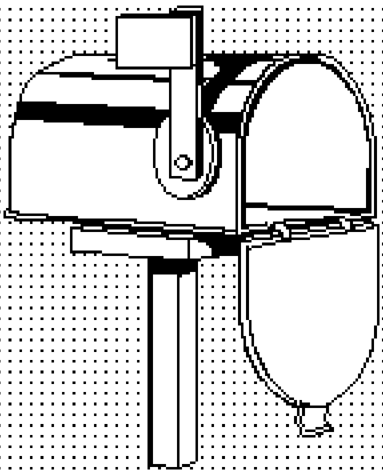
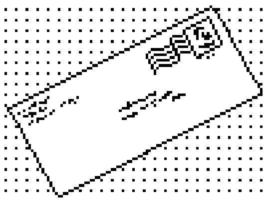
The Chinese Postman Problem





This Hypercard stack was prepared by:
Dennis Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242.
e-mail: dennis-bricker@uiowa.edu

author

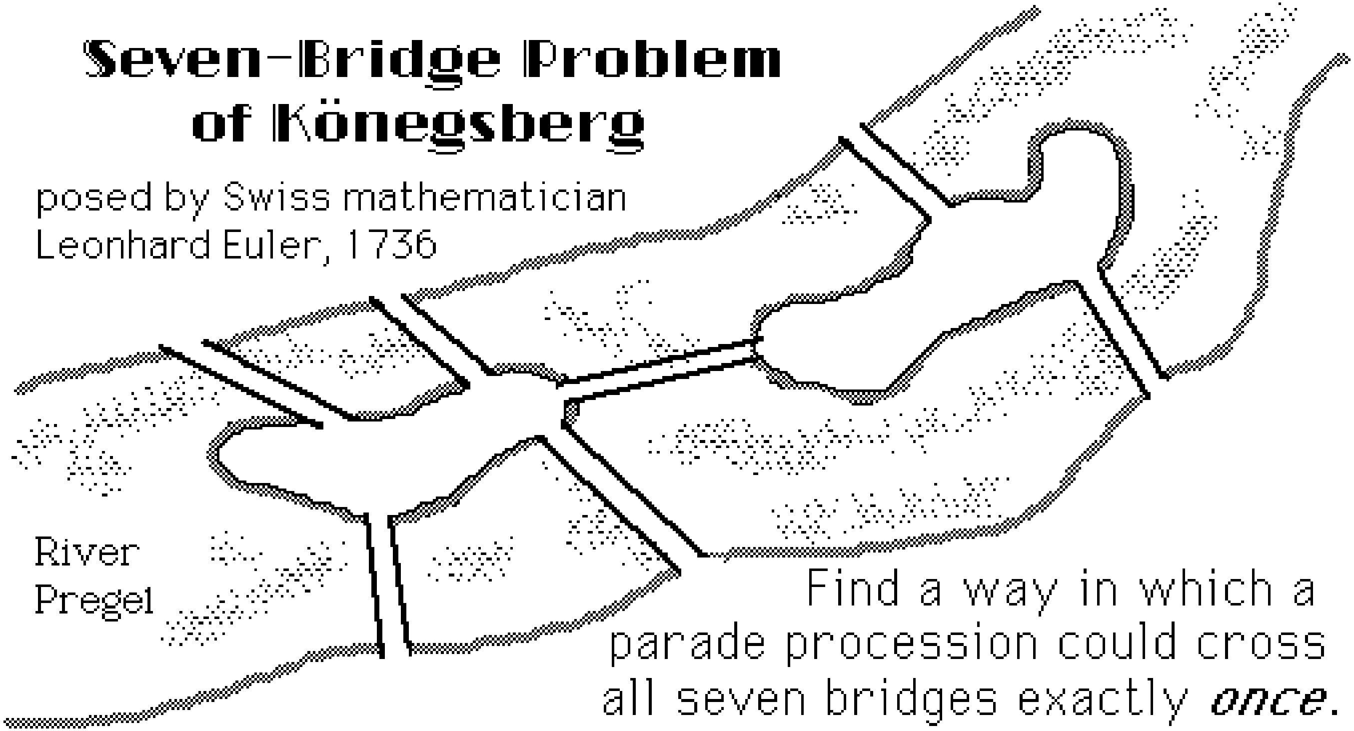
Contents

- ☞ Euler Paths & Tours
- ☞ Postman Problem in undirected network
- ☞ Postman Problem in directed network
- ☞ Summary & References

©D.Bricker, U. of Iowa, 1998

Seven-Bridge Problem of Könegsberg

posed by Swiss mathematician Leonhard Euler, 1736



Find a way in which a parade procession could cross all seven bridges exactly *once*.

©D.Bricker, U. of Iowa, 1998

Define:

Euler path: a path through a graph which traverses every edge of the graph ***exactly once***.

