
231	200	375	2	2	0.583
246	282	475	2	2	0.583
282	246	475	2	2	0.583
330	282	810	2	2	0.583
282	330	810	2	2	0.583
9	23	260	2	3	0.750
23	9	260	2	3	0.750
23	35	370	2	3	0.750
35	51	380	2	3	0.750
51	35	380	2	3	0.750
51	64	375	2	3	0.750
64	51	375	2	3	0.750
64	90	350	2	3	0.750
90	64	350	2	3	0.750
90	124	375	2	3	0.750
124	90	375	2	3	0.750
124	156	360	2	3	0.750
156	124	360	2	3	0.750
156	170	375	2	3	0.750
170	156	375	2	3	0.750
170	199	375	2	3	0.750
199	170	375	2	3	0.750
199	230	375	2	3	0.750
230	199	375	2	3	0.750
230	245	340	2	3	0.750
245	230	340	2	3	0.750
245	281	480	2	3	0.750
281	245	480	2	3	0.750
281	381	1320	2	3	0.750
381	281	1320	2	3	0.750
381	436	1650	2	3	0.750
436	381	1650	2	3	0.750
436	497	730	2	3	0.750
497	436	730	2	3	0.750
497	568	2650	2	3	0.750
568	497	2650	2	3	0.750
497	496	1650	2	2	0.583
496	497	1650	2	2	0.583
496	529	750	2	2	0.583
529	496	750	2	2	0.583
529	530	230	2	2	0.583
530	529	230	2	2	0.583
530	567	385	2	2	0.583
567	530	385	2	2	0.583
22	34	370	2	2	0.583
34	22	370	2	2	0.583
34	50	380	2	2	0.583
50	34	380	2	2	0.583

50	63	375	2	2	0.583
63	50	375	2	2	0.583
63	89	350	2	2	0.583
89	63	350	2	2	0.583
89	123	375	2	2	0.583
123	89	375	2	2	0.583
123	155	360	2	2	0.583
155	123	360	2	2	0.583
155	169	375	2	2	0.583
169	155	375	2	2	0.583
169	198	375	2	2	0.583
198	169	375	2	2	0.583
198	229	375	2	2	0.583
229	198	375	2	2	0.583
13	21	355	2	2	0.583
21	13	355	2	2	0.583
21	33	370	2	2	0.583
33	49	385	2	2	0.583
49	62	375	2	2	0.583
62	88	350	2	2	0.583
88	122	360	2	2	0.583
122	154	350	2	2	0.583
154	168	375	2	2	0.583
168	197	375	2	2	0.583
197	228	375	2	2	0.583
228	244	320	2	2	0.583
244	280	470	2	2	0.583
280	297	450	2	2	0.583
297	327	375	2	2	0.583
327	382	375	2	2	0.583
382	368	375	2	2	0.583
368	380	355	2	2	0.583
380	396	355	2	2	0.583
396	435	405	2	2	0.583
435	450	1500	2	2	0.583
450	434	1625	2	2	0.583
450	494	960	2	2	0.583
494	450	960	2	2	0.583
494	527	600	2	2	0.583
527	494	600	2	2	0.583
527	563	800	2	2	0.583
563	527	800	2	2	0.583
434	395	405	2	2	0.583
395	379	355	2	2	0.583
379	367	355	2	2	0.583
367	350	375	2	2	0.583
350	326	375	2	2	0.583
326	296	375	2	2	0.583
296	279	450	2	2	0.583
279	243	470	2	2	0.583
243	227	320	2	2	0.583
227	196	375	2	2	0.583
196	167	375	2	2	0.583
167	153	375	2	2	0.583
153	121	350	2	2	0.583
121	87	360	2	2	0.583

87	61	350	2	2	0.583
61	48	375	2	2	0.583
48	32	385	2	2	0.583
32	20	370	2	2	0.583
20	12	355	2	2	0.583
12	20	355	2	2	0.583
433	394	405	2	2	0.583
394	433	405	2	2	0.583
394	378	355	2	2	0.583
378	394	355	2	2	0.583
366	349	375	2	2	0.583
349	366	375	2	2	0.583
349	325	375	2	2	0.583
325	349	375	2	2	0.583
284	278	450	2	2	0.583
278	284	450	2	2	0.583
242	226	320	2	2	0.583
226	242	320	2	2	0.583
226	195	375	2	2	0.583
195	226	375	2	2	0.583
195	166	375	2	2	0.583
166	195	375	2	2	0.583
166	152	375	2	2	0.583
152	166	375	2	2	0.583
152	120	350	2	2	0.583
120	152	350	2	2	0.583
120	86	360	2	2	0.583
86	120	360	2	2	0.583
86	60	350	2	2	0.583
60	86	350	2	2	0.583
60	47	375	2	2	0.583
47	60	375	2	2	0.583
47	31	385	2	2	0.583
31	47	385	2	2	0.583
31	19	370	2	2	0.583
19	31	370	2	2	0.583
19	11	355	2	2	0.583
11	19	355	2	2	0.583
18	30	370	2	3	0.750
30	18	370	2	3	0.750
30	46	385	2	3	0.750
46	30	385	2	3	0.750
46	59	375	2	3	0.750
59	46	375	2	3	0.750
59	85	350	2	3	0.750
85	59	350	2	3	0.750
85	119	360	2	3	0.750
119	85	360	2	3	0.750
119	151	350	2	3	0.750
151	119	350	2	3	0.750
151	165	375	2	3	0.750
165	151	375	2	3	0.750
165	194	375	2	3	0.750
194	165	375	2	3	0.750
194	225	375	2	3	0.750
225	194	375	2	3	0.750

225	241	320	2	3	0.750
241	225	320	2	3	0.750
241	277	750	2	3	0.750
277	324	700	2	3	0.750
324	276	700	2	3	0.750
276	241	750	2	3	0.750
324	348	375	2	3	0.750
348	324	375	2	3	0.750
348	365	375	2	3	0.750
365	348	375	2	3	0.750
365	377	355	2	3	0.750
377	365	355	2	3	0.750
377	393	375	2	3	0.750
393	377	375	2	3	0.750
393	432	375	2	3	0.750
432	393	375	2	3	0.750
432	503	380	2	2	0.583
503	432	380	2	2	0.583
503	447	750	2	2	0.583
447	503	750	2	2	0.583
447	462	490	2	2	0.583
462	447	490	2	2	0.583
462	526	1000	2	2	0.583
526	462	1000	2	2	0.583
526	562	800	2	2	0.583
562	526	800	2	2	0.583
503	449	1430	2	2	0.583
449	503	1430	2	2	0.583
17	29	370	2	2	0.583
29	17	370	2	2	0.583
29	45	385	2	2	0.583
45	29	385	2	2	0.583
45	58	375	2	2	0.583
58	45	375	2	2	0.583
58	84	350	2	2	0.583
84	58	350	2	2	0.583
84	118	360	2	2	0.583
118	84	360	2	2	0.583
118	150	350	2	2	0.583
150	118	350	2	2	0.583
150	164	375	2	2	0.583
164	150	375	2	2	0.583
164	193	375	2	2	0.583
193	164	375	2	2	0.583
193	224	375	2	2	0.583
224	193	375	2	2	0.583
224	240	320	2	2	0.583
240	224	320	2	2	0.583
275	273	465	2	2	0.583
273	275	465	2	2	0.583
273	283	450	2	2	0.583
283	273	450	2	2	0.583
347	323	375	2	2	0.583
323	347	375	2	2	0.583
347	364	375	2	2	0.583
364	347	375	2	2	0.583

364	376	355	2	2	0.583
376	364	355	2	2	0.583
376	392	375	2	2	0.583
392	376	375	2	2	0.583
392	431	375	2	2	0.583
431	392	375	2	2	0.583
16	28	370	2	3	0.750
28	44	385	2	3	0.750
44	57	375	2	3	0.750
57	83	350	2	3	0.750
83	117	360	2	3	0.750
117	149	350	2	3	0.750
149	163	375	2	3	0.750
163	192	375	2	3	0.750
192	223	375	2	3	0.750
223	239	320	2	3	0.750
239	272	465	2	3	0.750
272	295	450	2	3	0.750
295	322	375	2	3	0.750
322	346	375	2	3	0.750
346	363	375	2	3	0.750
363	375	355	2	3	0.750
375	391	355	2	3	0.750
391	429	405	2	3	0.750
427	390	375	2	3	0.750
390	374	375	2	3	0.750
374	362	355	2	3	0.750
362	345	375	2	3	0.750
345	321	375	2	3	0.750
321	294	375	2	3	0.750
294	271	450	2	3	0.750
271	238	465	2	3	0.750
238	222	320	2	3	0.750
222	191	375	2	3	0.750
191	162	375	2	3	0.750
162	148	375	2	3	0.750
148	116	350	2	3	0.750
116	82	360	2	3	0.750
82	56	350	2	3	0.750
56	43	375	2	3	0.750
43	27	385	2	3	0.750
27	15	370	2	3	0.750
147	115	350	2	2	0.583
293	270	450	2	2	0.583
26	14	370	2	2	0.583
26	42	385	2	2	0.583
42	26	385	2	2	0.583
42	55	375	2	2	0.583
55	42	375	2	2	0.583
55	81	350	2	2	0.583
81	55	350	2	2	0.583
81	115	360	2	2	0.583
115	81	360	2	2	0.583
115	147	350	2	2	0.583
161	147	375	2	2	0.583
161	190	375	2	2	0.583

190	161	375	2	2	0.583
190	221	375	2	2	0.583
221	190	375	2	2	0.583
221	237	320	2	2	0.583
237	221	320	2	2	0.583
237	270	465	2	2	0.583
270	237	465	2	2	0.583
270	293	450	2	2	0.583
293	320	375	2	2	0.583
320	293	375	2	2	0.583
320	344	375	2	2	0.583
344	320	375	2	2	0.583
344	361	375	2	2	0.583
361	344	375	2	2	0.583
361	373	355	2	2	0.583
373	361	355	2	2	0.583
373	389	375	2	2	0.583
389	373	375	2	2	0.583
389	425	375	2	2	0.583
425	389	375	2	2	0.583
25	41	385	2	2	0.583
41	54	375	2	2	0.583
54	41	375	2	2	0.583
54	80	350	2	2	0.583
80	54	350	2	2	0.583
80	114	360	2	2	0.583
114	80	360	2	2	0.583
114	146	350	2	2	0.583
146	114	350	2	2	0.583
146	160	375	2	2	0.583
160	146	375	2	2	0.583
160	189	375	2	2	0.583
189	160	375	2	2	0.583
189	220	375	2	2	0.583
220	189	375	2	2	0.583
220	236	320	2	2	0.583
236	220	320	2	2	0.583
236	269	465	2	2	0.583
269	236	465	2	2	0.583
269	292	450	2	2	0.583
292	269	450	2	2	0.583
292	319	375	2	2	0.583
319	292	375	2	2	0.583
319	343	375	2	2	0.583
343	319	375	2	2	0.583
343	360	375	2	2	0.583
360	343	375	2	2	0.583
360	372	355	2	2	0.583
372	360	355	2	2	0.583
372	388	375	2	2	0.583
388	372	375	2	2	0.583
388	423	375	2	2	0.583
423	388	375	2	2	0.583
53	79	350	2	2	0.583
79	53	350	2	2	0.583
79	113	360	2	2	0.583

113	79	360	2	2	0.583
113	145	350	2	2	0.583
145	113	350	2	2	0.583
145	159	375	2	2	0.583
159	145	375	2	2	0.583
159	188	375	2	2	0.583
188	159	375	2	2	0.583
188	219	375	2	2	0.583
219	188	375	2	2	0.583
219	235	320	2	2	0.583
235	219	320	2	2	0.583
235	268	465	2	2	0.583
268	235	465	2	2	0.583
268	291	450	2	2	0.583
291	268	450	2	2	0.583
291	290	200	2	2	0.583
290	291	200	2	2	0.583
290	318	175	2	2	0.583
318	290	175	2	2	0.583
318	342	375	2	2	0.583
342	318	375	2	2	0.583
342	359	375	2	2	0.583
359	342	375	2	2	0.583
359	371	360	2	2	0.583
371	359	360	2	2	0.583
371	387	375	2	2	0.583
387	371	375	2	2	0.583
387	421	375	2	2	0.583
421	387	375	2	2	0.583
78	112	350	2	2	0.583
112	78	350	2	2	0.583
112	144	360	2	2	0.583
144	112	360	2	2	0.583
144	158	350	2	2	0.583
158	144	350	2	2	0.583
158	187	375	2	2	0.583
187	158	375	2	2	0.583
187	218	375	2	2	0.583
218	187	375	2	2	0.583
218	234	375	2	2	0.583
234	218	375	2	2	0.583
234	267	320	2	2	0.583
267	234	320	2	2	0.583
267	289	650	2	2	0.583
289	267	650	2	2	0.583
289	317	175	2	2	0.583
317	289	175	2	2	0.583
317	341	375	2	2	0.583
341	317	375	2	2	0.583
341	358	375	2	2	0.583
358	341	375	2	2	0.583
358	370	375	2	2	0.583
370	358	375	2	2	0.583
370	386	375	2	2	0.583
386	370	375	2	2	0.583
386	419	375	2	2	0.583
419	386	375	2	2	0.583
76	108	500	2	4	0.917
108	76	500	2	4	0.917
108	185	975	2	4	0.917
185	108	975	2	4	0.917
185	217	415	2	4	0.917
217	185	415	2	4	0.917
217	216	410	2	4	0.917
216	217	410	2	4	0.917
216	265	380	2	4	0.917
265	216	380	2	4	0.917
265	315	200	2	4	0.917
315	265	200	2	4	0.917
315	316	465	2	4	0.917
316	315	465	2	4	0.917
316	452	370	2	4	0.917
452	316	370	2	4	0.917
452	250	550	2	4	0.917
250	452	550	2	4	0.917
250	413	1250	2	4	0.917
413	250	1250	2	4	0.917
413	478	1280	2	4	0.917
478	413	1280	2	4	0.917
478	509	645	2	4	0.917
509	478	645	2	4	0.917
509	533	490	2	4	0.917
533	509	490	2	4	0.917
100	70	485	2	2	0.583
71	101	485	2	2	0.583
101	178	1120	2	2	0.583
178	101	1120	2	2	0.583
209	258	610	2	2	0.583
258	209	610	2	2	0.583
257	247	375	2	2	0.583
247	257	375	2	2	0.583
306	328	355	2	2	0.583
328	306	355	2	2	0.583
328	337	360	2	2	0.583
337	328	360	2	2	0.583
337	355	565	2	2	0.583
355	337	565	2	2	0.583
99	136	310	2	2	0.583
136	99	310	2	2	0.583
136	135	320	2	2	0.583
135	136	320	2	2	0.583
135	134	250	2	2	0.583
134	135	250	2	2	0.583
134	177	265	2	2	0.583
177	134	265	2	2	0.583
98	137	310	2	2	0.583
137	98	310	2	2	0.583
137	138	320	2	2	0.583
138	137	320	2	2	0.583
138	133	250	2	2	0.583
133	138	250	2	2	0.583
133	176	265	2	2	0.583

