Information Report FF-X-51

January 1975

DYNAMIC PROGRAMMING SUBROUTINES BASED ON THE DIJKSTRA ALGORITHM FOR FINDING MINIMUM COST PATHS IN DIRECTED NETWORKS

by

Gary H. Elsner, Michael R. Travis
and Peter H. Kourtz

F O R E S T   F I R E   R E S E A R C H   I N S T I T U T E

Canadian Forestry Service

Department of the Environment
Nicol Building
331 Cooper Street
Ottawa, Ontario
K1A 0H3

Authors

Gary H. Elsner and Michael R. Travis are with the Forest Recreation and Landscape Management Research Work Unit of the Pacific Southwest Forest and Range Experiment Station, Forest Service, U.S. Department of Agriculture in Berkeley, Calif. Peter H. Kourtz is with the Forest Fire Research Institute, Canadian Forestry Service, Department of the Environment, in Ottawa, Ontario.

INTRODUCTION

Dijkstra's (1959) tentative labeling algorithm has proved highly useful in solving a variety of network problems. These have ranged from prediction of the "flow" of fire along arcs in a dense network to estimation of travel times between population centers.

O'Regan, et al. (1973) predicted the future perimeter location of a forest fire by forecasting the time it took a fire to travel between node pairs. Travel time was calculated from data on fuel type, fuel moisture content, slope along an arc, wind speed, and wind direction. In a large network, a fire could take many possible paths to each node. The minimum time needed for it to reach a particular node was computed by applying the Dijkstra algorithm.

Elsner (1971) estimated minimum travel distances from population centers to recreation areas in California. These travel times were then used as a major variable in predicting ski usage of large resorts in the State.

While working with the Dijkstra algorithm, we soon realized that it had many other possible applications. This report describes two computer subroutines based on the algorithm. The subroutines are useful in finding the shortest path between a selected node and other nodes, or in finding all shortest paths in a directed network. They are also useful in determining minimum cost paths in transportation networks and in other, broader decision-making problems.

The algorithm assigns temporary and permanent labels (values) to the network nodes. A temporary node value represents an upper bound on the minimum time (or distance or cost) to go from the starting node to a particular node; a permanent label is the minimum time to reach that node. An iterative process is used to relabel nodes from temporary to permanent, and the process stops when either the particular node or all nodes are labeled permanent. Thus, the algorithm can be used to find the minimum time path between two network nodes or the minimum time from a specific node to all others in the network.

The steps of the procedure are:

(a) Assign to the starting node a permanent label value of 0 and to all other nodes the temporary value of infinity.

(b) Reassign values of those temporarily labeled nodes that are connected by arcs to the last permanently labeled node by adding to that permanent value the time associated with traveling along each arc.

(c) Scan all temporary node values and assign the one with the smallest value a permanent label. (In practice, only nodes adjacent to permanent nodes need to be scanned).

(d) Repeat steps b and c, until either the node of interest or all nodes have been permanently labeled.

We have not compared the computational efficiency of this algorithm with many others, but Dreyfus (1969) indicated that no more efficient algorithm existed at the time of his review. We did find the Dijkstra algorithm superior to both the Cascade algorithm (Farby, et al. 1967) and a "bends in the path" dynamic programming algorithm that was custom designed for use in a fire model.

To demonstrate an application of these subroutines, we present an illustrative network and data matrix with costs for each arc in the network (Fig. 1). The flows are directed and in this example, the cost of going from node I to node J may not be equal to the cost of going from node J to node I (Fig. 2).

The first subroutine (DIJKST) and corresponding illustrative calling routine (Appendix I) are appropriate for the case in which only minimum costs from a starting node to all others are required and the corresponding paths are not required. The second subroutine (DIJKS2) and corresponding illustrative calling routine (Appendix II) are for the situation in which the definition of the minimum paths is also required. This second routine requires additional computer storage space equal to the total number of nodes in the network.

The computer output for each of these programs includes results from a starting node to all other nodes. This is the structure of the problem that the Dijkstra algorithm is designed to solve. But by simply treating each node as a starting node and employing the algorithm as many times as there are nodes, one can easily determine minimum costs or routes or both between all pairs of nodes in the network. (The solution to this type of problem is illustrated in Appendix I and II.) It should be noted, however, that Floyd's (1962) algorithm and Dantzig's (1966) algorithm for solving all possible pairs in a network are more efficient than Dijkstra's algorithm for this purpose (Minieka, 1974).