Euler tour: a circuit of a graph which traverses every edge of the graph ***exactly once***, i.e., an Euler path beginning and ending at the same node.

©D.Bricker, U. of Iowa, 1998

Define:

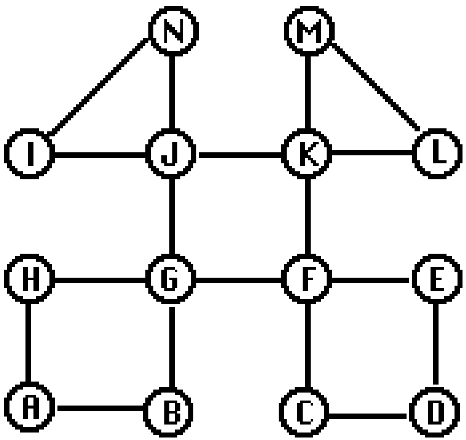
Degree of a node in an ***undirected*** graph is the number of incident edges of the node.

Indegree of a node in a ***directed*** graph is the number of edges ***into*** the node.

Outdegree of a node in a ***directed*** graph is the number of edges ***from*** the node.

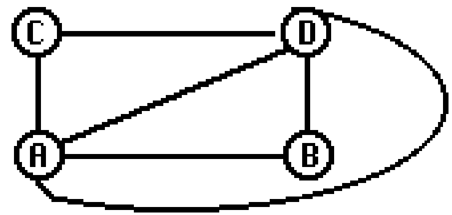
Polarity of a node of a directed graph is the difference: *indegree – outdegree*

©D.Bricker, U. of Iowa, 1998

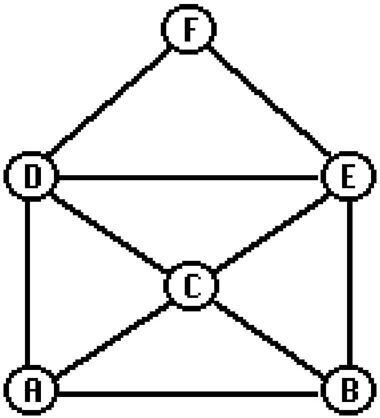


Do these graphs possess either Euler tours or Euler paths?

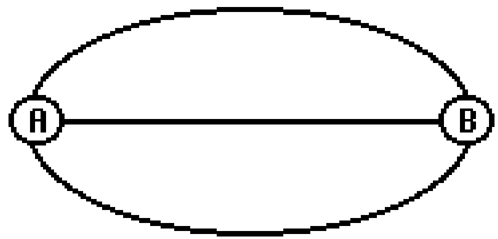
What is the degree of each node?



©D.Bricker, U. of Iowa, 1998



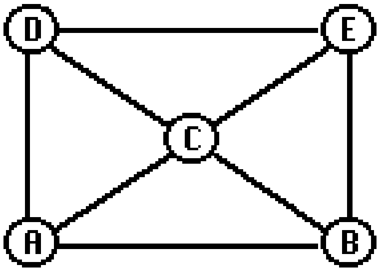
Do these graphs possess either Euler tours or Euler paths?



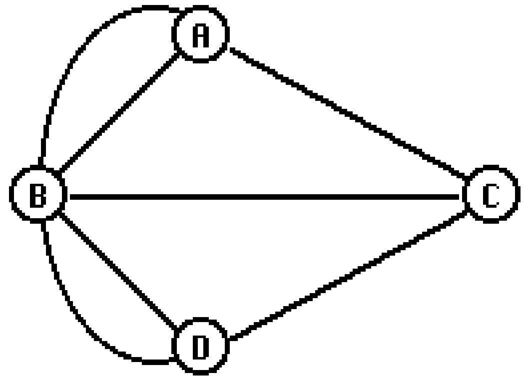
What is the degree of each node?

©D.Bricker, U. of Iowa, 1998

Do these graphs possess either Euler tours or Euler paths?



What is the degree of each node?



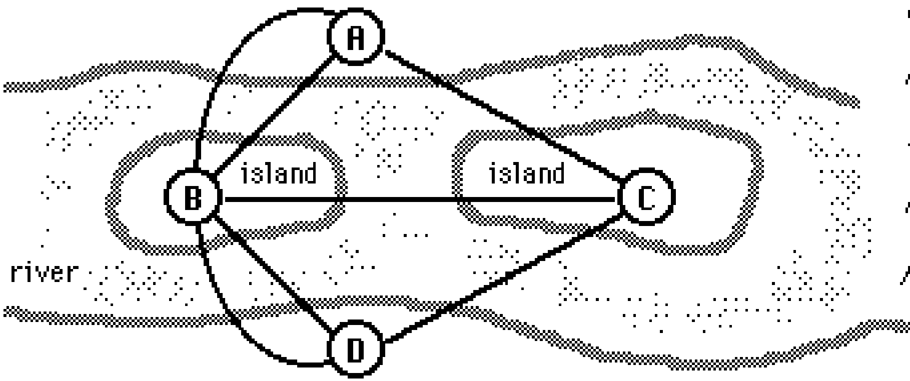
©D.Bricker, U. of Iowa, 1998

EULER'S THEOREM

- A connected *undirected* graph possesses
  - an Euler tour if & only if all nodes have even degree
  - an Euler path if & only if exactly two nodes have odd degree
- A connected *digraph* possesses
  - an Euler tour if & only if the polarity of each node is zero