176	133	265	2	2	0.583
105	142	465	2	2	0.583
142	105	465	2	2	0.583
142	141	200	2	2	0.583
141	142	200	2	2	0.583
141	183	260	2	2	0.583
183	141	260	2	2	0.583
183	182	100	2	2	0.583
182	183	100	2	2	0.583
182	213	510	2	2	0.583
213	182	510	2	2	0.583
213	261	610	2	2	0.583
261	213	610	2	2	0.583
261	310	430	2	2	0.583
310	261	430	2	2	0.583
310	339	280	2	2	0.583
339	310	480	2	2	0.583
102	179	1130	2	2	0.583
179	102	1130	2	2	0.583
181	212	610	2	2	0.583
212	181	610	2	2	0.583
211	260	610	2	2	0.583
260	211	610	2	2	0.583
260	308	400	2	2	0.583
308	260	400	2	2	0.583
210	259	610	2	2	0.583
259	210	610	2	2	0.583
259	307	405	2	2	0.583
307	259	405	2	2	0.583
307	338	550	2	2	0.583
338	307	550	2	2	0.583
338	357	450	2	2	0.583
357	338	450	2	2	0.583
404	369	360	2	2	0.583
369	385	375	2	2	0.583
385	369	375	2	2	0.583
385	417	375	2	2	0.583
417	385	375	2	2	0.583
384	274	375	2	2	0.583
384	416	375	2	2	0.583
416	384	375	2	2	0.583
109	186	975	2	2	0.583
186	109	975	2	2	0.583
110	77	455	2	2	0.583
77	110	455	2	2	0.583
77	69	440	2	2	0.583
69	77	440	2	2	0.583
69	38	800	2	2	0.583
38	69	800	2	2	0.583
38	75	430	2	2	0.583
75	38	430	2	2	0.583
75	106	500	2	2	0.583
106	75	500	2	2	0.583
107	143	485	2	2	0.583
143	107	485	2	2	0.583
143	184	485	2	2	0.583

184	143	485	2	2	0.583
214	215	410	2	2	0.583
215	214	410	2	2	0.583
215	264	380	2	2	0.583
264	215	380	2	2	0.583
264	314	585	2	2	0.583
314	264	585	2	2	0.583
73	104	485	2	2	0.583
104	73	485	2	2	0.583
405	467	250	2	2	0.583
467	405	250	2	2	0.583
467	263	870	2	2	0.583
263	467	870	2	2	0.583
263	472	280	2	2	0.583
472	263	280	2	2	0.583
472	505	100	2	2	0.583
505	472	100	2	2	0.583
332	352	360	2	2	0.583
352	332	360	2	2	0.583
352	400	250	2	2	0.583
400	352	250	2	2	0.583
400	399	320	2	2	0.583
399	400	320	2	2	0.583
399	465	365	2	2	0.583
465	399	365	2	2	0.583
465	466	125	2	2	0.583
466	465	125	2	2	0.583
466	468	365	2	2	0.583
468	466	365	2	2	0.583
468	500	940	2	2	0.583
500	468	940	2	2	0.583
249	332	480	2	2	0.583
331	249	510	2	2	0.583
331	398	980	2	2	0.583
398	331	980	2	2	0.583
397	464	330	2	2	0.583
464	397	330	2	2	0.583
464	383	460	2	2	0.583
383	464	460	2	2	0.583
469	471	415	2	2	0.583
471	469	415	2	2	0.583
471	501	485	2	2	0.583
501	471	485	2	2	0.583
286	298	340	2	2	0.583
298	286	340	2	2	0.583
298	333	350	2	2	0.583
333	298	350	2	2	0.583
333	353	550	2	2	0.583
353	333	550	2	2	0.583
353	402	320	2	2	0.583
402	353	320	2	2	0.583
401	402	360	2	2	0.583
402	401	360	2	2	0.583
351	410	800	2	2	0.583
410	351	800	2	2	0.583
340	412	450	2	2	0.583

412	340	450	2	2	0.583
410	412	370	2	2	0.583
412	410	370	2	2	0.583
412	411	620	2	2	0.583
411	412	620	2	2	0.583
411	408	860	2	2	0.583
408	411	860	2	2	0.583
408	477	1000	2	2	0.583
477	408	1000	2	2	0.583
407	473	660	2	2	0.583
473	407	660	2	2	0.583
329	357	920	2	2	0.583
357	329	920	2	2	0.583
357	356	265	2	2	0.583
356	357	365	2	2	0.583
356	355	260	2	2	0.583
355	356	260	2	2	0.583
355	405	750	2	2	0.583
405	355	750	2	2	0.583
303	335	100	2	2	0.583
335	303	100	2	2	0.583
335	354	575	2	2	0.583
354	335	575	2	2	0.583
336	354	660	2	2	0.583
354	336	660	2	2	0.583
354	406	450	2	2	0.583
406	354	450	2	2	0.583
132	174	280	2	2	0.583
174	132	280	2	2	0.583
132	139	500	2	2	0.583
139	132	500	2	2	0.583
139	205	280	2	2	0.583
205	139	280	2	2	0.583
204	252	325	2	2	0.583
252	204	325	2	2	0.583
252	233	140	2	2	0.583
233	252	140	2	2	0.583
233	287	200	2	2	0.583
287	233	200	2	2	0.583
203	251	325	2	2	0.583
251	203	325	2	2	0.583
251	285	340	2	2	0.583
285	251	340	2	2	0.583
300	334	225	2	2	0.583
334	300	225	2	2	0.583
334	403	1020	2	2	0.583
403	334	1020	2	2	0.583
403	248	500	2	2	0.583
248	403	500	2	2	0.583
248	470	300	2	2	0.583
470	248	300	2	2	0.583
470	263	300	2	2	0.583
263	470	300	2	2	0.583
208	256	610	2	2	0.583
256	208	610	2	2	0.583
255	305	375	2	2	0.583
305	255	375	2	2	0.583
207	254	610	2	2	0.583
254	207	610	2	2	0.583
97	96	150	2	2	0.583
96	68	275	2	2	0.583
67	95	280	2	2	0.583
95	94	150	2	2	0.583
94	174	1110	2	2	0.583
174	94	1110	2	2	0.583
174	175	90	2	2	0.583
175	174	90	2	2	0.583
175	205	410	2	2	0.583
205	175	410	2	2	0.583
205	206	75	2	2	0.583
206	205	75	2	2	0.583
206	232	140	2	2	0.583
232	206	140	2	2	0.583
232	253	475	2	2	0.583
253	232	475	2	2	0.583
253	288	190	2	2	0.583
288	253	190	2	2	0.583
288	309	145	2	2	0.583
309	288	145	2	2	0.583
309	302	130	2	2	0.583
302	309	130	2	2	0.583
302	301	195	2	2	0.583
301	302	195	2	2	0.583
301	303	200	2	2	0.583
303	301	200	2	2	0.583
522	544	440	2	2	0.583
544	522	440	2	2	0.583
544	559	390	2	2	0.583
559	544	390	2	2	0.583
446	430	485	2	2	0.583
430	446	485	2	2	0.583
443	446	440	2	2	0.583
428	445	380	2	2	0.583
445	428	380	2	2	0.583
445	442	440	2	2	0.583
442	445	440	2	2	0.583
426	444	310	2	3	0.750
444	426	310	2	3	0.750
446	443	440	2	2	0.583
444	441	440	2	3	0.750
441	444	440	2	3	0.750
441	461	450	2	3	0.750
461	441	450	2	3	0.750
461	492	480	2	3	0.750
492	461	480	2	3	0.750
492	521	480	2	3	0.750
521	492	480	2	3	0.750
521	543	420	2	3	0.750
543	521	420	2	3	0.750
543	542	120	2	3	0.750
542	543	120	2	3	0.750
542	558	285	2	3	0.750

558	542	285	2	3	0.750
558	557	235	2	3	0.750
557	558	235	2	3	0.750
424	440	680	2	2	0.583
440	460	450	2	2	0.583
460	491	480	2	2	0.583
491	499	480	2	2	0.583
518	541	525	2	2	0.583
541	556	525	2	2	0.583
498	490	480	2	2	0.583
490	459	480	2	2	0.583
459	439	450	2	2	0.583
439	422	575	2	2	0.583
555	540	525	2	2	0.583
540	517	525	2	2	0.583
420	438	500	2	2	0.583
438	458	450	2	2	0.583
458	483	270	2	2	0.583
483	489	210	2	2	0.583
489	488	245	2	2	0.583
488	515	235	2	2	0.583
515	519	265	2	2	0.583
519	538	260	2	2	0.583
538	548	245	2	2	0.583
548	553	280	2	2	0.583
516	520	265	2	2	0.583
520	516	265	2	2	0.583
520	539	260	2	2	0.583
539	520	260	2	2	0.583
539	554	525	2	2	0.583
554	539	525	2	2	0.583
418	437	380	2	2	0.583
437	418	380	2	2	0.583
437	457	435	2	2	0.583
457	437	435	2	2	0.583
457	482	270	2	2	0.583
482	457	270	2	2	0.583
482	487	420	2	2	0.583
487	482	420	2	2	0.583
514	537	500	2	2	0.583
537	514	495	2	2	0.583
537	547	245	2	2	0.583
547	537	245	2	2	0.583
547	550	280	2	2	0.583
550	547	280	2	2	0.583
416	456	735	2	2	0.583
456	416	735	2	2	0.583
456	481	270	2	2	0.583
481	456	270	2	2	0.583
481	485	320	2	2	0.583
485	481	320	2	2	0.583
485	486	100	2	2	0.583
486	485	100	2	2	0.583
486	513	235	2	2	0.583
513	486	235	2	2	0.583
513	536	495	2	2	0.583

536	513	495	2	2	0.583
536	546	180	2	2	0.583
546	536	180	2	2	0.583
546	549	330	2	2	0.583
549	546	330	2	2	0.583
415	455	625	2	2	0.583
455	415	625	2	2	0.583
455	480	315	2	2	0.583
480	455	315	2	2	0.583
480	484	305	2	2	0.583
484	480	305	2	2	0.583
484	506	330	2	2	0.583
506	484	330	2	2	0.583
506	535	500	2	2	0.583
535	506	500	2	2	0.583
535	545	190	2	2	0.583
545	535	190	2	2	0.583
545	532	335	2	2	0.583
532	545	335	2	2	0.583
414	454	750	2	2	0.583
454	414	750	2	2	0.583
463	479	350	2	2	0.583
479	463	350	2	2	0.583
479	512	735	2	2	0.583
512	479	735	2	2	0.583
478	510	980	2	2	0.583
510	478	980	2	2	0.583
511	534	480	2	2	0.583
534	511	480	2	2	0.583
452	451	380	2	2	0.583
451	250	405	2	2	0.583
262	312	550	2	2	0.583
312	262	550	2	2	0.583
603	6	500	1	2	0.583
604	6	500	1	2	0.583
605	10	500	1	2	0.583
606	13	300	1	2	0.583
610	36	500	1	2	0.583
611	52	500	1	2	0.583
612	65	500	1	2	0.583
619	125	500	1	2	0.583
621	171	500	1	2	0.583
623	231	500	1	3	0.750
630	246	500	1	2	0.583
631	282	500	1	2	0.583
632	330	300	1	2	0.583
633	381	500	1	4	0.917
634	436	500	1	4	0.917
655	568	500	1	3	0.750
654	568	500	1	3	0.750
653	566	500	1	2	0.583
652	565	500	1	2	0.583
651	564	500	1	2	0.583
650	563	500	1	2	0.583
649	562	500	1	2	0.583
647	560	500	1	2	0.583

646	559	500	1	2	0.583
645	557	500	1	3	0.750
644	555	500	1	2	0.583
642	552	500	1	2	0.583
641	551	500	1	2	0.583
640	533	500	1	4	0.917
639	502	500	1	2	0.583
638	501	500	1	2	0.583
637	500	500	1	2	0.583
636	500	500	1	2	0.583
635	397	300	1	2	0.583
629	331	500	1	4	0.917
628	249	500	1	2	0.583
627	285	500	1	2	0.583
626	251	500	1	2	0.583
625	201	500	1	2	0.583
624	172	500	1	2	0.583
618	127	300	1	2	0.583
616	66	500	1	3	0.750
615	78	500	1	2	0.583
614	76	500	1	4	0.917
608	14	500	1	2	0.583
607	16	500	1	2	0.583
6	603	500	2	2	0.583
6	604	500	0	2	0.583
10	605	500	2	2	0.583
13	606	300	0	2	0.583
52	611	500	2	2	0.583
65	612	500	2	2	0.583
91	613	500	2	3	0.750
92	617	500	2	2	0.583
157	620	500	2	2	0.583
200	622	500	2	3	0.750
246	630	500	2	2	0.583
282	631	500	2	2	0.583
330	632	300	2	2	0.583
381	633	500	2	4	0.917
436	634	500	2	4	0.917
568	655	500	0	3	0.750
568	654	500	0	3	0.750
566	653	500	0	2	0.583
565	652	500	0	2	0.583
564	651	500	0	2	0.583
563	650	500	0	2	0.583
562	649	500	0	2	0.583
560	647	500	0	2	0.583
559	646	500	0	2	0.583
557	645	500	0	3	0.750
555	644	500	0	2	0.583
552	642	500	0	2	0.583
551	641	500	0	2	0.583
533	640	500	0	4	0.917
502	639	500	0	2	0.583
501	638	500	0	2	0.583
500	637	500	0	2	0.583
500	636	500	2	2	0.583