The interpretation of the minimum path costs is straightforward, i.e., in the example problem the minimum cost of going from node 3 to node 9 is 11 units.

The interpretation of the routing list is not so straightforward, but is quite attractive in terms of defining all the minimum routes in a small amount of space. The Ith element of the routing list contains the node number from which node I was reached on the minimum path from the chosen starting node to node I. For example, in the problem using DIJKS2, if we pick the starting node as node 1, then to get to node 10 by way of a minimum path we came (interpreting the routing list backwards) to node 10 from node 9, then to node 9 from node 8, to node 8 from node 4, and to node 4 from node 1. The corresponding link costs for this route add up to 17 as shown in the minimum path cost matrix.

A slight modification to the Dijkstra algorithm offers a possibility of improving its efficiency under special conditions. A bi-directional search 1/ procedure (Pohl 1971) can be used, if (a) both the starting and ending node are known, (b) the minimum path is significantly less costly than others, and (c) the costs associated with each arc are independent of the path to the adjacent nodes. That is, the Dijkstra algorithm is applied simultaneously at the start and end nodes. (The end node is treated as if it were the beginning.) The algorithm is applied to two distinct sets of permenent and temporary labels alternately. The process stops as soon as a node has two permanent label values. The minimum cost is the sum of these two permenent values and the route associated with this value can be found by decoding the network in both directions to the start and end nodes.

Another method for improving the efficiency of the algorithm is to incorporate special computer storage and searching techniques. O'Regan, et al, (1973) and Yen, (1972) describe special hashing, linking, and storage procedures that are useful for large networks. However, computer programs incorporating the bi-directional search method in the specialized storage and searching techniques are not sufficiently general to warrant inclusion in this report.

1/ In the fire perimeter problem, for example, the time for the fire to cross an arc depends partly upon the time that the fire reaches the arc of interest.

REFERENCES

1. Dantzig, G., 1966. All Shortest Routes in a Graph, Theories des Graphes, Proc. Internat. Symp., Rome. Dunod, Paris, p. 91-92.

2. Dijkstra, E.W., 1959. A note on Two Problems in Connexion with Graphs, Numerische Mathematik, Vol. 1, pgs. 269-271.

3. Dreyfus, S., 1969. Appraisal of Some Shortest-Path Algorithms, Journal of Operations Research Society of America, Vol. 17, pgs. 395-412.

4. Elsner, Gary H., 1971. A Regression Method for Estimating the Level of Use and Market Area of a Proposed Large Ski Resort, Journal of Leisure Research, Vol. 3, No. 3, Summer 1971, pgs. 160-167.

5. Farby, B.A., A.H. Land, and J.D. Murchland, 1967. The Cascade Algorithm for Finding All Shortest Distances in a Directed Graph, Management Science, Vol. 14, No. 1.

6. Floyd, R.W., 1962. Algorithm 97: Shortest Path, Communications of the Association for Computing Machinery, Vol. 5, p. 345.

7. Minieka, E., 1974. On Computing Sets of Shortest Paths in a Graph, Communications of the Association for Computing Machinery, Vol. 17, No. 6., pgs. 351-353.

8. O'Regan, W.G., S. Nozaki, and P.H. Kourtz, 1973. A Method for Using Directional Rates of Spread to Predict Forest Fire Configurations. WSS/CI Paper No. 73-17.

9. Pohl, Ira, 1971. Bi-Directional Search. Machine Intelligence, Vol. 6.

10. Yen, J.Y., 1972. Finding the Lengths of all Shortest Paths in N-Node Nonnegative Distances Complete Networks Using $1/2N^3$ Additions and N^3 Additions and N^3 Comparisons. Journal of the Association for Computing Machinery, Vol. 19, No. 3.