©D.Bricker, U. of Iowa, 1998

7-Bridges Problem



There are 4 odd-degree nodes, and therefore neither an Euler path nor an Euler tour!

Nodes **B** & **C** represent the islands
Nodes **A** & **D** represent the two riverbanks
Edges represent bridges

©D.Bricker, U. of Iowa, 1998

Finding an Euler Tour

Begin at any node.

Traverse the edges, deleting each as it is traversed.

The choice of the edge from a node is arbitrary, except for the rule:

Never traverse an edge which is a bridge (an edge whose deletion would disconnect the graph).

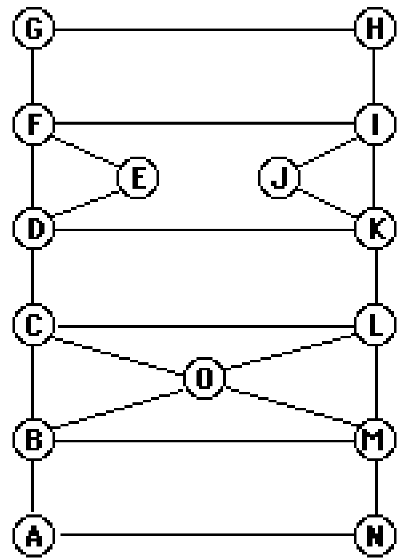
©D.Bricker, U. of Iowa, 1998

Example

Suppose that you hold a summer job as highway inspector.

You must periodically drive along the highways, checking on debris & the need for repairs.

If you live in town A, is it possible to find a round trip which takes you over each section of highway ***exactly once?***

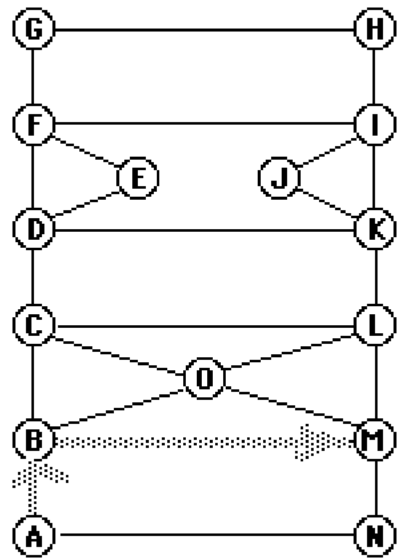


©D.Bricker, U. of Iowa, 1998

All nodes have even degree, so an Euler tour exists!

Suppose that we begin to construct an Euler tour by including edges AB and BM:

Then in choosing the next edge, we cannot choose edge MN, which has become a "bridge". Either MO or ML must be the next edge included in the tour!

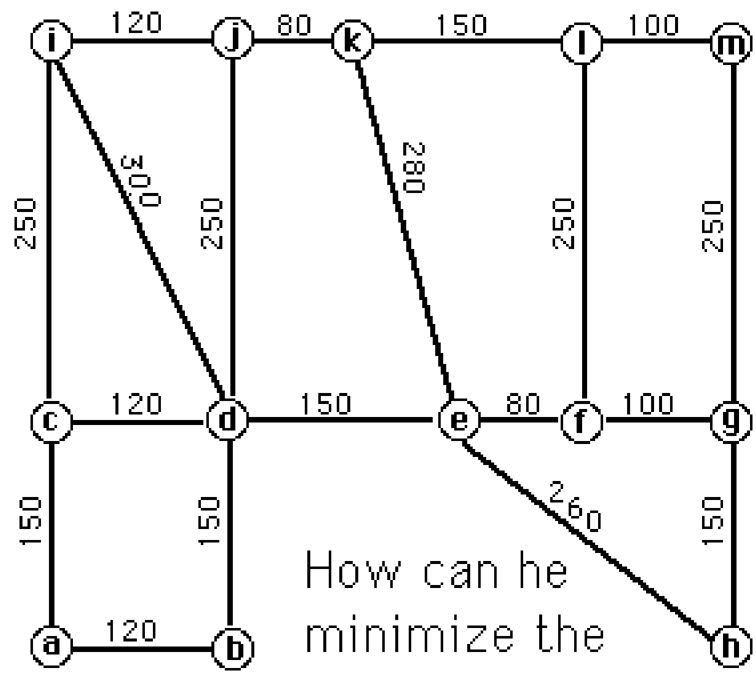


©D.Bricker, U. of Iowa, 1998

THE CHINESE POSTMAN PROBLEM

A mailman must deliver mail to residents on the streets shown.