397	635	300	0	2	0.583
331	629	500	0	4	0.917
249	628	500	0	2	0.583
285	627	500	0	2	0.583
251	626	500	0	2	0.583
172	624	500	2	2	0.583
127	618	300	0	2	0.583
78	615	500	0	2	0.583
76	614	500	2	4	0.917
16	607	500	2	2	0.583
602	603	335	0	5	0.917
603	602	335	0	5	0.917
603	605	1220	0	5	0.917
605	603	1220	0	5	0.917
605	609	260	0	5	0.917
609	605	260	0	5	0.917
609	610	370	0	5	0.917
610	609	370	0	5	0.917
610	611	380	0	5	0.917
611	610	380	0	5	0.917
611	612	375	0	5	0.917
612	611	375	0	5	0.917
612	613	350	0	5	0.917
613	612	350	0	5	0.917
613	619	375	0	5	0.917
619	613	375	0	5	0.917
619	620	360	0	5	0.917
620	619	360	0	5	0.917
620	621	375	0	5	0.917
621	620	375	0	5	0.917
621	622	375	0	5	0.917
622	621	375	0	5	0.917
622	623	375	0	5	0.917
623	622	375	0	5	0.917
623	630	340	0	5	0.917
630	623	340	0	5	0.917
630	631	480	0	5	0.917
631	630	480	0	5	0.917
631	632	825	0	5	0.917
632	631	825	0	5	0.917
632	633	375	0	5	0.917
633	632	375	0	5	0.917
633	634	1500	0	5	0.917
634	633	1500	0	5	0.917
634	654	1500	0	5	0.917
654	634	1500	0	5	0.917
636	635	1800	0	5	0.917
635	636	1800	0	5	0.917
635	629	980	0	5	0.917
629	635	980	0	5	0.917
629	625	1800	0	5	0.917
625	629	1800	0	5	0.917
625	624	450	0	5	0.917
624	625	450	0	5	0.917
624	617	600	0	5	0.917
617	624	600	0	5	0.917

```
617 616 400 0 5 0.917
616 617 400 0 5 0.917
571 572 600 1 5 0.917
572 571 600 0 5 0.917
572 573 600 1 5 0.917
573 572 600 0 5 0.917
573 574 600 1 5 0.917
574 573 600 0 5 0.917
574 575 600 1 5 0.917
575 574 600 0 5 0.917
575 602 600 1 5 0.917
602 575 600 0 5 0.917
576 577 600 1 5 0.917
577 576 600 0 5 0.917
577 578 600 1 5 0.917
578 577 600 0 5 0.917
578 579 600 1 5 0.917
579 578 600 0 5 0.917
579 580 600 1 5 0.917
580 579 600 0 5 0.917
580 616 600 1 5 0.917
616 580 600 0 5 0.917
581 582 600 1 5 0.917
582 581 600 0 5 0.917
582 583 600 1 5 0.917
583 582 600 0 5 0.917
583 584 600 1 5 0.917
584 583 600 0 5 0.917
584 585 600 1 5 0.917
585 584 600 0 5 0.917
585 636 600 1 5 0.917
636 585 600 0 5 0.917
586 587 600 1 5 0.917
587 586 600 0 5 0.917
587 588 600 1 5 0.917
588 587 600 0 5 0.917
588 589 600 1 5 0.917
589 588 600 0 5 0.917
589 590 600 1 5 0.917
590 589 600 0 5 0.917
590 654 600 1 5 0.917
654 590 600 0 5 0.917
83 111 400 0 4 6.000
149 111 400 0 4 6.000
163 111 400 0 4 6.000
119 111 400 0 4 6.000
86 103 400 0 4 6.000
166 103 400 0 4 6.000
154 103 400 0 4 6.000
122 103 400 0 4 6.000
171 40 400 0 4 6.000
90 40 400 0 4 6.000
123 40 400 0 4 6.000
219 39 400 0 4 6.000
188 39 400 0 4 6.000
236 39 400 0 4 6.000
```

```
146 39 400 0 4 6.000
276 37 400 0 4 6.000
347 37 400 0 4 6.000
349 37 400 0 4 6.000
345 37 400 0 4 6.000
364 7 400 0 4 6.000
393 7 400 0 4 6.000
379 7 400 0 4 6.000
526 448 400 0 4 6.000
450 448 400 0 4 6.000
432 448 400 0 4 6.000
445 448 400 0 4 6.000
223 453 400 0 4 6.000
240 453 400 0 4 6.000
225 453 400 0 4 6.000
193 453 400 0 4 6.000
226 474 400 0 4 6.000
243 474 400 0 4 6.000
228 474 400 0 4 6.000
196 474 400 0 4 6.000
8
0.0 5.0 10.0 15.0 20.0 25.0 30.0 35.0
1290.3 0.0 0.0 245.8 0.0 0.0
0.0 0.0 2.8 3.5 9.6 5.5
73.2 4.3 4.9 13.4 1.9 1.8
3.5 40.1 19.1 23.3 36.2 11.2
31.2 70.4 73.1 4.5 48.5 32.8
30.1 0.5 16.1 10.7 2.1 3.9
1290.3 0.0 0.0 245.8 0.0 0.0
0.0 0.0 2.8 3.5 9.6 5.5
73.2 4.3 4.9 13.4 1.9 1.8
3.5 40.1 19.1 23.3 36.2 11.2
31.2 70.4 73.1 4.5 48.5 32.8
30.1 0.5 16.1 10.7 2.1 3.9
645.2 0.0 0.0 122.9 0.0 0.0
0.0 0.0 1.4 1.8 4.8 2.7
36.6 2.2 2.5 6.7 0.9 0.9
1.7 20.0 9.6 11.6 18.1 5.6
15.6 35.2 36.5 2.3 24.2 16.4
15.1 0.3 8.0 5.4 1.1 1.9
967.7 0.0 0.0 184.4 0.0 0.0
0.0 0.0 2.1 2.7 7.2 4.1
54.9 3.3 3.7 10.1 1.4 1.4
2.6 30.0 14.3 17.4 27.2 8.4
23.4 52.8 54.8 3.4 36.4 24.6
22.6 0.4 12.1 8.0 1.6 2.9
1290.3 0.0 0.0 245.8 0.0 0.0
0.0 0.0 2.8 3.5 9.6 5.5
73.2 4.3 4.9 13.4 1.9 1.8
3.5 40.1 19.1 23.3 36.2 11.2
31.2 70.4 73.1 4.5 48.5 32.8
30.1 0.5 16.1 10.7 2.1 3.9
967.7 0.0 0.0 184.4 0.0 0.0
0.0 0.0 2.1 2.7 7.2 4.1
54.9 3.3 3.7 10.1 1.4 1.4
2.6 30.0 14.3 17.4 27.2 8.4
```

23.4	52.8	54.8	3.4	36.4	24.6
22.6	0.4	12.1	8.0	1.6	2.9
645.2	0.0	0.0	122.9	0.0	0.0
0.0	0.0	1.4	1.8	4.8	2.7
36.6	2.2	2.5	6.7	0.9	0.9
1.7	20.0	9.6	11.6	18.1	5.6
15.6	35.2	36.5	2.3	24.2	16.4
15.1	0.3	8.0	5.4	1.1	1.9
3870.9	0.0	0.0	737.4	0.0	0.0
0.0	0.0	8.5	10.6	28.9	16.5
219.7	13.0	14.8	40.3	5.6	5.5
10.4	120.2	57.3	69.8	108.7	33.7
93.5	211.3	219.2	13.6	145.4	98.4
90.3	1.5	48.2	32.2	6.3	11.6
0.0	1145.9	148.1	0.0	0.0	0.0
0.0	0.0	2.6	3.9	9.6	4.8
42.0	2.0	6.1	15.6	1.7	1.2
3.6	30.2	11.3	22.3	25.3	6.4
27.2	51.8	47.5	2.9	36.4	22.3
17.4	0.4	11.0	7.1	1.3	2.0
0.0	1145.9	148.1	0.0	0.0	0.0
0.0	0.0	2.6	3.9	9.6	4.8
42.0	2.0	6.1	15.6	1.7	1.2
3.6	30.2	11.3	22.3	25.3	6.4
27.2	51.8	47.5	2.9	36.4	22.3
17.4	0.4	11.0	7.1	1.3	2.0
0.0	572.9	74.0	0.0	0.0	0.0
0.0	0.0	1.3	1.9	4.8	2.4
21.0	1.0	3.1	7.8	0.9	0.6
1.8	15.1	5.7	11.1	12.7	3.2
13.6	25.9	23.8	1.5	18.2	11.2
8.7	0.2	5.5	3.6	0.6	1.0
0.0	859.4	111.1	0.0	0.0	0.0
0.0	0.0	1.9	2.9	7.2	3.6
31.5	1.5	4.6	11.7	1.3	0.9
2.7	22.7	8.5	16.7	19.0	4.8
20.4	38.9	35.7	2.2	27.3	16.7
13.1	0.3	8.3	5.3	0.9	1.5
0.0	1145.9	148.1	0.0	0.0	0.0
0.0	0.0	2.6	3.9	9.6	4.8
42.0	2.0	6.1	15.6	1.7	1.2
3.6	30.2	11.3	22.3	25.3	6.4
27.2	51.8	47.5	2.9	36.4	22.3
17.4	0.4	11.0	7.1	1.3	2.0
0.0	859.4	111.1	0.0	0.0	0.0
0.0	0.0	1.9	2.9	7.2	3.6
31.5	1.5	4.6	11.7	1.3	0.9
2.7	22.7	8.5	16.7	19.0	4.8
20.4	38.9	35.7	2.2	27.3	16.7
13.1	0.3	8.3	5.3	0.9	1.5
0.0	572.9	74.0	0.0	0.0	0.0
0.0	0.0	1.3	1.9	4.8	2.4
21.0	1.0	3.1	7.8	0.9	0.6
1.8	15.1	5.7	11.1	12.7	3.2
13.6	25.9	23.8	1.5	18.2	11.2
8.7	0.2	5.5	3.6	0.6	1.0

0.0	3437.7	444.3	0.0	0.0	0.0
0.0	0.0	7.8	11.7	28.9	14.5
126.0	5.9	18.4	46.9	5.2	3.7
10.7	90.7	34.0	66.9	75.9	19.3
81.7	155.4	142.6	8.7	109.3	67.0
52.2	1.3	33.0	21.3	3.8	6.0
0.0	147.7	2251.0	0.0	0.0	0.0
0.0	0.0	3.3	4.3	8.4	4.7
50.7	2.3	5.2	9.4	1.3	1.2
2.8	25.0	9.3	15.1	19.2	4.9
15.7	30.1	27.8	1.8	20.5	13.1
9.4	0.3	5.8	3.8	0.8	1.0
0.0	147.7	2251.0	0.0	0.0	0.0
0.0	0.0	3.3	4.3	8.4	4.7
50.7	2.3	5.2	9.4	1.3	1.2
2.8	25.0	9.3	15.1	19.2	4.9
15.7	30.1	27.8	1.8	20.5	13.1
9.4	0.3	5.8	3.8	0.8	1.0
0.0	73.9	1125.5	0.0	0.0	0.0
0.0	0.0	1.6	2.2	4.2	2.4
25.3	1.2	2.6	4.7	0.6	0.6
1.4	12.5	4.6	7.5	9.6	2.5
7.8	15.0	13.9	0.9	10.2	6.5
4.7	0.1	2.9	1.9	0.4	0.5
0.0	110.8	1688.3	0.0	0.0	0.0
0.0	0.0	2.5	3.2	6.3	3.5
38.0	1.7	3.9	7.1	1.0	0.9
2.1	18.7	7.0	11.3	14.4	3.7
11.7	22.6	20.9	1.4	15.4	9.8
7.0	0.2	4.4	2.9	0.6	0.7
0.0	147.7	2251.0	0.0	0.0	0.0
0.0	0.0	3.3	4.3	8.4	4.7
50.7	2.3	5.2	9.4	1.3	1.2
2.8	25.0	9.3	15.1	19.2	4.9
15.7	30.1	27.8	1.8	20.5	13.1
9.4	0.3	5.8	3.8	0.8	1.0
0.0	110.8	1688.3	0.0	0.0	0.0
0.0	0.0	2.5	3.2	6.3	3.5
38.0	1.7	3.9	7.1	1.0	0.9
2.1	18.7	7.0	11.3	14.4	3.7
11.7	22.6	20.9	1.4	15.4	9.8
7.0	0.2	4.4	2.9	0.6	0.7
0.0	73.9	1125.5	0.0	0.0	0.0
0.0	0.0	1.6	2.2	4.2	2.4
25.3	1.2	2.6	4.7	0.6	0.6
1.4	12.5	4.6	7.5	9.6	2.5
7.8	15.0	13.9	0.9	10.2	6.5
4.7	0.1	2.9	1.9	0.4	0.5
0.0	443.1	6753.1	0.0	0.0	0.0
0.0	0.0	9.9	13.0	25.1	14.2
152.0	7.0	15.7	28.3	3.9	3.7
8.3	74.9	27.9	45.2	57.6	14.8
47.0	90.3	83.4	5.5	61.4	39.2
28.2	0.8	17.5	11.5	2.3	3.0
247.7	0.0	0.0	2661.4	0.0	0.0
0.0	0.0	6.3	8.2	15.5	9.0