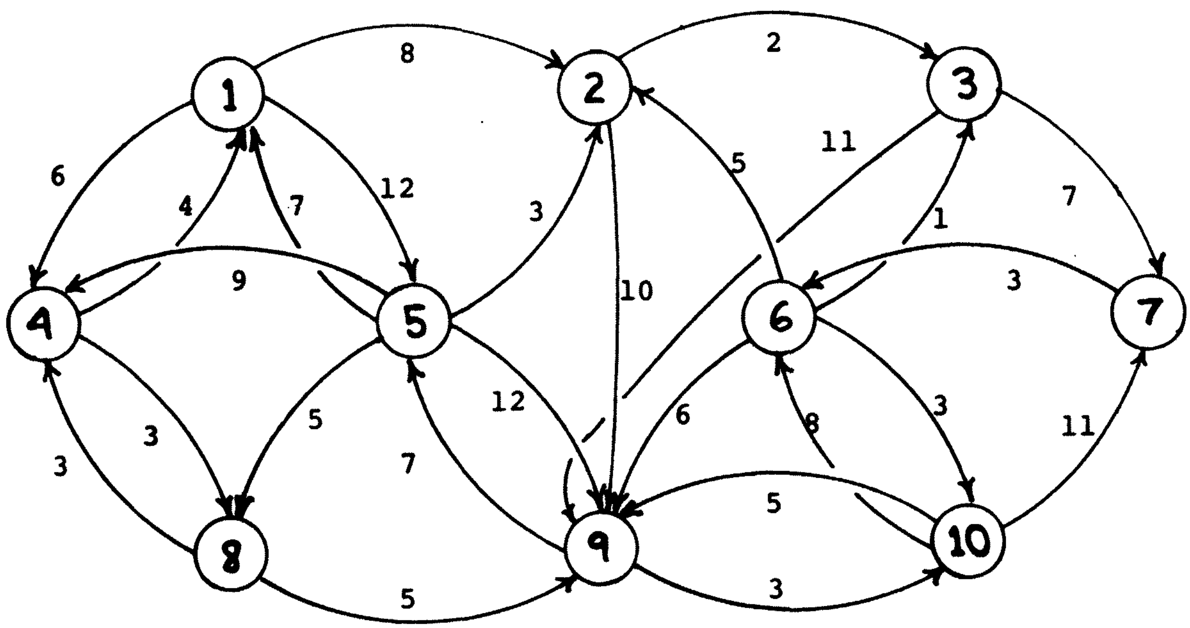

Fig. 1 Network for Illustrating Application of the DIJKSTRA Algorithm

TO NODE

FROM NODE	1	2	3	4	5	6	7	8	9	10
1	0	8	∞	6	12	∞	∞	∞	∞	∞
2	∞	0	2	∞	∞	∞	∞	∞	10	∞
3	∞	∞	0	∞	∞	∞	7	∞	11	∞
4	4	∞	∞	0	∞	∞	∞	3	∞	∞
5	7	3	∞	9	0	∞	∞	5	12	∞
6	∞	5	1	∞	∞	0	∞	∞	6	3
7	∞	∞	∞	∞	∞	3	0	∞	∞	∞
8	∞	∞	∞	3	∞	∞	∞	0	5	∞
9	∞	∞	∞	∞	7	∞	∞	∞	0	3
10	∞	∞	∞	∞	∞	8	11	∞	5	0

Fig. 2 Directed arc "travel" costs for the illustrative network.