He begins at node **a**, and must traverse each street at least once, and return to node **a**.



How can he minimize the total distance travelled?

©D.Bricker, U. of Iowa, 1998

Solving the postman problem:

If an Euler tour exists, it is the optimal route.

Otherwise,

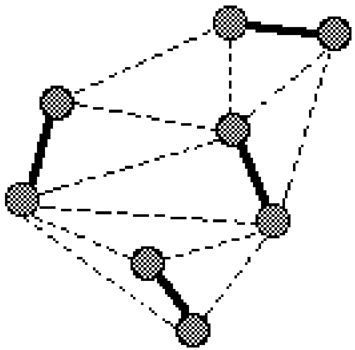
add "artificial" edges, parallel to the existing edges, which will turn all odd-degree nodes into even-degree nodes. *(There must be an even number of such odd-degree nodes.)*

The edges to be added are found by a **minimum length pairwise-matching** algorithm.

(In practice, this might be estimated by inspection, for a near-optimal solution.)

©D.Bricker, U. of Iowa, 1998

Matching Problem

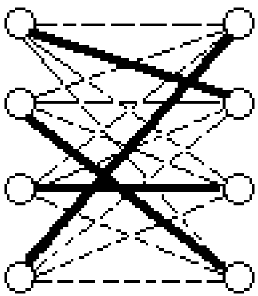


Given a set of nodes, assign (match) each node to exactly one other node, so that the sum of the matching costs are minimized, where
cost of matching i & j is C_{ij}

©D.Bricker, U. of Iowa, 1998

Matching Problem

If the graph is "bipartite", then this is the ordinary assignment problem, solvable by, for example, the "Hungarian Method".



In a bipartite network, the nodes may be partitioned into 2 sets, such that edge (i,j) exists only if nodes i & j are not contained in the same set.

©D.Bricker, U. of Iowa, 1998

Matching Problem

For the more general (non-bipartite) matching problem, there is an "efficient" (i.e., polynomial-time) algorithm by J. Edmonds, which, however, is rather complicated to implement.

©D.Bricker, U. of Iowa, 1998

Matching Problem

Formulation: Define

$$X_{ij} = \begin{cases} 1 \text{ if nodes } i \ \&\ j \text{ are matched} \\ 0 \text{ otherwise} \end{cases}$$

Minimize $\displaystyle\sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij} X_{ij}$

subject to

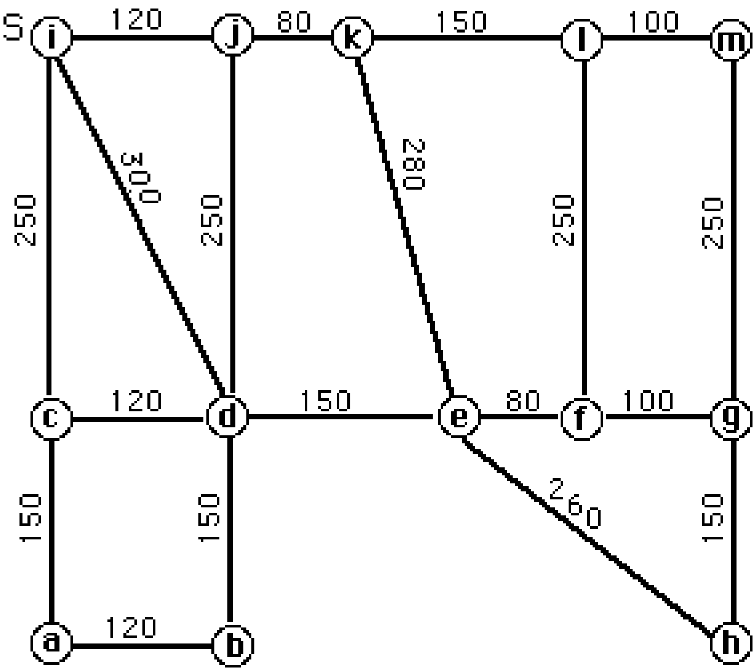
$$\sum_{j \neq i} X_{ij} = 1, \quad i=1,2, \ldots n$$

$$X_{ij} \in \{0,1\}$$

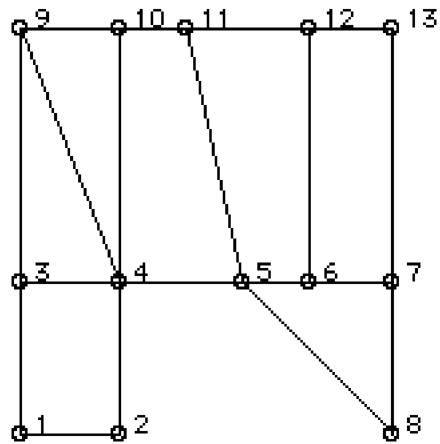
©D.Bricker, U. of Iowa, 1998

The odd-degree nodes are: c, d, f, g,
i, j, k, & l.