106.6	8.0	8.0	12.1	1.6	2.7
4.7	50.1	23.8	22.7	37.9	13.3
25.2	56.0	59.8	4.2	36.5	25.1
24.0	0.5	10.1	7.3	1.5	2.4
247.7	0.0	0.0	2661.4	0.0	0.0
0.0	0.0	6.3	8.2	15.5	9.0
106.6	8.0	8.0	12.1	1.6	2.7
4.7	50.1	23.8	22.7	37.9	13.3
25.2	56.0	59.8	4.2	36.5	25.1
24.0	0.5	10.1	7.3	1.5	2.4
123.8	0.0	0.0	1330.7	0.0	0.0
0.0	0.0	3.2	4.1	7.8	4.5
53.3	4.0	4.0	6.1	0.8	1.4
2.3	25.1	11.9	11.3	19.0	6.6
12.6	28.0	29.9	2.1	18.3	12.6
12.0	0.2	5.0	3.7	0.7	1.2
185.8	0.0	0.0	1996.1	0.0	0.0
0.0	0.0	4.8	6.2	11.6	6.7
79.9	6.0	6.0	9.1	1.2	2.0
3.5	37.6	17.9	17.0	28.4	10.0
18.9	42.0	44.9	3.1	27.4	18.9
18.0	0.3	7.5	5.5	1.1	1.8
247.7	0.0	0.0	2661.4	0.0	0.0
0.0	0.0	6.3	8.2	15.5	9.0
106.6	8.0	8.0	12.1	1.6	2.7
4.7	50.1	23.8	22.7	37.9	13.3
25.2	56.0	59.8	4.2	36.5	25.1
24.0	0.5	10.1	7.3	1.5	2.4
185.8	0.0	0.0	1996.1	0.0	0.0
0.0	0.0	4.8	6.2	11.6	6.7
79.9	6.0	6.0	9.1	1.2	2.0
3.5	37.6	17.9	17.0	28.4	10.0
18.9	42.0	44.9	3.1	27.4	18.9
18.0	0.3	7.5	5.5	1.1	1.8
123.8	0.0	0.0	1330.7	0.0	0.0
0.0	0.0	3.2	4.1	7.8	4.5
53.3	4.0	4.0	6.1	0.8	1.4
2.3	25.1	11.9	11.3	19.0	6.6
12.6	28.0	29.9	2.1	18.3	12.6
12.0	0.2	5.0	3.7	0.7	1.2
743.1	0.0	0.0	7984.3	0.0	0.0
0.0	0.0	19.0	24.7	46.5	27.0
319.7	24.0	24.0	36.3	4.8	8.2
14.0	150.4	71.5	68.0	113.7	39.9
75.7	168.0	179.5	12.5	109.6	75.4
72.1	1.4	30.2	21.9	4.5	7.1
0.0	0.0	0.0	0.0	41.7	0.0
0.0	0.0	1.2	2.0	3.4	1.8
19.5	0.7	3.1	6.0	0.4	0.3
1.2	8.4	3.6	7.9	7.2	1.9
6.4	12.1	10.7	0.6	7.9	4.6
3.7	0.1	2.3	1.4	0.2	0.4
0.0	0.0	0.0	0.0	41.7	0.0
0.0	0.0	1.2	2.0	3.4	1.8
19.5	0.7	3.1	6.0	0.4	0.3
1.2	8.4	3.6	7.9	7.2	1.9

6.4	12.1	10.7	0.6	7.9	4.6
3.7	0.1	2.3	1.4	0.2	0.4
0.0	0.0	0.0	0.0	20.8	0.0
0.0	0.0	0.6	1.0	1.7	0.9
9.8	0.4	1.6	3.0	0.2	0.1
0.6	4.2	1.8	3.9	3.6	1.0
3.2	6.1	5.4	0.3	4.0	2.3
1.9	0.1	1.2	0.7	0.1	0.2
0.0	0.0	0.0	0.0	31.3	0.0
0.0	0.0	0.9	1.5	2.6	1.4
14.6	0.5	2.3	4.5	0.3	0.2
0.9	6.3	2.7	5.9	5.4	1.4
4.8	9.1	8.0	0.5	5.9	3.4
2.8	0.1	1.7	1.1	0.2	0.3
0.0	0.0	0.0	0.0	41.7	0.0
0.0	0.0	1.2	2.0	3.4	1.8
19.5	0.7	3.1	6.0	0.4	0.3
1.2	8.4	3.6	7.9	7.2	1.9
6.4	12.1	10.7	0.6	7.9	4.6
3.7	0.1	2.3	1.4	0.2	0.4
0.0	0.0	0.0	0.0	31.3	0.0
0.0	0.0	0.9	1.5	2.6	1.4
14.6	0.5	2.3	4.5	0.3	0.2
0.9	6.3	2.7	5.9	5.4	1.4
4.8	9.1	8.0	0.5	5.9	3.4
2.8	0.1	1.7	1.1	0.2	0.3
0.0	0.0	0.0	0.0	20.8	0.0
0.0	0.0	0.6	1.0	1.7	0.9
9.8	0.4	1.6	3.0	0.2	0.1
0.6	4.2	1.8	3.9	3.6	1.0
3.2	6.1	5.4	0.3	4.0	2.3
1.9	0.1	1.2	0.7	0.1	0.2
0.0	0.0	0.0	0.0	125.1	0.0
0.0	0.0	3.7	6.0	10.3	5.5
58.5	2.1	9.4	17.9	1.3	0.9
3.7	25.3	10.7	23.7	21.7	5.8
19.3	36.4	32.2	1.9	23.8	13.7
11.2	0.3	7.0	4.3	0.7	1.2
0.0	0.0	0.0	0.0	0.0	121.8
0.0	0.0	3.7	5.8	6.9	4.0
56.6	4.0	5.0	5.8	0.6	0.7
2.2	15.7	8.1	9.7	11.3	3.9
8.2	17.4	17.3	1.0	10.1	6.4
6.8	0.1	2.7	1.9	0.3	0.8
0.0	0.0	0.0	0.0	0.0	121.8
0.0	0.0	3.7	5.8	6.9	4.0
56.6	4.0	5.0	5.8	0.6	0.7
2.2	15.7	8.1	9.7	11.3	3.9
8.2	17.4	17.3	1.0	10.1	6.4
6.8	0.1	2.7	1.9	0.3	0.8
0.0	0.0	0.0	0.0	0.0	60.9
0.0	0.0	1.9	2.9	3.5	2.0
28.3	2.0	2.5	2.9	0.3	0.4
1.1	7.8	4.1	4.9	5.7	2.0
4.1	8.7	8.7	0.5	5.1	3.2
3.4	0.1	1.4	1.0	0.2	0.4

0.0	0.0	0.0	0.0	0.0	91.4	26.2	1.9	3.1	5.7	0.7	1.0
0.0	0.0	2.8	4.3	5.2	3.0	1.8	19.3	11.3	9.7	16.6	6.6
42.4	3.0	3.8	4.3	0.4	0.6	11.2	30.5	33.8	2.4	18.2	13.9
1.7	11.7	6.1	7.3	8.5	3.0	15.7	0.2	5.7	4.3	0.8	2.1
6.2	13.0	13.0	0.8	7.6	4.8	0.0	0.0	0.0	0.0	0.0	0.0
5.1	0.1	2.1	1.5	0.2	0.6	146.4	0.0	1.4	2.0	3.9	2.5
0.0	0.0	0.0	0.0	0.0	121.8	19.6	1.5	2.3	4.3	0.5	0.7
0.0	0.0	3.7	5.8	6.9	4.0	1.4	14.5	8.4	7.2	12.5	5.0
56.6	4.0	5.0	5.8	0.6	0.7	8.4	22.8	25.4	1.8	13.7	10.4
2.2	15.7	8.1	9.7	11.3	3.9	11.7	0.1	4.3	3.2	0.6	1.5
8.2	17.4	17.3	1.0	10.1	6.4	0.0	0.0	0.0	0.0	0.0	0.0
6.8	0.1	2.7	1.9	0.3	0.8	97.6	0.0	0.9	1.3	2.6	1.6
0.0	0.0	0.0	0.0	0.0	91.4	13.1	1.0	1.5	2.8	0.3	0.5
0.0	0.0	2.8	4.3	5.2	3.0	0.9	9.7	5.6	4.8	8.3	3.3
42.4	3.0	3.8	4.3	0.4	0.6	5.6	15.2	16.9	1.2	9.1	6.9
1.7	11.7	6.1	7.3	8.5	3.0	7.8	0.1	2.8	2.1	0.4	1.0
6.2	13.0	13.0	0.8	7.6	4.8	0.0	0.0	0.0	0.0	0.0	0.0
5.1	0.1	2.1	1.5	0.2	0.6	585.7	0.0	5.6	8.0	15.7	9.8
0.0	0.0	0.0	0.0	0.0	60.9	78.5	5.8	9.2	17.0	2.0	2.9
0.0	0.0	1.9	2.9	3.5	2.0	5.5	58.0	33.8	29.0	49.9	19.9
28.3	2.0	2.5	2.9	0.3	0.4	33.5	91.4	101.5	7.1	54.7	41.6
1.1	7.8	4.1	4.9	5.7	2.0	47.0	0.6	17.0	12.8	2.3	6.2
4.1	8.7	8.7	0.5	5.1	3.2	0.0	0.0	0.0	0.0	0.0	0.0
3.4	0.1	1.4	1.0	0.2	0.4	0.0	3.6	0.3	0.6	0.9	0.5
0.0	0.0	0.0	0.0	0.0	365.5	3.4	0.2	0.7	1.7	0.1	0.1
0.0	0.0	11.2	17.4	20.8	11.9	0.4	2.4	1.2	1.9	2.3	0.6
169.7	12.0	15.1	17.4	1.7	2.2	2.5	5.2	4.9	0.3	3.8	2.4
6.7	47.0	24.4	29.2	34.0	11.8	2.0	0.1	1.2	0.9	0.1	0.4
24.7	52.1	52.0	3.1	30.4	19.1	0.0	0.0	0.0	0.0	0.0	0.0
20.3	0.3	8.2	5.8	1.0	2.5	0.0	3.6	0.3	0.6	0.9	0.5
0.0	0.0	0.0	0.0	0.0	0.0	3.4	0.2	0.7	1.7	0.1	0.1
195.2	0.0	1.9	2.7	5.2	3.3	0.4	2.4	1.2	1.9	2.3	0.6
26.2	1.9	3.1	5.7	0.7	1.0	2.5	5.2	4.9	0.3	3.8	2.4
1.8	19.3	11.3	9.7	16.6	6.6	2.0	0.1	1.2	0.9	0.1	0.4
11.2	30.5	33.8	2.4	18.2	13.9	0.0	0.0	0.0	0.0	0.0	0.0
15.7	0.2	5.7	4.3	0.8	2.1	0.0	1.8	0.1	0.3	0.5	0.3
0.0	0.0	0.0	0.0	0.0	0.0	1.7	0.1	0.4	0.9	0.1	0.0
195.2	0.0	1.9	2.7	5.2	3.3	0.2	1.2	0.6	0.9	1.1	0.3
26.2	1.9	3.1	5.7	0.7	1.0	1.3	2.6	2.4	0.1	1.9	1.2
1.8	19.3	11.3	9.7	16.6	6.6	1.0	0.0	0.6	0.5	0.1	0.2
11.2	30.5	33.8	2.4	18.2	13.9	0.0	0.0	0.0	0.0	0.0	0.0
15.7	0.2	5.7	4.3	0.8	2.1	0.0	2.7	0.2	0.4	0.7	0.4
0.0	0.0	0.0	0.0	0.0	0.0	2.6	0.2	0.5	1.3	0.1	0.0
97.6	0.0	0.9	1.3	2.6	1.6	0.3	1.8	0.9	1.4	1.7	0.5
13.1	1.0	1.5	2.8	0.3	0.5	1.9	3.9	3.7	0.2	2.8	1.8
0.9	9.7	5.6	4.8	8.3	3.3	1.5	0.0	0.9	0.7	0.1	0.3
5.6	15.2	16.9	1.2	9.1	6.9	0.0	0.0	0.0	0.0	0.0	0.0
7.8	0.1	2.8	2.1	0.4	1.0	0.0	3.6	0.3	0.6	0.9	0.5
0.0	0.0	0.0	0.0	0.0	0.0	3.4	0.2	0.7	1.7	0.1	0.1
146.4	0.0	1.4	2.0	3.9	2.5	0.4	2.4	1.2	1.9	2.3	0.6
19.6	1.5	2.3	4.3	0.5	0.7	2.5	5.2	4.9	0.3	3.8	2.4
1.4	14.5	8.4	7.2	12.5	5.0	2.0	0.1	1.2	0.9	0.1	0.4
8.4	22.8	25.4	1.8	13.7	10.4	0.0	0.0	0.0	0.0	0.0	0.0
11.7	0.1	4.3	3.2	0.6	1.5	0.0	2.7	0.2	0.4	0.7	0.4
0.0	0.0	0.0	0.0	0.0	0.0	2.6	0.2	0.5	1.3	0.1	0.0
195.2	0.0	1.9	2.7	5.2	3.3	0.3	1.8	0.9	1.4	1.7	0.5