Appendix I

DIJKST Subroutine and Illustration for Finding Minimum Cost Routes in a Directed Network

```
      SUBROUTINE DIJKST(COST,T,FLAG,NSIZE,N,ISTART)
C  PROGRAM DIJKST - MINIMUM PATH SOLUTION BY DIJKSTRA
C  TENTATIVE LABEL ALGORITHM
C  SEE:
C  DIJKSTRA, E.W., A NOTE ON TWO PROBLEMS IN CONNEXION WITH
C  GRAPHS.
C  NUMERISCHE MATHEMATIK 1:269-271 (1959)
C
C  PROGRAMMED BY MIKE TRAVIS, JUNE 1970
      INTEGER COST(NSIZE,1),T(1)
C  (A NEGATIVE VALUE MEANS INFINITE COST)
      LOGICAL FLAG(N),FOUND
C  COST(I,J) = COST OF GOING FROM NODE I TO NODE J
C  T(I) = TOTAL COST OF GOING FROM STARTING POINT TO NODE I
C  FLAG(I) = .FALSE. IF T(I) IS TENTATIVE, .TRUE. IF PERMANENT
      DATA LARGE/999999/
C  INITIALIZE TOTAL COSTS AND FLAGS
      DO 100 I=1,N
      T(I)=LARGE
  100 FLAG(I)=.FALSE.
C  SET STARTING NODE LABEL
      T(ISTART)=0
C  BEGIN - FIND SMALLEST TEMP. LABEL
    1 ISMALL=LARGE
      FOUND=.FALSE.
      DO 200 I=1,N
      IF(FLAG(I).OR.T(I).GE.ISMALL)GOTO 200
      ISMALL=T(I)
      IS=I
      FOUND=.TRUE.
  200 CONTINUE
C  DONE IF ALL LABELS ARE PERMANENT
      IF(.NOT.FOUND)GOTO 2
C  MAKE PERMANENT THE SMALLEST TEMP LABEL
      FLAG(IS)=.TRUE.
C  UPDATE LABELS ON NONPERMANENT CONNECTED NODES
      DO 300 I=1,N
      IF(FLAG(I).OR.COST(IS,I).LT.0)GOTO 300
      IT=T(IS)+COST(IS,I)
      IF(IT.LT.T(I))T(I)=IT
  300 CONTINUE
C  REPEAT
      GOTO 1
C  COMPLETED
    2 CONTINUE
      RETURN
      END
```

```
C   DEMONSTRATE DIJKSTRA ALGORITHM MINIMUM PATH SOLUTION.
C   MIKE TRAVIS      US FOREST SERVICE    BERKELEY
C   ====================================================================
C   NET(I,J) IS THE COST OF THE PATH FROM NODE I TO NODE J.
C   MIN(J) RETURNS THE MINIMUM COST TO NODE J FROM THE DESIGNATED
C   STARTING NODE.
C   ARRAY IF HOLDS FLAGS DURING ITERATION.
      DIMENSION NET(10,10),MIN(10)
      DIMENSION IF(10)
C   ====================================================================
C   READ IN THE NETWORK OF PATHS
C   NOTE THAT -
C       A NEGATIVE COST IS TAKEN TO MEAN INFINITE (NO CONNECTION)
C       NET(I,J) NEED NOT EQUAL NET(J,I)
      DO 1 I=1,9,2
    1 READ 10,(NET(I,J),J=1,10),(NET(I+1,J),J=1,10)
   10 FORMAT(20I4)
C   PRINT OUT THE NETWORK
      PRINT 4
    4 FORMAT(1H1,'COST ARRAY'//)
      DO 2 I=1,10
    2 PRINT 5,(NET(I,J),J=1,10)
    5 FORMAT(1X,10I6)
C   ====================================================================
C   CHOOSE POINT 3 AS A STARTING POINT
      ISTART=3
C   PERFORM MINIMUM PATH ANALYSIS
C   ---
      CALL DIJKST(NET,MIN,IF,10,10,ISTART)
C   ---
C   PRINT MINIMA
      PRINT 6
    6 FORMAT(1H1//2X,'NODE',7X,'MINIMUM PATH COSTS')
      PRINT 65
   65 FORMAT(1X,9(6H------))
      PRINT 66,(K,K=1,10)
   66 FORMAT(10X,10I4)
      PRINT 65
      PRINT 11,ISTART,MIN
   11 FORMAT(1X,I4,5X,10I4)
C   ====================================================================
C   WE CAN DO ALL NODES BY ITERATING OVER THE STARTING POINT
      PRINT 6
      PRINT 65
      PRINT 66,(K,K=1,10)
      PRINT 65
      DO 100 IS=1,10
C   ---
      CALL DIJKST(NET,MIN,IF,10,10,IS)
C   ---
      PRINT 11,IS,MIN
  100 CONTINUE
C   ====================================================================
C   END OF DEMONSTRATION
      STOP
      END
```

COST ARRAY

8	-1	6	12	-1	-1	-1	-1	-1
0	2	-1	-1	-1	-1	-1	10	-1
-1	0	-1	-1	-1	7	-1	11	-1
-1	-1	0	-1	-1	-1	3	-1	-1
3	-1	9	0	-1	-1	5	12	-1
5	1	-1	-1	0	-1	-1	6	3
-1	-1	-1	-1	3	0	-1	-1	-1
-1	-1	3	-1	-1	-1	0	5	-1
-1	-1	-1	7	-1	-1	-1	0	3
-1	-1	-1	-1	8	11	-1	5	0