We need to compute the shortest path lengths between each pair of nodes from this set.



©D.Bricker, U. of Iowa, 1998



©D.Bricker, U. of Iowa, 1998

Using Floyd's Algorithm for finding shortest paths:

Path Lengths

from \ to	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	120	150	270	420	500	600	680	400	520	600	750	850
2	120	0	270	150	300	380	480	560	450	400	480	630	730
3	150	270	0	120	270	350	450	530	250	370	450	600	700
4	270	150	120	0	150	230	330	410	300	250	330	480	580
5	420	300	270	150	0	80	180	260	450	360	280	330	430
6	500	380	350	230	80	0	100	250	530	440	360	250	350
7	600	480	450	330	180	100	0	150	630	540	460	350	250
8	680	560	530	410	260	250	150	0	710	620	540	500	400
9	400	450	250	300	450	530	630	710	0	120	200	350	450
10	520	400	370	250	360	440	540	620	120	0	80	230	330
11	600	480	450	330	280	360	460	540	200	80	0	150	250
12	750	630	600	480	330	250	350	500	350	230	150	0	100
13	850	730	700	580	430	350	250	400	450	330	250	100	0

©D.Bricker, U. of Iowa, 1998

Predecessors

from \ to	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	1	1	2	4	5	6	5	3	4	10	6	7
2	2	0	1	2	4	5	6	5	4	4	10	6	7
3	3	1	0	3	4	5	6	5	3	4	10	6	7
4	2	4	4	0	4	5	6	5	4	4	10	6	7
5	2	4	4	5	0	5	6	5	4	11	5	6	7
6	2	4	4	5	6	0	6	7	4	11	5	6	7
7	2	4	4	5	6	7	0	7	4	11	5	6	7
8	2	4	4	5	8	7	8	0	4	11	5	6	7
9	3	4	9	9	4	5	6	5	0	9	10	11	12
10	2	4	4	10	11	5	6	5	10	0	10	11	12
11	2	4	4	10	11	5	6	5	10	11	0	11	12
12	2	4	4	5	6	12	6	7	10	11	12	0	12
13	2	4	4	5	6	7	13	7	10	11	12	13	0

©D.Bricker, U. of Iowa, 1998

Chinese Postman Problem in a Graph

Odd-degree nodes

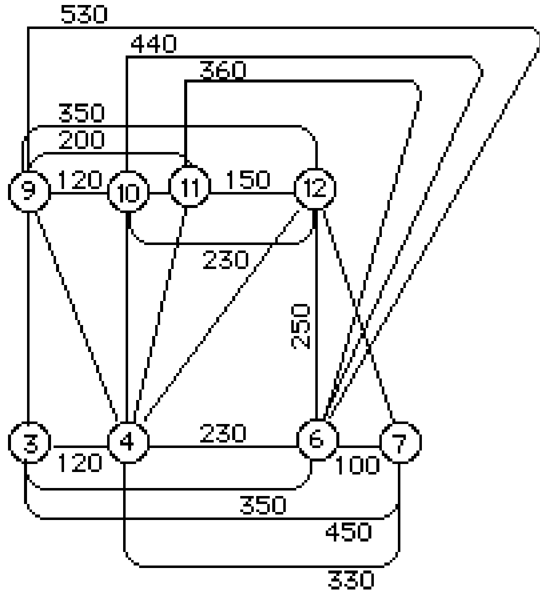
i:	3	4	6	7	9	10	11	12
Degree:	3	5	3	3	3	3	3	3

©D.Bricker, U. of Iowa, 1998

Shortest Paths

from \ to	3	4	6	7	9	10	11	12
3	0	120	350	450	250	370	450	600
4	120	0	230	330	300	250	330	480
6	350	230	0	100	530	440	360	250
7	450	330	100	0	630	540	460	350
9	250	300	530	630	0	120	200	350
10	370	250	440	540	120	0	80	230
11	450	330	360	460	200	80	0	150
12	600	480	250	350	350	230	150	0

©D.Bricker, U. of Iowa, 1998



We must find an optimal matching in a network with 8 nodes and edges between every pair (with length = the length of the shortest path in original network)

(not all edges are shown in the diagram!)