1.9	3.9	3.7	0.2	2.8	1.8	0.0	0.0	0.0	0.0	0.0	0.0
1.5	0.0	0.9	0.7	0.1	0.3	0.0	0.0	0.3	0.7	1.3	0.7
0.0	0.0	0.0	0.0	0.0	0.0	14.4	0.5	0.8	0.8	0.1	0.1
0.0	1.8	0.1	0.3	0.5	0.3	0.3	2.6	1.3	2.0	2.3	0.5
1.7	0.1	0.4	0.9	0.1	0.0	1.4	3.3	2.5	0.1	1.4	0.9
0.2	1.2	0.6	0.9	1.1	0.3	0.9	0.0	0.4	0.2	0.0	0.1
1.3	2.6	2.4	0.1	1.9	1.2	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	0.6	0.5	0.1	0.2	0.0	0.0	0.3	0.3	0.8	0.4
0.0	0.0	0.0	0.0	0.0	0.0	12.8	0.2	0.4	0.6	0.0	0.0
0.0	10.9	0.8	1.7	2.8	1.5	0.2	1.8	0.7	1.5	1.2	0.3
10.3	0.7	2.2	5.2	0.4	0.2	1.0	1.7	1.7	0.1	1.0	0.5
1.3	7.1	3.5	5.7	6.9	1.8	0.5	0.0	0.3	0.1	0.0	0.0
7.6	15.7	14.6	0.8	11.3	7.2	0.0	0.0	0.0	0.0	0.0	0.0
6.1	0.2	3.7	2.8	0.4	1.2	0.0	0.0	0.3	0.3	0.8	0.4
0.0	0.0	0.0	0.0	0.0	0.0	12.8	0.2	0.4	0.6	0.0	0.0
0.0	0.0	0.1	0.2	0.4	0.2	0.2	1.8	0.7	1.5	1.2	0.3
4.8	0.2	0.3	0.3	0.0	0.0	1.0	1.7	1.7	0.1	1.0	0.5
0.1	0.9	0.4	0.7	0.8	0.2	0.5	0.0	0.3	0.1	0.0	0.0
0.5	1.1	0.8	0.0	0.5	0.3	0.0	0.0	0.0	0.0	0.0	0.0
0.3	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.1	0.4	0.2
0.0	0.0	0.0	0.0	0.0	0.0	6.4	0.1	0.2	0.3	0.0	0.0
0.0	0.0	0.1	0.2	0.4	0.2	0.1	0.9	0.4	0.7	0.6	0.1
4.8	0.2	0.3	0.3	0.0	0.0	0.5	0.9	0.9	0.0	0.5	0.2
0.1	0.9	0.4	0.7	0.8	0.2	0.2	0.0	0.1	0.1	0.0	0.0
0.5	1.1	0.8	0.0	0.5	0.3	0.0	0.0	0.0	0.0	0.0	0.0
0.3	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.2	0.2	0.6	0.3
0.0	0.0	0.0	0.0	0.0	0.0	9.6	0.1	0.3	0.5	0.0	0.0
0.0	0.0	0.0	0.1	0.2	0.1	0.1	1.4	0.5	1.1	0.9	0.2
2.4	0.1	0.1	0.1	0.0	0.0	0.7	1.3	1.3	0.0	0.7	0.4
0.1	0.4	0.2	0.3	0.4	0.1	0.3	0.0	0.2	0.1	0.0	0.0
0.2	0.5	0.4	0.0	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.2	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.3	0.3	0.8	0.4
0.0	0.0	0.0	0.0	0.0	0.0	12.8	0.2	0.4	0.6	0.0	0.0
0.0	0.0	0.1	0.2	0.3	0.2	0.2	1.8	0.7	1.5	1.2	0.3
3.6	0.1	0.2	0.2	0.0	0.0	1.0	1.7	1.7	0.1	1.0	0.5
0.1	0.6	0.3	0.5	0.6	0.1	0.5	0.0	0.3	0.1	0.0	0.0
0.4	0.8	0.6	0.0	0.3	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.2	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.2	0.2	0.6	0.3
0.0	0.0	0.0	0.0	0.0	0.0	9.6	0.1	0.3	0.5	0.0	0.0
0.0	0.0	0.1	0.2	0.4	0.2	0.1	1.4	0.5	1.1	0.9	0.2
4.8	0.2	0.3	0.3	0.0	0.0	0.7	1.3	1.3	0.0	0.7	0.4
0.1	0.9	0.4	0.7	0.8	0.2	0.3	0.0	0.2	0.1	0.0	0.0
0.5	1.1	0.8	0.0	0.5	0.3	0.0	0.0	0.0	0.0	0.0	0.0
0.3	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.1	0.4	0.2
0.0	0.0	0.0	0.0	0.0	0.0	6.4	0.1	0.2	0.3	0.0	0.0
0.0	0.0	0.1	0.2	0.3	0.2	0.1	0.9	0.4	0.7	0.6	0.1
3.6	0.1	0.2	0.2	0.0	0.0	0.5	0.9	0.9	0.0	0.5	0.2
0.1	0.6	0.3	0.5	0.6	0.1	0.2	0.0	0.1	0.1	0.0	0.0
0.4	0.8	0.6	0.0	0.3	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.2	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.8	0.8	2.3	1.3
0.0	0.0	0.0	0.0	0.0	0.0	38.3	0.5	1.3	1.9	0.1	0.1
0.0	0.0	0.0	0.1	0.2	0.1	0.6	5.4	2.1	4.4	3.7	0.9
2.4	0.1	0.1	0.1	0.0	0.0	3.0	5.2	5.1	0.2	2.9	1.4
0.1	0.4	0.2	0.3	0.4	0.1	1.4	0.0	0.8	0.4	0.1	0.1
0.2	0.5	0.4	0.0	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.0
0.2	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.4	0.8	0.7	0.6

14.2	0.3	0.8	1.0	0.1	0.1
0.4	3.0	1.3	2.3	2.1	0.5
1.8	3.1	2.3	0.1	1.8	0.9
0.9	0.0	0.4	0.3	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.4	0.8	0.7	0.6
14.2	0.3	0.8	1.0	0.1	0.1
0.4	3.0	1.3	2.3	2.1	0.5
1.8	3.1	2.3	0.1	1.8	0.9
0.9	0.0	0.4	0.3	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.4	0.4	0.3
7.1	0.1	0.4	0.5	0.1	0.0
0.2	1.5	0.6	1.2	1.0	0.2
0.9	1.6	1.1	0.1	0.9	0.4
0.4	0.0	0.2	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.3	0.6	0.6	0.5
10.6	0.2	0.6	0.7	0.1	0.1
0.3	2.2	1.0	1.7	1.6	0.4
1.3	2.3	1.7	0.1	1.3	0.7
0.6	0.0	0.3	0.2	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.4	0.8	0.7	0.6
14.2	0.3	0.8	1.0	0.1	0.1
0.4	3.0	1.3	2.3	2.1	0.5
1.8	3.1	2.3	0.1	1.8	0.9
0.9	0.0	0.4	0.3	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.3	0.6	0.6	0.5
10.6	0.2	0.6	0.7	0.1	0.1
0.3	2.2	1.0	1.7	1.6	0.4
1.3	2.3	1.7	0.1	1.3	0.7
0.6	0.0	0.3	0.2	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.4	0.4	0.3
7.1	0.1	0.4	0.5	0.1	0.0
0.2	1.5	0.6	1.2	1.0	0.2
0.9	1.6	1.1	0.1	0.9	0.4
0.4	0.0	0.2	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	1.3	2.4	2.2	1.9
42.6	0.8	2.5	3.0	0.4	0.2
1.1	9.0	3.8	7.0	6.2	1.4
5.4	9.4	6.9	0.3	5.3	2.7
2.6	0.1	1.2	0.8	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.4	0.6	0.2
9.9	0.3	0.4	0.5	0.1	0.0
0.2	1.5	0.8	1.1	1.0	0.3
0.9	1.5	1.5	0.1	0.9	0.6
0.4	0.0	0.2	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.4	0.6	0.2
9.9	0.3	0.4	0.5	0.1	0.0
0.2	1.5	0.8	1.1	1.0	0.3

0.9	1.5	1.5	0.1	0.9	0.6
0.4	0.0	0.2	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.3	0.1
5.0	0.1	0.2	0.3	0.0	0.0
0.1	0.7	0.4	0.5	0.5	0.2
0.4	0.8	0.7	0.0	0.4	0.3
0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.3	0.5	0.2
7.5	0.2	0.3	0.4	0.0	0.0
0.2	1.1	0.6	0.8	0.7	0.2
0.7	1.1	1.1	0.1	0.7	0.4
0.3	0.0	0.2	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.4	0.6	0.2
9.9	0.3	0.4	0.5	0.1	0.0
0.2	1.5	0.8	1.1	1.0	0.3
0.9	1.5	1.5	0.1	0.9	0.6
0.4	0.0	0.2	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.3	0.5	0.2
7.5	0.2	0.3	0.4	0.0	0.0
0.2	1.1	0.6	0.8	0.7	0.2
0.7	1.1	1.1	0.1	0.7	0.4
0.3	0.0	0.2	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.3	0.1
5.0	0.1	0.2	0.3	0.0	0.0
0.1	0.7	0.4	0.5	0.5	0.2
0.4	0.8	0.7	0.0	0.4	0.3
0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.7	1.3	1.9	0.7
29.8	0.9	1.3	1.5	0.2	0.1
0.7	4.4	2.3	3.2	3.0	1.0
2.6	4.6	4.4	0.2	2.7	1.7
1.3	0.0	0.7	0.4	0.0	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	4.8	12.7	14.2	9.9
59.6	3.6	2.4	5.5	0.3	0.4
1.6	8.5	5.3	6.0	6.7	2.8
5.0	14.1	13.6	0.7	6.3	4.0
4.0	0.0	1.7	1.7	0.2	0.9
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	4.8	12.7	14.2	9.9
59.6	3.6	2.4	5.5	0.3	0.4
1.6	8.5	5.3	6.0	6.7	2.8
5.0	14.1	13.6	0.7	6.3	4.0
4.0	0.0	1.7	1.7	0.2	0.9
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	2.4	6.4	7.1	5.0
29.8	1.8	1.2	2.8	0.2	0.2
0.8	4.3	2.6	3.0	3.3	1.4
2.5	7.1	6.8	0.3	3.1	2.0
2.0	0.0	0.9	0.9	0.1	0.4

0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	3.6	9.6	10.6	7.5
44.7	2.7	1.8	4.2	0.2	0.3
1.2	6.4	3.9	4.5	5.0	2.1
3.7	10.6	10.2	0.5	4.7	3.0
3.0	0.0	1.3	1.3	0.2	0.7
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	4.8	12.7	14.2	9.9
59.6	3.6	2.4	5.5	0.3	0.4
1.6	8.5	5.3	6.0	6.7	2.8
5.0	14.1	13.6	0.7	6.3	4.0
4.0	0.0	1.7	1.7	0.2	0.9
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	3.6	9.6	10.6	7.5
44.7	2.7	1.8	4.2	0.2	0.3
1.2	6.4	3.9	4.5	5.0	2.1
3.7	10.6	10.2	0.5	4.7	3.0
3.0	0.0	1.3	1.3	0.2	0.7
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	2.4	6.4	7.1	5.0
29.8	1.8	1.2	2.8	0.2	0.2
0.8	4.3	2.6	3.0	3.3	1.4
2.5	7.1	6.8	0.3	3.1	2.0
2.0	0.0	0.9	0.9	0.1	0.4
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	14.4	38.2	42.6	29.8
178.8	10.7	7.3	16.6	0.9	1.2
4.9	25.6	15.8	17.9	20.1	8.4
14.9	42.3	40.7	2.1	18.9	12.1
12.0	0.1	5.2	5.2	0.6	2.6
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.2	0.2	0.3
3.6	0.1	0.1	0.3	0.0	0.0
0.1	0.6	0.4	0.2	0.5	0.2
0.3	0.6	0.8	0.0	0.4	0.3
0.3	0.0	0.1	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.2	0.2	0.3
3.6	0.1	0.1	0.3	0.0	0.0
0.1	0.6	0.4	0.2	0.5	0.2
0.3	0.6	0.8	0.0	0.4	0.3
0.3	0.0	0.1	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.1	0.2
1.8	0.1	0.0	0.1	0.0	0.0
0.0	0.3	0.2	0.1	0.2	0.1
0.2	0.3	0.4	0.0	0.2	0.2
0.1	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.2	0.2
2.7	0.1	0.1	0.2	0.0	0.0
0.0	0.4	0.3	0.2	0.4	0.1
0.2	0.5	0.6	0.0	0.3	0.2
0.2	0.0	0.1	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.2	0.2	0.3

3.6	0.1	0.1	0.3	0.0	0.0
0.1	0.6	0.4	0.2	0.5	0.2
0.3	0.6	0.8	0.0	0.4	0.3
0.3	0.0	0.1	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.2	0.2
2.7	0.1	0.1	0.2	0.0	0.0
0.0	0.4	0.3	0.2	0.4	0.1
0.2	0.5	0.6	0.0	0.3	0.2
0.2	0.0	0.1	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.1	0.2
1.8	0.1	0.0	0.1	0.0	0.0
0.0	0.3	0.2	0.1	0.2	0.1
0.2	0.3	0.4	0.0	0.2	0.2
0.1	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	0.6	0.7	0.9
10.8	0.3	0.3	0.8	0.0	0.1
0.2	1.7	1.1	0.7	1.4	0.5
0.9	1.9	2.3	0.1	1.1	1.0
0.9	0.0	0.3	0.3	0.0	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.3	0.4	0.8	0.4
2.4	0.1	0.6	1.2	0.1	0.0
0.3	1.8	0.6	2.0	1.5	0.3
1.2	2.1	1.7	0.1	1.3	0.7
0.6	0.0	0.4	0.2	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.3	0.4	0.8	0.4
2.4	0.1	0.6	1.2	0.1	0.0
0.3	1.8	0.6	2.0	1.5	0.3
1.2	2.1	1.7	0.1	1.3	0.7
0.6	0.0	0.4	0.2	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.4	0.2
1.2	0.0	0.3	0.6	0.1	0.0
0.1	0.9	0.3	1.0	0.7	0.2
0.6	1.1	0.8	0.0	0.6	0.3
0.3	0.0	0.2	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.3	0.6	0.3
1.8	0.1	0.5	0.9	0.1	0.0
0.2	1.3	0.5	1.5	1.1	0.3
0.9	1.6	1.3	0.1	0.9	0.5
0.4	0.0	0.3	0.2	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.3	0.4	0.8	0.4
2.4	0.1	0.6	1.2	0.1	0.0
0.3	1.8	0.6	2.0	1.5	0.3
1.2	2.1	1.7	0.1	1.3	0.7
0.6	0.0	0.4	0.2	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.3	0.6	0.3
1.8	0.1	0.5	0.9	0.1	0.0
0.2	1.3	0.5	1.5	1.1	0.3