NODE	MINIMUM PATH COSTS									
	1	2	3	4	5	6	7	8	9	10
3	25	15	0	26	18	10	7	23	11	13

NODE MINIMUM PATH COSTS

	1	2	3	4	5	6	7	8	9	10
1	0	8	10	6	12	20	17	9	14	17
2	24	0	2	25	17	12	9	22	10	13
3	25	15	0	26	18	10	7	23	11	13
4	4	12	14	0	15	19	21	3	8	11
5	7	3	5	8	0	15	12	5	10	13
6	20	5	1	21	13	0	8	18	6	3
7	23	8	4	24	16	3	0	21	9	6
8	7	15	17	3	12	16	19	0	5	8
9	14	10	12	15	7	11	14	12	0	3
10	19	13	9	20	12	8	11	17	5	0

Appendix II

DIJKS2 Subroutine and Illustration for Finding Minimum Costs and Route Definitions in a Directed Newtwork

```
      SUBROUTINE DIJKS2(COST,T,IR,FLAG,NSIZE,N,ISTART)
C     PROGRAM DIJKST - MINIMUM PATH SOLUTION BY DIJKSTRA
C     TENTATIVE LABEL ALGORITHM
C     VERSION 2 - GENERATES THE ASSOCIATED ROUTING VECTOR.
C     SEE:
C     DIJKSTRA, E.W., A NOTE ON TWO PROBLEMS IN CONNEXION WITH
C     GRAPHS.
C     NUMERISCHE MATHEMATIK 1:269-271 (1959)
C
C     PROGRAMMED BY MIKE TRAVIS, JUNE 1970
      INTEGER COST(NSIZE,1),T(1)
C     (A NEGATIVE VALUE MEANS INFINITE COST)
      LOGICAL FLAG(N),FOUND
C     COST(I,J) = COST OF GOING FROM NODE I TO NODE J
C     T(I) = TOTAL COST OF GOING FROM STARTING POINT TO NODE I
C     FLAG(I) = .FALSE. IF T(I) IS TENTATIVE, .TRUE. IF PERMANENT
      DIMENSION IR(N)
C     IR(I) CONTAINS THE INDEX OF THE NODE FROM WHICH NODE I WAS
C     REACHED ON THE MINIMUM PATH FROM THE CHOSEN STARTING NODE TO
C     NODE I.
C     SO IF IR(I) CONTAINS, SAY, J, CONSULT IR(J) TO FIND THE NEXT NODE INWARD
C     TOWARD THE START.  REPEAT UNTIL WE REACH THE STARTING NODE.
C     (IR(ISTART) IS ALWAYS 0 )
      DATA LARGE/999999/
C     INITIALIZE TOTAL COSTS AND FLAGS
      DO 100 I=1,N
      T(I)=LARGE
  100 FLAG(I)=.FALSE.
C     SET STARTING NODE LABEL
      T(ISTART)=0
      IR(ISTART)=0
C     BEGIN - FIND SMALLEST TEMP. LABEL
    1 ISMALL=LARGE
      FOUND=.FALSE.
      DO 200 I=1,N
      IF(FLAG(I).OR.T(I).GE.ISMALL)GOTO 200
      ISMALL=T(I)
      IS=I
      FOUND=.TRUE.
  200 CONTINUE
C     DONE IF ALL LABELS ARE PERMANENT
      IF(.NOT.FOUND)GOTO 2
C     MAKE PERMANENT THE SMALLEST TEMP LABEL
      FLAG(IS)=.TRUE.
C     UPDATE LABELS ON NONPERMANENT CONNECTED NODES
      DO 300 I=1,N
      IF(FLAG(I).OR.COST(IS,I).LT.0)GOTO 300
      IT=T(IS)+COST(IS,I)
      IF(IT.GE.T(I))GOTO 300
      T(I)=IT
      IR(I)=IS
  300 CONTINUE
C     REPEAT
      GOTO 1
C     COMPLETED
    2 CONTINUE

      RETURN
      END
```

```
C   DEMONSTRATE DIJKSTRA ALGORITHM SOLUTION TO MINIMUM PATH
C   COST AND ROUTING PROBLEM.
C   MIKE TRAVIS        US FOREST SERVICE      BERKELEY
C   ====================================================================
C   NET(I,J) IS THE COST OF THE PATH FROM NODE I TO NODE J.
C   MIN(J) RETURNS THE MINIMUM COST TO NODE J FROM THE DESIGNATED
C   STARTING NODE.
      DIMENSION NET(10,10),MIN(10)
C   ARRAY IF HOLDS FLAGS DURING ITERATION.
      DIMENSION IF(10)
C   IROUTE RETURNS THE ROUTING INFORMATION ASSOCIATED WITH A
C   MINIMUM PATH SOLUTION.
      DIMENSION IROUTE(10)
C   ====================================================================
C   READ IN THE NETWORK OF PATHS
C   NOTE THAT -
C       A NEGATIVE COST IS TAKEN TO MEAN INFINITE (NO CONNECTION)
C       NET(I,J) NEED NOT EQUAL NET(J,I)
      DO 1 I=1,9,2
    1 READ 10,(NET(I,J),J=1,10),(NET(I+1,J),J=1,10)
   10 FORMAT(20I4)
C   PRINT OUT THE NETWORK
      PRINT 4
    4 FORMAT(1H1,'COST ARRAY'//)
      DO 2 I=1,10
    2 PRINT 5,(NET(I,J),J=1,10)
    5 FORMAT(1X,10I6)
C   ====================================================================
C   CHOOSE POINT 3 AS A STARTING POINT
      ISTART=3
C   PERFORM MINIMUM PATH ANALYSIS
C   ---
      CALL DIJKS2(NET,MIN,IROUTE,IF,10,10,ISTART)
C   ---
C   PRINT MINIMUM COSTS AND OPTIMAL ROUTES
      PRINT 6
    6 FORMAT(1H1//2X,'NODE',7X,'MINIMUM PATH COSTS',T63,'ROUTING LIST')
      PRINT 65,(K,K=1,10),(K,K=1,10)
   65 FORMAT(1X,16(6H------),4H----,
     $/10X,10I4,T60,10I4,
     $/1X,16(6H------),4H----)
      PRINT 7,ISTART,MIN,IROUTE
    7 FORMAT(1X,I4,5X,10I4,T60,10I4)
C   ====================================================================
C   WE CAN DO ALL POSSIBLE INTERCONNECTIONS BY ITERATING OVER
C   THE STARTING NODE.
      PRINT 6
      PRINT 65,(K,K=1,10),(K,K=1,10)
      DO 100 IS=1,10
C   ---
      CALL DIJKS2(NET,MIN,IROUTE,IF,10,10,IS)
C   ---
      PRINT 7,IS,MIN,IROUTE
  100 CONTINUE
C   ====================================================================
C   END OF DEMONSTRATION

      STOP
      END
```