©D.Bricker, U. of Iowa, 1998

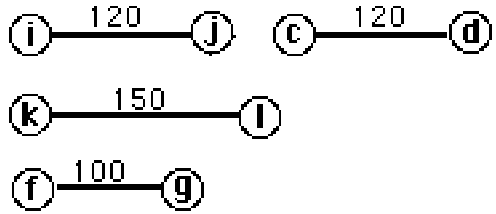
Matching

```
from:   3   6   9  11
  to:   4   7  10  12

length 120     path: 3 4
length 100     path: 6 7
length 120     path: 9 10
length 150     path: 11 12
```

Total length of paths added: 490

The minimum-length pairwise matching in this network is

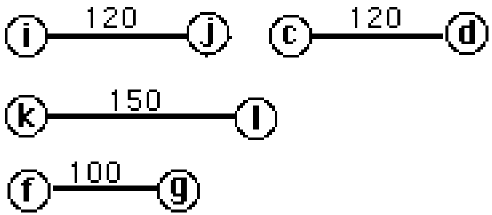


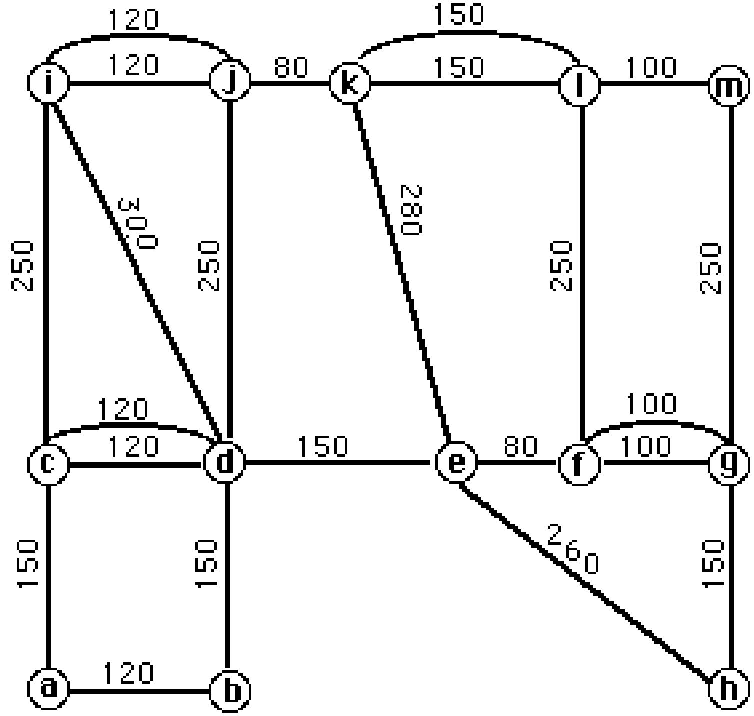
Total length: 490

(length to be traversed twice!)

©D.Bricker, U. of Iowa, 1998

Add paths to the network:



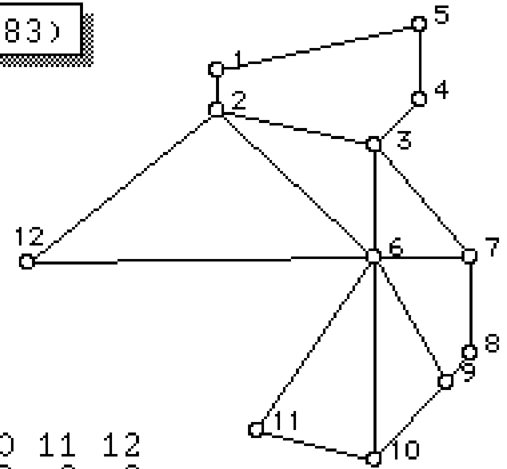


The result is a network with only even-degree nodes. We need only now to find an Euler tour!

©D.Bricker, U. of Iowa, 1998

Example

Random Network (seed= 136383)



Node Degrees

node #:	1	2	3	4	5	6	7	8	9	10	11	12
degree:	2	4	4	2	2	7	3	2	3	3	2	2

©D.Bricker, U. of Iowa, 1998

Shortest Path Lengths

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	8	37	49	37	48	65	83	76	88	88	53
2	8	0	29	41	45	40	57	75	68	80	80	45
3	37	29	0	12	27	22	28	47	50	62	62	74
4	49	41	12	0	15	34	40	59	62	74	74	86
5	37	45	27	15	0	49	55	74	77	89	89	90
6	48	40	22	34	49	0	17	35	28	40	40	62
7	65	57	28	40	55	17	0	19	26	46	57	79
8	83	75	47	59	74	35	19	0	7	27	49	97
9	76	68	50	62	77	28	26	7	0	20	42	90
10	88	80	62	74	89	40	46	27	20	0	22	102
11	88	80	62	74	89	40	57	49	42	22	0	102
12	53	45	74	86	90	62	79	97	90	102	102	0