0.9	1.6	1.3	0.1	0.9	0.5	0.0	0.0	0.0	0.0	0.0	0.0
0.4	0.0	0.3	0.2	0.0	0.0	0.0	0.0	0.7	2.0	3.0	1.6
0.0	0.0	0.0	0.0	0.0	0.0	16.6	0.9	3.7	5.6	0.5	0.2
0.0	0.0	0.1	0.2	0.4	0.2	1.5	6.7	2.7	8.3	5.7	1.3
1.2	0.0	0.3	0.6	0.1	0.0	6.6	11.7	8.6	0.4	6.5	4.2
0.1	0.9	0.3	1.0	0.7	0.2	4.1	0.1	1.9	1.3	0.2	0.4
0.6	1.1	0.8	0.0	0.6	0.3	0.0	0.0	0.0	0.0	0.0	0.0
0.3	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.1	0.2	0.0	0.0
0.0	0.0	0.8	1.3	2.5	1.3	0.0	0.2	0.1	0.2	0.2	0.0
7.2	0.3	1.8	3.7	0.3	0.1	0.2	0.3	0.3	0.0	0.3	0.1
0.9	5.3	1.9	5.9	4.5	1.0	0.1	0.0	0.1	0.0	0.0	0.0
3.7	6.4	5.1	0.2	3.8	2.0	0.0	0.0	0.0	0.0	0.0	0.0
1.8	0.0	1.1	0.6	0.1	0.2	0.0	0.0	0.0	0.1	0.1	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.1	0.2	0.0	0.0
0.0	0.0	0.2	0.7	1.0	0.5	0.0	0.2	0.1	0.2	0.2	0.0
5.5	0.3	1.2	1.9	0.2	0.1	0.2	0.3	0.3	0.0	0.3	0.1
0.5	2.2	0.9	2.8	1.9	0.4	0.1	0.0	0.1	0.0	0.0	0.0
2.2	3.9	2.9	0.1	2.2	1.4	0.0	0.0	0.0	0.0	0.0	0.0
1.4	0.0	0.6	0.4	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.2	0.7	1.0	0.5	0.0	0.1	0.0	0.1	0.1	0.0
5.5	0.3	1.2	1.9	0.2	0.1	0.1	0.2	0.2	0.0	0.1	0.1
0.5	2.2	0.9	2.8	1.9	0.4	0.1	0.0	0.0	0.0	0.0	0.0
2.2	3.9	2.9	0.1	2.2	1.4	0.0	0.0	0.0	0.0	0.0	0.0
1.4	0.0	0.6	0.4	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.1	0.3	0.5	0.3	0.0	0.1	0.1	0.1	0.1	0.0
2.8	0.1	0.6	0.9	0.1	0.0	0.2	0.3	0.3	0.0	0.2	0.1
0.2	1.1	0.4	1.4	1.0	0.2	0.1	0.0	0.1	0.0	0.0	0.0
1.1	1.9	1.4	0.1	1.1	0.7	0.0	0.0	0.0	0.0	0.0	0.0
0.7	0.0	0.3	0.2	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.1	0.2	0.0	0.0
0.0	0.0	0.2	0.5	0.7	0.4	0.0	0.2	0.1	0.2	0.2	0.0
4.2	0.2	0.9	1.4	0.1	0.1	0.2	0.3	0.3	0.0	0.3	0.1
0.4	1.7	0.7	2.1	1.4	0.3	0.1	0.0	0.1	0.0	0.0	0.0
1.7	2.9	2.1	0.1	1.6	1.1	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	0.5	0.3	0.1	0.1	0.0	0.0	0.0	0.1	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.2	0.7	1.0	0.5	0.0	0.1	0.1	0.1	0.1	0.0
5.5	0.3	1.2	1.9	0.2	0.1	0.2	0.3	0.3	0.0	0.2	0.1
0.5	2.2	0.9	2.8	1.9	0.4	0.1	0.0	0.1	0.0	0.0	0.0
2.2	3.9	2.9	0.1	2.2	1.4	0.0	0.0	0.0	0.0	0.0	0.0
1.4	0.0	0.6	0.4	0.1	0.1	0.0	0.0	0.0	0.0	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.2	0.5	0.7	0.4	0.0	0.1	0.0	0.1	0.1	0.0
4.2	0.2	0.9	1.4	0.1	0.1	0.1	0.2	0.2	0.0	0.1	0.1
0.4	1.7	0.7	2.1	1.4	0.3	0.1	0.0	0.0	0.0	0.0	0.0
1.7	2.9	2.1	0.1	1.6	1.1	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	0.5	0.3	0.1	0.1	0.0	0.0	0.1	0.2	0.3	0.2
0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.3	0.5	0.0	0.0
0.0	0.0	0.1	0.3	0.5	0.3	0.1	0.6	0.3	0.6	0.5	0.1
2.8	0.1	0.6	0.9	0.1	0.0	0.6	1.0	1.0	0.1	0.8	0.3
0.2	1.1	0.4	1.4	1.0	0.2	0.4	0.0	0.2	0.1	0.0	0.0
1.1	1.9	1.4	0.1	1.1	0.7	0.0	0.0	0.0	0.0	0.0	0.0
0.7	0.0	0.3	0.2	0.0	0.1	0.0	0.0	0.0	0.1	0.1	0.1

0.4	0.0	0.1	0.1	0.0	0.0
0.0	0.2	0.1	0.1	0.1	0.0
0.1	0.2	0.2	0.0	0.1	0.1
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.1	0.1
0.4	0.0	0.1	0.1	0.0	0.0
0.0	0.2	0.1	0.1	0.1	0.0
0.1	0.2	0.2	0.0	0.1	0.1
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.2	0.0	0.0	0.0	0.0	0.0
0.0	0.1	0.0	0.0	0.1	0.0
0.1	0.1	0.1	0.0	0.1	0.0
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.0
0.3	0.0	0.0	0.0	0.0	0.0
0.0	0.1	0.1	0.0	0.1	0.0
0.1	0.2	0.2	0.0	0.1	0.0
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.1	0.1
0.4	0.0	0.1	0.1	0.0	0.0
0.0	0.2	0.1	0.1	0.1	0.0
0.1	0.2	0.2	0.0	0.1	0.1
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.0
0.3	0.0	0.0	0.0	0.0	0.0
0.0	0.1	0.1	0.0	0.1	0.0
0.1	0.2	0.2	0.0	0.1	0.0
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.2	0.0	0.0	0.0	0.0	0.0
0.0	0.1	0.0	0.0	0.1	0.0
0.1	0.1	0.1	0.0	0.1	0.0
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.3	0.2
1.2	0.0	0.2	0.2	0.0	0.0
0.1	0.6	0.3	0.2	0.4	0.1
0.3	0.7	0.7	0.0	0.3	0.2
0.3	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.4	0.2
1.6	0.1	0.3	0.5	0.1	0.0
0.1	1.0	0.4	0.6	0.9	0.2
0.8	1.4	1.0	0.1	0.6	0.4
0.4	0.0	0.2	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.4	0.2
1.6	0.1	0.3	0.5	0.1	0.0
0.1	1.0	0.4	0.6	0.9	0.2

0.8	1.4	1.0	0.1	0.6	0.4
0.4	0.0	0.2	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.2	0.1
0.8	0.0	0.1	0.2	0.0	0.0
0.0	0.5	0.2	0.3	0.4	0.1
0.4	0.7	0.5	0.0	0.3	0.2
0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.3	0.2
1.2	0.0	0.2	0.4	0.0	0.0
0.1	0.8	0.3	0.5	0.7	0.2
0.6	1.0	0.7	0.0	0.5	0.3
0.3	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.4	0.2
1.6	0.1	0.3	0.5	0.1	0.0
0.1	1.0	0.4	0.6	0.9	0.2
0.8	1.4	1.0	0.1	0.6	0.4
0.4	0.0	0.2	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.3	0.2
1.2	0.0	0.2	0.4	0.0	0.0
0.1	0.8	0.3	0.5	0.7	0.2
0.6	1.0	0.7	0.0	0.5	0.3
0.3	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.2	0.1
0.8	0.0	0.1	0.2	0.0	0.0
0.0	0.5	0.2	0.3	0.4	0.1
0.4	0.7	0.5	0.0	0.3	0.2
0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.4	0.6	1.1	0.7
4.9	0.2	0.9	1.5	0.2	0.1
0.3	3.1	1.2	1.9	2.7	0.7
2.3	4.1	3.0	0.2	1.9	1.2
1.1	0.0	0.6	0.3	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.8	1.8	3.0	1.5
8.5	0.6	1.8	2.2	0.2	0.2
1.0	0.0	2.0	3.0	3.0	1.0
3.6	7.5	7.3	0.4	3.8	2.4
2.3	0.0	1.0	0.9	0.1	0.4
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.8	1.8	3.0	1.5
8.5	0.6	1.8	2.2	0.2	0.2
1.0	0.0	2.0	3.0	3.0	1.0
3.6	7.5	7.3	0.4	3.8	2.4
2.3	0.0	1.0	0.9	0.1	0.4
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.4	0.9	1.5	0.7
4.3	0.3	0.9	1.1	0.1	0.1
0.5	0.0	1.0	1.5	1.5	0.5
1.8	3.7	3.7	0.2	1.9	1.2
1.1	0.0	0.5	0.4	0.1	0.2

0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.6	1.4	2.2	1.1
6.4	0.4	1.3	1.7	0.1	0.1
0.8	0.0	1.5	2.2	2.3	0.8
2.7	5.6	5.5	0.3	2.8	1.8
1.7	0.0	0.8	0.7	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.8	1.8	3.0	1.5
8.5	0.6	1.8	2.2	0.2	0.2
1.0	0.0	2.0	3.0	3.0	1.0
3.6	7.5	7.3	0.4	3.8	2.4
2.3	0.0	1.0	0.9	0.1	0.4
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.6	1.4	2.2	1.1
6.4	0.4	1.3	1.7	0.1	0.1
0.8	0.0	1.5	2.2	2.3	0.8
2.7	5.6	5.5	0.3	2.8	1.8
1.7	0.0	0.8	0.7	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.4	0.9	1.5	0.7
4.3	0.3	0.9	1.1	0.1	0.1
0.5	0.0	1.0	1.5	1.5	0.5
1.8	3.7	3.7	0.2	1.9	1.2
1.1	0.0	0.5	0.4	0.1	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	2.5	5.4	8.9	4.4
25.6	1.7	5.3	6.6	0.6	0.6
3.1	0.0	5.9	9.0	9.0	3.1
10.7	22.4	22.0	1.1	11.4	7.3
6.8	0.1	3.1	2.7	0.4	1.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	0.7	1.3	0.8
5.3	0.4	0.6	0.9	0.1	0.1
0.4	2.0	0.0	1.2	1.5	0.5
1.4	3.2	3.8	0.2	1.9	1.3
1.4	0.0	0.5	0.4	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	0.7	1.3	0.8
5.3	0.4	0.6	0.9	0.1	0.1
0.4	2.0	0.0	1.2	1.5	0.5
1.4	3.2	3.8	0.2	1.9	1.3
1.4	0.0	0.5	0.4	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.4	0.6	0.4
2.6	0.2	0.3	0.4	0.0	0.0
0.2	1.0	0.0	0.6	0.7	0.3
0.7	1.6	1.9	0.1	1.0	0.6
0.7	0.0	0.3	0.2	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.3	0.5	0.9	0.6
3.9	0.3	0.5	0.7	0.1	0.0
0.3	1.5	0.0	0.9	1.1	0.4
1.1	2.4	2.8	0.1	1.5	1.0
1.1	0.0	0.4	0.3	0.0	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	0.7	1.3	0.8

5.3	0.4	0.6	0.9	0.1	0.1
0.4	2.0	0.0	1.2	1.5	0.5
1.4	3.2	3.8	0.2	1.9	1.3
1.4	0.0	0.5	0.4	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.3	0.5	0.9	0.6
3.9	0.3	0.5	0.7	0.1	0.0
0.3	1.5	0.0	0.9	1.1	0.4
1.1	2.4	2.8	0.1	1.5	1.0
1.1	0.0	0.4	0.3	0.0	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.4	0.6	0.4
2.6	0.2	0.3	0.4	0.0	0.0
0.2	1.0	0.0	0.6	0.7	0.3
0.7	1.6	1.9	0.1	1.0	0.6
0.7	0.0	0.3	0.2	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	1.4	2.2	3.8	2.3
15.8	1.1	1.9	2.7	0.2	0.2
1.3	5.9	0.0	3.7	4.5	1.6
4.3	9.6	11.3	0.6	5.8	3.8
4.3	0.0	1.6	1.3	0.2	0.8
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.7	1.5	2.3	1.1
6.0	0.2	2.0	2.8	0.2	0.1
0.6	3.0	1.2	0.0	2.3	0.7
2.4	5.0	3.9	0.2	2.5	1.6
1.2	0.0	0.7	0.6	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.7	1.5	2.3	1.1
6.0	0.2	2.0	2.8	0.2	0.1
0.6	3.0	1.2	0.0	2.3	0.7
2.4	5.0	3.9	0.2	2.5	1.6
1.2	0.0	0.7	0.6	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.3	0.7	1.1	0.5
3.0	0.1	1.0	1.4	0.1	0.0
0.3	1.5	0.6	0.0	1.2	0.3
1.2	2.5	2.0	0.1	1.3	0.8
0.6	0.0	0.3	0.3	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	1.1	1.7	0.8
4.5	0.2	1.5	2.1	0.1	0.1
0.5	2.2	0.9	0.0	1.8	0.5
1.8	3.7	2.9	0.1	1.9	1.2
0.9	0.0	0.5	0.5	0.0	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.7	1.5	2.3	1.1
6.0	0.2	2.0	2.8	0.2	0.1
0.6	3.0	1.2	0.0	2.3	0.7
2.4	5.0	3.9	0.2	2.5	1.6
1.2	0.0	0.7	0.6	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	1.1	1.7	0.8
4.5	0.2	1.5	2.1	0.1	0.1
0.5	2.2	0.9	0.0	1.8	0.5