COST ARRAY

8	-1	6	12	-1	-1	-1	-1	-1
0	2	-1	-1	-1	-1	-1	10	-1
-1	0	-1	-1	-1	7	-1	11	-1
-1	-1	0	-1	-1	-1	3	-1	-1
3	-1	9	0	-1	-1	5	12	-1
5	1	-1	-1	0	-1	-1	6	3
-1	-1	-1	-1	3	0	-1	-1	-1
-1	-1	3	-1	-1	-1	0	5	-1
-1	-1	-1	7	-1	-1	-1	0	3
-1	-1	-1	-1	8	11	-1	5	0

NODE	MINIMUM PATH COSTS										ROUTING LIST									
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
3	25	15	0	26	18	10	7	23	11	13	5	6	0	8	9	7	3	5	3	6

NODE	MINIMUM PATH COSTS										ROUTING LIST									
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10
1	0	8	10	6	12	20	17	9	14	17	0	1	2	1	1	7	3	4	8	9
2	24	0	2	25	17	12	9	22	10	13	5	0	2	8	9	7	3	5	2	9
3	25	15	0	26	18	10	7	23	11	13	5	6	0	8	9	7	3	5	3	6
4	4	12	14	0	15	19	21	3	8	11	4	1	2	0	9	10	3	4	8	9
5	7	3	5	8	0	15	12	5	10	13	5	5	2	8	0	7	3	5	8	9
6	20	5	1	21	13	0	8	18	6	3	5	6	6	8	9	0	3	5	6	6
7	23	8	4	24	16	3	0	21	9	6	5	6	6	8	9	7	0	5	6	6
8	7	15	17	3	12	16	19	0	5	8	4	1	2	8	9	10	10	0	8	9
9	14	10	12	15	7	11	14	12	0	3	5	5	2	8	9	10	10	5	0	9
10	19	13	9	20	12	8	11	17	5	0	5	6	6	8	9	10	10	5	10	0