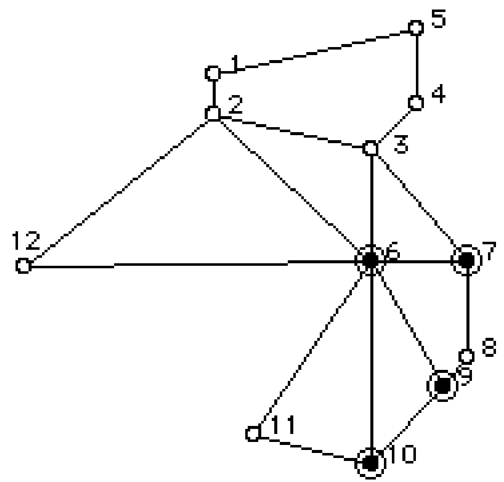
©D.Bricker, U. of Iowa, 1998

Odd-degree nodes

i:	6	7	9	10
Degree:	7	3	3	3

Shortest Paths

from \ to	6	7	9	10
6	0	17	28	40
7	17	0	26	46
9	28	26	0	20
10	40	46	20	0



©D.Bricker, U. of Iowa, 1998

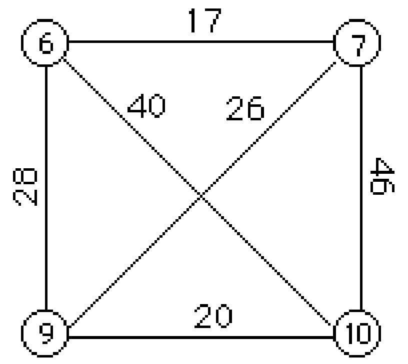
Odd-degree nodes

i:	6	7	9	10
Degree:	7	3	3	3

Shortest Paths

from \ to	6	7	9	10
6	0	17	28	40
7	17	0	26	46
9	28	26	0	20
10	40	46	20	0

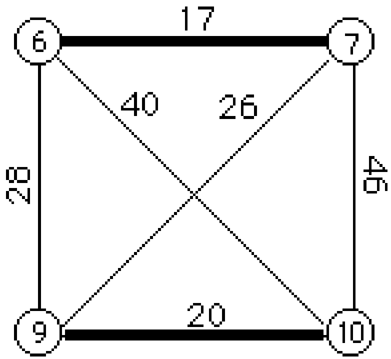
Minimum-weight matching to be solved in this network:

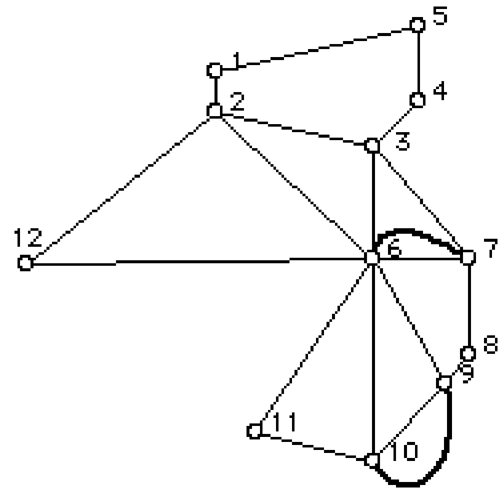


©D.Bricker, U. of Iowa, 1998

Matching

from: 6 9
to: 7 10





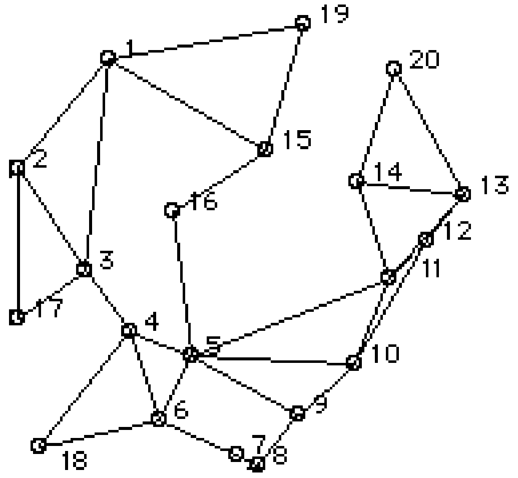
length 17 path: 6 7
length 20 path: 9 10

Total length of paths added: 37

©D.Bricker, U. of Iowa, 1998

Example

Random Network (seed= 454621)