1.8	3.7	2.9	0.1	1.9	1.2	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.5	0.5	0.0	0.2	0.0	0.0	2.2	3.7	6.3	2.9
0.0	0.0	0.0	0.0	0.0	0.0	20.1	1.3	4.4	5.7	0.5	0.3
0.0	0.0	0.3	0.7	1.1	0.5	2.7	9.1	4.5	7.0	0.0	2.4
3.0	0.1	1.0	1.4	0.1	0.0	8.3	16.0	14.6	0.9	11.0	7.0
0.3	1.5	0.6	0.0	1.2	0.3	6.6	0.0	2.9	2.7	0.3	1.3
1.2	2.5	2.0	0.1	1.3	0.8	0.0	0.0	0.0	0.0	0.0	0.0
0.6	0.0	0.3	0.3	0.0	0.1	0.0	0.0	0.2	0.3	0.5	0.3
0.0	0.0	0.0	0.0	0.0	0.0	2.8	0.2	0.3	0.5	0.0	0.0
0.0	0.0	2.0	4.5	6.9	3.2	0.2	1.0	0.5	0.6	0.8	0.0
17.9	0.7	5.9	8.3	0.6	0.3	0.8	1.7	2.0	0.1	1.0	0.7
1.9	9.0	3.7	0.0	7.0	2.0	0.8	0.0	0.3	0.2	0.0	0.1
7.1	15.0	11.8	0.5	7.5	4.9	0.0	0.0	0.0	0.0	0.0	0.0
3.6	0.1	2.0	1.9	0.2	0.8	0.0	0.0	0.2	0.3	0.5	0.3
0.0	0.0	0.0	0.0	0.0	0.0	2.8	0.2	0.3	0.5	0.0	0.0
0.0	0.0	0.7	1.2	2.1	1.0	0.2	1.0	0.5	0.6	0.8	0.0
6.7	0.4	1.5	1.9	0.2	0.1	0.8	1.7	2.0	0.1	1.0	0.7
0.9	3.0	1.5	2.3	0.0	0.8	0.8	0.0	0.3	0.2	0.0	0.1
2.8	5.3	4.9	0.3	3.7	2.3	0.0	0.0	0.0	0.0	0.0	0.0
2.2	0.0	1.0	0.9	0.1	0.4	0.0	0.0	0.1	0.1	0.3	0.2
0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.1	0.2	0.2	0.0	0.0
0.0	0.0	0.7	1.2	2.1	1.0	0.1	0.5	0.3	0.3	0.4	0.0
6.7	0.4	1.5	1.9	0.2	0.1	0.4	0.8	1.0	0.1	0.5	0.3
0.9	3.0	1.5	2.3	0.0	0.8	0.4	0.0	0.1	0.1	0.0	0.1
2.8	5.3	4.9	0.3	3.7	2.3	0.0	0.0	0.0	0.0	0.0	0.0
2.2	0.0	1.0	0.9	0.1	0.4	0.0	0.0	0.1	0.2	0.4	0.2
0.0	0.0	0.0	0.0	0.0	0.0	2.1	0.1	0.2	0.3	0.0	0.0
0.0	0.0	0.4	0.6	1.0	0.5	0.2	0.8	0.4	0.5	0.6	0.0
3.4	0.2	0.7	1.0	0.1	0.1	0.6	1.2	1.5	0.1	0.8	0.5
0.4	1.5	0.8	1.2	0.0	0.4	0.6	0.0	0.2	0.2	0.0	0.1
1.4	2.7	2.4	0.2	1.8	1.2	0.0	0.0	0.0	0.0	0.0	0.0
1.1	0.0	0.5	0.5	0.1	0.2	0.0	0.0	0.2	0.3	0.5	0.3
0.0	0.0	0.0	0.0	0.0	0.0	2.8	0.2	0.3	0.5	0.0	0.0
0.0	0.0	0.6	0.9	1.6	0.7	0.2	1.0	0.5	0.6	0.8	0.0
5.0	0.3	1.1	1.4	0.1	0.1	0.8	1.7	2.0	0.1	1.0	0.7
0.7	2.3	1.1	1.8	0.0	0.6	0.8	0.0	0.3	0.2	0.0	0.1
2.1	4.0	3.6	0.2	2.7	1.8	0.0	0.0	0.0	0.0	0.0	0.0
1.7	0.0	0.7	0.7	0.1	0.3	0.0	0.0	0.1	0.2	0.4	0.2
0.0	0.0	0.0	0.0	0.0	0.0	2.1	0.1	0.2	0.3	0.0	0.0
0.0	0.0	0.7	1.2	2.1	1.0	0.2	0.8	0.4	0.5	0.6	0.0
6.7	0.4	1.5	1.9	0.2	0.1	0.6	1.2	1.5	0.1	0.8	0.5
0.9	3.0	1.5	2.3	0.0	0.8	0.6	0.0	0.2	0.2	0.0	0.1
2.8	5.3	4.9	0.3	3.7	2.3	0.0	0.0	0.0	0.0	0.0	0.0
2.2	0.0	1.0	0.9	0.1	0.4	0.0	0.0	0.1	0.1	0.3	0.2
0.0	0.0	0.0	0.0	0.0	0.0	1.4	0.1	0.2	0.2	0.0	0.0
0.0	0.0	0.6	0.9	1.6	0.7	0.1	0.5	0.3	0.3	0.4	0.0
5.0	0.3	1.1	1.4	0.1	0.1	0.4	0.8	1.0	0.1	0.5	0.3
0.7	2.3	1.1	1.8	0.0	0.6	0.4	0.0	0.1	0.1	0.0	0.1
2.1	4.0	3.6	0.2	2.7	1.8	0.0	0.0	0.0	0.0	0.0	0.0
1.7	0.0	0.7	0.7	0.1	0.3	0.0	0.0	0.5	0.9	1.5	1.0
0.0	0.0	0.0	0.0	0.0	0.0	8.3	0.6	1.0	1.4	0.1	0.1
0.0	0.0	0.4	0.6	1.0	0.5	0.7	3.1	1.6	1.9	2.4	0.0
3.4	0.2	0.7	1.0	0.1	0.1	2.3	5.0	5.9	0.3	3.0	2.0
0.4	1.5	0.8	1.2	0.0	0.4	2.3	0.0	0.8	0.7	0.1	0.4
1.4	2.7	2.4	0.2	1.8	1.2	0.0	0.0	0.0	0.0	0.0	0.0
1.1	0.0	0.5	0.5	0.1	0.2	0.0	0.0	0.5	1.0	1.8	0.9

5.0	0.3	1.2	2.2	0.2	0.1	5.9	0.0	11.3	0.6	7.4	4.8
0.8	3.5	1.4	2.4	2.8	0.8	4.8	0.1	2.1	1.8	0.2	0.9
0.0	5.9	5.8	0.2	3.7	2.4	0.0	0.0	0.0	0.0	0.0	0.0
1.8	0.0	1.0	0.9	0.1	0.3	0.0	0.0	0.6	0.9	1.6	0.8
0.0	0.0	0.0	0.0	0.0	0.0	7.1	0.3	1.1	1.9	0.2	0.1
0.0	0.0	0.5	1.0	1.8	0.9	0.7	3.7	1.6	2.5	2.7	0.8
5.0	0.3	1.2	2.2	0.2	0.1	2.9	0.0	5.7	0.3	3.7	2.4
0.8	3.5	1.4	2.4	2.8	0.8	2.4	0.0	1.1	0.9	0.1	0.4
0.0	5.9	5.8	0.2	3.7	2.4	0.0	0.0	0.0	0.0	0.0	0.0
1.8	0.0	1.0	0.9	0.1	0.3	0.0	0.0	0.8	1.3	2.3	1.1
0.0	0.0	0.0	0.0	0.0	0.0	10.6	0.5	1.6	2.9	0.3	0.2
0.0	0.0	0.2	0.5	0.9	0.4	1.0	5.6	2.4	3.7	4.0	1.2
2.5	0.1	0.6	1.1	0.1	0.0	4.4	0.0	8.5	0.5	5.5	3.6
0.4	1.8	0.7	1.2	1.4	0.4	3.6	0.1	1.6	1.4	0.2	0.6
0.0	2.9	2.9	0.1	1.8	1.2	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.5	0.4	0.0	0.1	0.0	0.0	1.1	1.7	3.1	1.5
0.0	0.0	0.0	0.0	0.0	0.0	14.1	0.7	2.1	3.9	0.3	0.2
0.0	0.0	0.3	0.7	1.4	0.7	1.4	7.5	3.2	5.0	5.3	1.7
3.7	0.2	0.9	1.7	0.1	0.1	5.9	0.0	11.3	0.6	7.4	4.8
0.6	2.7	1.1	1.8	2.1	0.6	4.8	0.1	2.1	1.8	0.2	0.9
0.0	4.4	4.3	0.2	2.8	1.8	0.0	0.0	0.0	0.0	0.0	0.0
1.3	0.0	0.8	0.6	0.1	0.2	0.0	0.0	0.8	1.3	2.3	1.1
0.0	0.0	0.0	0.0	0.0	0.0	10.6	0.5	1.6	2.9	0.3	0.2
0.0	0.0	0.5	1.0	1.8	0.9	1.0	5.6	2.4	3.7	4.0	1.2
5.0	0.3	1.2	2.2	0.2	0.1	4.4	0.0	8.5	0.5	5.5	3.6
0.8	3.5	1.4	2.4	2.8	0.8	3.6	0.1	1.6	1.4	0.2	0.6
0.0	5.9	5.8	0.2	3.7	2.4	0.0	0.0	0.0	0.0	0.0	0.0
1.8	0.0	1.0	0.9	0.1	0.3	0.0	0.0	0.6	0.9	1.6	0.8
0.0	0.0	0.0	0.0	0.0	0.0	7.1	0.3	1.1	1.9	0.2	0.1
0.0	0.0	0.3	0.7	1.4	0.7	0.7	3.7	1.6	2.5	2.7	0.8
3.7	0.2	0.9	1.7	0.1	0.1	2.9	0.0	5.7	0.3	3.7	2.4
0.6	2.7	1.1	1.8	2.1	0.6	2.4	0.0	1.1	0.9	0.1	0.4
0.0	4.4	4.3	0.2	2.8	1.8	0.0	0.0	0.0	0.0	0.0	0.0
1.3	0.0	0.8	0.6	0.1	0.2	0.0	0.0	3.3	5.2	9.3	4.6
0.0	0.0	0.0	0.0	0.0	0.0	42.3	2.0	6.4	11.6	1.0	0.6
0.0	0.0	0.2	0.5	0.9	0.4	4.1	22.4	9.7	15.0	16.0	5.0
2.5	0.1	0.6	1.1	0.1	0.0	17.7	0.0	33.9	1.9	22.2	14.4
0.4	1.8	0.7	1.2	1.4	0.4	14.4	0.3	6.4	5.5	0.7	2.6
0.0	2.9	2.9	0.1	1.8	1.2	0.0	0.0	0.0	0.0	0.0	0.0
0.9	0.0	0.5	0.4	0.0	0.1	0.0	0.0	0.8	1.7	2.3	1.5
0.0	0.0	0.0	0.0	0.0	0.0	13.6	0.8	1.7	2.9	0.3	0.2
0.0	0.0	1.4	3.0	5.4	2.7	1.0	7.3	3.8	3.9	4.8	2.0
14.9	0.9	3.7	6.7	0.6	0.3	5.8	11.3	0.0	0.6	6.8	4.4
2.3	10.6	4.3	7.1	8.3	2.3	4.0	0.1	2.1	1.8	0.2	1.1
0.0	17.7	17.3	0.7	11.1	7.2	0.0	0.0	0.0	0.0	0.0	0.0
5.3	0.1	3.0	2.6	0.3	0.9	0.0	0.0	0.8	1.7	2.3	1.5
0.0	0.0	0.0	0.0	0.0	0.0	13.6	0.8	1.7	2.9	0.3	0.2
0.0	0.0	1.1	1.7	3.1	1.5	1.0	7.3	3.8	3.9	4.8	2.0
14.1	0.7	2.1	3.9	0.3	0.2	5.8	11.3	0.0	0.6	6.8	4.4
1.4	7.5	3.2	5.0	5.3	1.7	4.0	0.1	2.1	1.8	0.2	1.1
5.9	0.0	11.3	0.6	7.4	4.8	0.0	0.0	0.0	0.0	0.0	0.0
4.8	0.1	2.1	1.8	0.2	0.9	0.0	0.0	0.4	0.9	1.2	0.7
0.0	0.0	0.0	0.0	0.0	0.0	6.8	0.4	0.8	1.4	0.2	0.1
0.0	0.0	1.1	1.7	3.1	1.5	0.5	3.7	1.9	2.0	2.4	1.0
14.1	0.7	2.1	3.9	0.3	0.2	2.9	5.7	0.0	0.3	3.4	2.2
1.4	7.5	3.2	5.0	5.3	1.7	2.0	0.0	1.0	0.9	0.1	0.5

0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.6	1.3	1.7	1.1
10.2	0.6	1.2	2.2	0.2	0.2
0.7	5.5	2.8	3.0	3.6	1.5
4.3	8.5	0.0	0.4	5.1	3.3
3.0	0.0	1.6	1.4	0.2	0.8
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.8	1.7	2.3	1.5
13.6	0.8	1.7	2.9	0.3	0.2
1.0	7.3	3.8	3.9	4.8	2.0
5.8	11.3	0.0	0.6	6.8	4.4
4.0	0.1	2.1	1.8	0.2	1.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.6	1.3	1.7	1.1
10.2	0.6	1.2	2.2	0.2	0.2
0.7	5.5	2.8	3.0	3.6	1.5
4.3	8.5	0.0	0.4	5.1	3.3
3.0	0.0	1.6	1.4	0.2	0.8
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.4	0.9	1.2	0.7
6.8	0.4	0.8	1.4	0.2	0.1
0.5	3.7	1.9	2.0	2.4	1.0
2.9	5.7	0.0	0.3	3.4	2.2
2.0	0.0	1.0	0.9	0.1	0.5
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	2.4	5.2	7.0	4.4
40.7	2.3	5.0	8.6	1.0	0.7
3.0	22.0	11.3	11.8	14.5	5.9
17.3	33.9	0.0	1.7	20.4	13.3
12.0	0.2	6.2	5.5	0.7	3.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.1	0.1
0.7	0.1	0.1	0.1	0.0	0.0
0.1	0.4	0.2	0.2	0.3	0.1
0.2	0.7	0.5	0.0	0.4	0.2
0.2	0.0	0.1	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.1	0.1
0.7	0.1	0.1	0.1	0.0	0.0
0.1	0.4	0.2	0.2	0.3	0.1
0.2	0.7	0.5	0.0	0.4	0.2
0.2	0.0	0.1	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.0
0.3	0.0	0.1	0.1	0.0	0.0
0.0	0.2	0.1	0.1	0.1	0.0
0.1	0.3	0.3	0.0	0.2	0.1
0.1	0.0	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.1	0.1
0.5	0.0	0.1	0.1	0.0	0.0
0.0	0.3	0.2	0.1	0.2	0.1
0.2	0.5	0.4	0.0	0.3	0.2
0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.1	0.1

0.7	0.1	0.1	0.1	0.0	0.0
0.1	0.4	0.2	0.2	0.3	0.1
0.2	0.7	0.5	0.0	0.4	0.2
0.2	0.0	0.1	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.1	0.1
0.5	0.0	0.1	0.1	0.0	0.0
0.0	0.3	0.2	0.1	0.2	0.1
0.2	0.5	0.4	0.0	0.3	0.2
0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.0
0.3	0.0	0.1	0.1	0.0	0.0
0.0	0.2	0.1	0.1	0.1	0.0
0.1	0.3	0.3	0.0	0.2	0.1
0.1	0.0	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.3	0.3
2.0	0.2	0.3	0.4	0.0	0.0
0.2	1.1	0.6	0.5	0.9	0.3
0.7	2.0	1.6	0.0	1.2	0.7
0.6	0.0	0.3	0.3	0.0	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	0.9	1.8	0.9
6.3	0.4	1.3	2.1	0.3	0.1
0.6	3.8	2.0	2.5	3.7	1.0
3.7	7.4	6.8	0.4	0.0	2.9
3.0	0.0	1.2	1.0	0.2	0.6
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	0.9	1.8	0.9
6.3	0.4	1.3	2.1	0.3	0.1
0.6	3.8	2.0	2.5	3.7	1.0
3.7	7.4	6.8	0.4	0.0	2.9
3.0	0.0	1.2	1.0	0.2	0.6
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.5	0.9	0.4
3.1	0.2	0.6	1.1	0.1	0.1
0.3	1.9	1.0	1.3	1.8	0.5
1.8	3.7	3.4	0.2	0.0	1.5
1.5	0.0	0.6	0.5	0.1	0.3
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.4	0.7	1.3	0.7
4.7	0.3	1.0	1.6	0.2	0.1
0.5	2.8	1.5	1.9	2.8	0.8
2.8	5.5	5.1	0.3	0.0	2.2
2.3	0.0	0.9	0.8	0.1	0.5
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.5	0.9	1.8	0.9
6.3	0.4	1.3	2.1	0.3	0.1
0.6	3.8	2.0	2.5	3.7	1.0
3.7	7.4	6.8	0.4	0.0	2.9
3.0	0.0	1.2	1.0	0.2	0.6
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.4	0.7	1.3	0.7
4.7	0.3	1.0	1.6	0.2	0.1
0.5	2.8	1.5	1.9	2.8	0.8

2.8	5.5	5.1	0.3	0.0	2.2	0.0	0.0	0.0	0.0	0.0	0.0
2.3	0.0	0.9	0.8	0.1	0.5	0.0	0.0	0.9	1.5	2.7	1.7
0.0	0.0	0.0	0.0	0.0	0.0	12.1	0.9	2.0	4.2	0.4	0.2
0.0	0.0	0.2	0.5	0.9	0.4	1.2	7.3	3.8	4.9	7.1	2.0
3.1	0.2	0.6	1.1	0.1	0.1	7.2	14.4	13.2	0.7	8.8	0.0
0.3	1.9	1.0	1.3	1.8	0.5	5.1	0.1	2.3	2.0	0.3	1.2
1.8	3.7	3.4	0.2	0.0	1.5	0.0	0.0	0.0	0.0	0.0	0.0
1.5	0.0	0.6	0.5	0.1	0.3	0.0	0.0	0.3	0.5	0.9	0.4
0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.3	0.6	1.4	0.1	0.1
0.0	0.0	1.4	2.8	5.3	2.7	0.3	2.3	1.5	1.2	2.2	0.8
18.8	1.1	3.8	6.4	0.8	0.3	1.8	4.8	4.0	0.2	3.0	1.7
1.9	11.3	5.9	7.5	11.0	3.0	0.0	0.0	0.8	0.6	0.1	0.4
11.1	22.2	20.4	1.2	0.0	8.8	0.0	0.0	0.0	0.0	0.0	0.0
9.1	0.1	3.7	3.0	0.5	1.8	0.0	0.0	0.3	0.5	0.9	0.4
0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.3	0.6	1.4	0.1	0.1
0.0	0.0	0.3	0.5	0.9	0.6	0.3	2.3	1.5	1.2	2.2	0.8
4.0	0.3	0.7	1.4	0.1	0.1	1.8	4.8	4.0	0.2	3.0	1.7
0.4	2.4	1.3	1.6	2.4	0.7	0.0	0.0	0.8	0.6	0.1	0.4
2.4	4.8	4.4	0.2	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.7	0.0	0.8	0.7	0.1	0.4	0.0	0.0	0.2	0.2	0.4	0.2
0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.1	0.3	0.7	0.1	0.0
0.0	0.0	0.3	0.5	0.9	0.6	0.2	1.1	0.7	0.6	1.1	0.4
4.0	0.3	0.7	1.4	0.1	0.1	0.9	2.4	2.0	0.1	1.5	0.9
0.4	2.4	1.3	1.6	2.4	0.7	0.0	0.0	0.4	0.3	0.0	0.2
2.4	4.8	4.4	0.2	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.7	0.0	0.8	0.7	0.1	0.4	0.0	0.0	0.2	0.4	0.6	0.3
0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.2	0.4	1.0	0.1	0.1
0.0	0.0	0.2	0.2	0.4	0.3	0.3	1.7	1.1	0.9	1.7	0.6
2.0	0.2	0.3	0.7	0.1	0.0	1.3	3.6	3.0	0.2	2.3	1.3
0.2	1.2	0.6	0.8	1.2	0.3	0.0	0.0	0.6	0.5	0.1	0.3
1.2	2.4	2.2	0.1	1.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.8	0.0	0.4	0.3	0.0	0.2	0.0	0.0	0.3	0.5	0.9	0.4
0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.3	0.6	1.4	0.1	0.1
0.0	0.0	0.2	0.4	0.7	0.4	0.3	2.3	1.5	1.2	2.2	0.8
3.0	0.2	0.5	1.1	0.1	0.1	1.8	4.8	4.0	0.2	3.0	1.7
0.3	1.8	0.9	1.2	1.8	0.5	0.0	0.0	0.8	0.6	0.1	0.4
1.8	3.6	3.3	0.2	2.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.3	0.0	0.6	0.5	0.1	0.3	0.0	0.0	0.2	0.4	0.6	0.3
0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.2	0.4	1.0	0.1	0.1
0.0	0.0	0.3	0.5	0.9	0.6	0.3	1.7	1.1	0.9	1.7	0.6
4.0	0.3	0.7	1.4	0.1	0.1	1.3	3.6	3.0	0.2	2.3	1.3
0.4	2.4	1.3	1.6	2.4	0.7	0.0	0.0	0.6	0.5	0.1	0.3
2.4	4.8	4.4	0.2	2.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.7	0.0	0.8	0.7	0.1	0.4	0.0	0.0	0.2	0.2	0.4	0.2
0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.1	0.3	0.7	0.1	0.0
0.0	0.0	0.2	0.4	0.7	0.4	0.2	1.1	0.7	0.6	1.1	0.4
3.0	0.2	0.5	1.1	0.1	0.1	0.9	2.4	2.0	0.1	1.5	0.9
0.3	1.8	0.9	1.2	1.8	0.5	0.0	0.0	0.4	0.3	0.0	0.2
1.8	3.6	3.3	0.2	2.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.3	0.0	0.6	0.5	0.1	0.3	0.0	0.0	0.9	1.4	2.6	1.3
0.0	0.0	0.0	0.0	0.0	0.0	12.1	0.9	1.7	4.1	0.4	0.3
0.0	0.0	0.2	0.2	0.4	0.3	1.0	6.8	4.4	3.6	6.6	2.3
2.0	0.2	0.3	0.7	0.1	0.0	5.3	14.4	12.1	0.7	9.0	5.1
0.2	1.2	0.6	0.8	1.2	0.3	0.0	0.1	2.4	1.9	0.2	1.3
1.2	2.4	2.2	0.1	1.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.8	0.0	0.4	0.3	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0

0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.1	0.1	0.0	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.1	0.1	0.0	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.1	0.1	0.0	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.1	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.0
0.2	0.0	0.0	0.1	0.0	0.0
0.0	0.1	0.0	0.1	0.1	0.0
0.1	0.3	0.2	0.0	0.2	0.1
0.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.3	0.4	0.3
1.7	0.1	0.4	0.6	0.1	0.0
0.2	1.0	0.5	0.7	1.0	0.3
1.0	2.1	2.1	0.1	1.2	0.8
0.8	0.0	0.0	0.3	0.0	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.3	0.4	0.3
1.7	0.1	0.4	0.6	0.1	0.0
0.2	1.0	0.5	0.7	1.0	0.3

1.0	2.1	2.1	0.1	1.2	0.8
0.8	0.0	0.0	0.3	0.0	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.2	0.1
0.9	0.1	0.2	0.3	0.0	0.0
0.1	0.5	0.3	0.3	0.5	0.1
0.5	1.1	1.0	0.1	0.6	0.4
0.4	0.0	0.0	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.3	0.2
1.3	0.1	0.3	0.5	0.1	0.0
0.1	0.8	0.4	0.5	0.7	0.2
0.8	1.6	1.6	0.1	0.9	0.6
0.6	0.0	0.0	0.2	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.3	0.4	0.3
1.7	0.1	0.4	0.6	0.1	0.0
0.2	1.0	0.5	0.7	1.0	0.3
1.0	2.1	2.1	0.1	1.2	0.8
0.8	0.0	0.0	0.3	0.0	0.2
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.3	0.2
1.3	0.1	0.3	0.5	0.1	0.0
0.1	0.8	0.4	0.5	0.7	0.2
0.8	1.6	1.6	0.1	0.9	0.6
0.6	0.0	0.0	0.2	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.2	0.1
0.9	0.1	0.2	0.3	0.0	0.0
0.1	0.5	0.3	0.3	0.5	0.1
0.5	1.1	1.0	0.1	0.6	0.4
0.4	0.0	0.0	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.4	0.9	1.2	0.8
5.2	0.3	1.1	1.9	0.3	0.1
0.5	3.1	1.6	2.0	3.0	0.8
3.0	6.4	6.3	0.3	3.7	2.4
2.4	0.0	0.0	0.8	0.1	0.6
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.3	0.1
1.7	0.1	0.2	0.4	0.0	0.0
0.1	0.9	0.5	0.6	0.9	0.2
0.9	1.8	1.8	0.1	1.0	0.7
0.6	0.0	0.3	0.0	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.3	0.1
1.7	0.1	0.2	0.4	0.0	0.0
0.1	0.9	0.5	0.6	0.9	0.2
0.9	1.8	1.8	0.1	1.0	0.7
0.6	0.0	0.3	0.0	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.1	0.1
0.9	0.1	0.1	0.2	0.0	0.0
0.1	0.4	0.2	0.3	0.5	0.1
0.4	0.9	0.9	0.0	0.5	0.3
0.3	0.0	0.2	0.0	0.0	0.1

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
0.0	0.0	0.1	0.1	0.2	0.1	0.2	0.0	0.1	0.1	0.0	0.0
1.3	0.1	0.1	0.3	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1
0.1	0.7	0.3	0.5	0.7	0.2	0.1	0.3	0.2	0.0	0.2	0.1
0.7	1.4	1.4	0.1	0.7	0.5	0.1	0.0	0.0	0.0	0.0	0.0
0.5	0.0	0.2	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.3	0.1	0.2	0.0	0.0	0.1	0.0	0.0
1.7	0.1	0.2	0.4	0.0	0.0	0.0	0.1	0.0	0.1	0.1	0.0
0.1	0.9	0.5	0.6	0.9	0.2	0.1	0.2	0.2	0.0	0.1	0.1
0.9	1.8	1.8	0.1	1.0	0.7	0.1	0.0	0.0	0.0	0.0	0.0
0.6	0.0	0.3	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.1	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.0
1.3	0.1	0.1	0.3	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.0
0.1	0.7	0.3	0.5	0.7	0.2	0.0	0.1	0.1	0.0	0.1	0.1
0.7	1.4	1.4	0.1	0.7	0.5	0.0	0.0	0.0	0.0	0.0	0.0
0.5	0.0	0.2	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.2	0.1
0.0	0.0	0.0	0.1	0.1	0.1	0.6	0.0	0.2	0.2	0.0	0.0
0.9	0.1	0.1	0.2	0.0	0.0	0.1	0.3	0.2	0.2	0.3	0.2
0.1	0.4	0.2	0.3	0.5	0.1	0.2	0.8	0.7	0.0	0.5	0.3
0.4	0.9	0.9	0.0	0.5	0.3	0.3	0.0	0.1	0.1	0.0	0.0
0.3	0.0	0.2	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0
0.0	0.0	0.3	0.4	0.8	0.4	0.9	0.1	0.0	0.2	0.0	0.0
						0.0	0.3	0.3	0.3	0.4	0.1
5.1	0.3	0.6	1.3	0.1	0.1	0.3	0.9	1.1	0.0	0.6	0.4
0.3	2.7	1.4	1.9	2.7	0.7	0.4	0.0	0.2	0.1	0.0	0.0
2.6	5.5	5.4	0.3	3.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0
1.8	0.0	0.9	0.0	0.0	0.4	0.0	0.0	0.0	0.1	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.1	0.0	0.2	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.3	0.3	0.3	0.4	0.1
0.2	0.0	0.1	0.1	0.0	0.0	0.3	0.9	1.1	0.0	0.6	0.4
0.0	0.1	0.1	0.1	0.1	0.1	0.4	0.0	0.2	0.1	0.0	0.0
0.1	0.3	0.2	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0
0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.2	0.1	0.1	0.2	0.1
0.2	0.0	0.1	0.1	0.0	0.0	0.1	0.4	0.5	0.0	0.3	0.2
0.0	0.1	0.1	0.1	0.1	0.1	0.2	0.0	0.1	0.1	0.0	0.0
0.1	0.3	0.2	0.0	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.0
0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.1	0.0	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.2	0.2	0.3	0.1
0.1	0.0	0.0	0.0	0.0	0.0	0.2	0.6	0.8	0.0	0.5	0.3
0.0	0.1	0.0	0.0	0.1	0.0	0.3	0.0	0.1	0.1	0.0	0.0
0.0	0.1	0.1	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.1	0.0	0.2	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.3	0.3	0.4	0.1
0.2	0.0	0.0	0.1	0.0	0.0	0.3	0.9	1.1	0.0	0.6	0.4
0.0	0.1	0.0	0.1	0.1	0.0	0.4	0.0	0.2	0.1	0.0	0.0
0.1	0.2	0.2	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0
0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.1	0.0	0.1	0.0	0.0

0.0	0.3	0.2	0.2	0.3	0.1
0.2	0.6	0.8	0.0	0.5	0.3
0.3	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.4	0.0	0.0	0.1	0.0	0.0
0.0	0.2	0.1	0.1	0.2	0.1
0.1	0.4	0.5	0.0	0.3	0.2
0.2	0.0	0.1	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.2	0.3	0.1
2.6	0.2	0.1	0.5	0.0	0.0
0.1	1.0	0.8	0.8	1.3	0.4
0.8	2.6	3.2	0.1	1.8	1.2
1.2	0.0	0.6	0.3	0.0	0.1

VITA

R. Jayakrishnan was born at Chengannur, Kerala, India on May 23, 1963, the son of K. Radhakrishnan Nair and A. Chinmayi Devi. After completing his work at the Government Model Boys' High School, Haripad, Kerala, in 1978, he entered the St. Berchmans' College, Changanacherry, Kerala. After completing his pre-degree education there in 1980, he joined the Indian Institute of Technology, Madras, India. He received the Bachelor of Technology in Civil Engineering from there in May 1985. He entered the University of Texas at Austin in September, 1985 and received his Master of Science in Engineering (Civil) in December 1987. He continued his studies at the University of Texas towards a Doctorate. In January 1991 he started teaching as an Assistant Professor at the University of California at Irvine.

Permanent address: Madathinakathu Veedu
Chunkappara (P.O)
Pathanamthitta District
Kerala, India, 686547.

This dissertation was typed by the author.