©D.Bricker, U. of Iowa, 1998

from \ to	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	31	46	63	77	83	102	107	104	113	124	135	148	146	40	65	64	95	44	172
2	31	0	27	44	58	64	83	88	85	94	105	116	129	127	71	90	33	76	75	153
3	46	27	0	17	31	37	56	61	58	67	78	89	102	100	86	63	19	49	90	126
4	63	44	17	0	14	20	39	44	41	50	61	72	85	83	71	46	36	32	100	109
5	77	58	31	14	0	16	35	40	27	36	47	58	71	69	57	32	50	43	86	95
6	83	64	37	20	16	0	19	24	38	52	63	74	87	85	73	48	56	27	102	111
7	102	83	56	39	35	19	0	5	19	35	56	66	79	78	92	67	75	46	121	104
8	107	88	61	44	40	24	5	0	14	30	51	61	74	73	97	72	80	51	126	99
9	104	85	58	41	27	38	19	14	0	16	37	47	60	59	84	59	77	65	113	85
10	113	94	67	50	36	52	35	30	16	0	21	31	44	43	93	68	86	79	122	69
11	124	105	78	61	47	63	56	51	37	21	0	11	24	22	104	79	97	90	133	48
12	135	116	89	72	58	74	66	61	47	31	11	0	13	33	115	90	108	101	144	45
13	148	129	102	85	71	87	79	74	60	44	24	13	0	23	128	103	121	114	157	32
14	146	127	100	83	69	85	78	73	59	43	22	33	23	0	126	101	119	112	155	26
15	40	71	86	71	57	73	92	97	84	93	104	115	128	126	0	25	104	100	29	152
16	65	90	63	46	32	48	67	72	59	68	79	90	103	101	25	0	82	75	54	127
17	64	33	19	36	50	56	75	80	77	86	97	108	121	119	104	82	0	68	108	145
18	95	76	49	32	43	27	46	51	65	79	90	101	114	112	100	75	68	0	129	138
19	44	75	90	100	86	102	121	126	113	122	133	144	157	155	29	54	108	129	0	181
20	172	153	126	109	95	111	104	99	85	69	48	45	32	26	152	127	145	138	181	0

Shortest Path Lengths

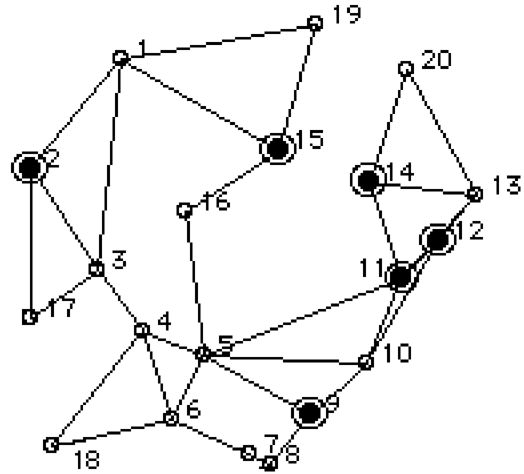
©D.Bricker, U. of Iowa, 1998

Odd-degree nodes

i:	2	9	11	12	14	15
Degree:	3	3	5	3	3	3

Shortest Path Lengths

from \ to	2	9	11	12	14	15
2	0	85	105	116	127	71
9	85	0	37	47	59	84
11	105	37	0	11	22	104
12	116	47	11	0	33	115
14	127	59	22	33	0	126
15	71	84	104	115	126	0



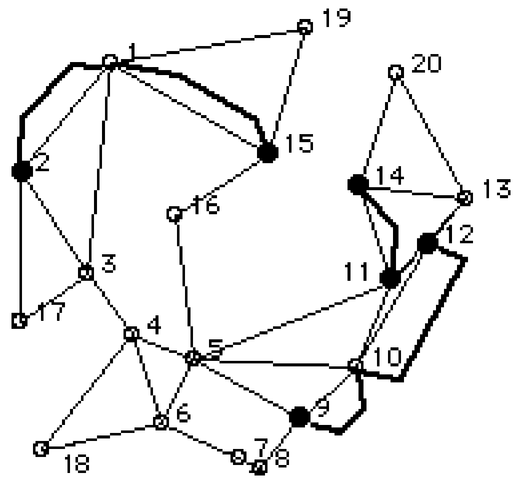
©D.Bricker, U. of Iowa, 1998

Optimal Matching

```
from:   2   9   11
  to:  15  12   14
```

```
length 71     path: 2 1 15
length 47     path: 9 10 12
length 22     path: 11 14
```

Total length of paths added: 140



The augmented network now possesses an Euler tour, which solves the postman problem!

©D.Bricker, U. of Iowa, 1998

Summary

- Solving the Postman Problem in an **undirected** network requires finding an optimal matching of the odd-degree nodes into pairs.
- Solving the Postman Problem in a **directed** network requires the solution of a transportation problem, with positive-polarity nodes as "sources" and negative-polarity nodes as "destinations".

In either case, the "cost" of a match or a shipment is the length of the shortest path between the two nodes.

©D.Bricker, U. of Iowa, 1998