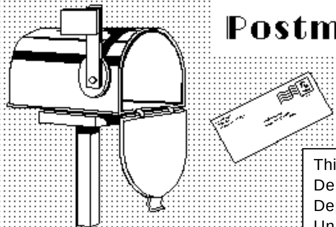
The Chinese Postman Problem



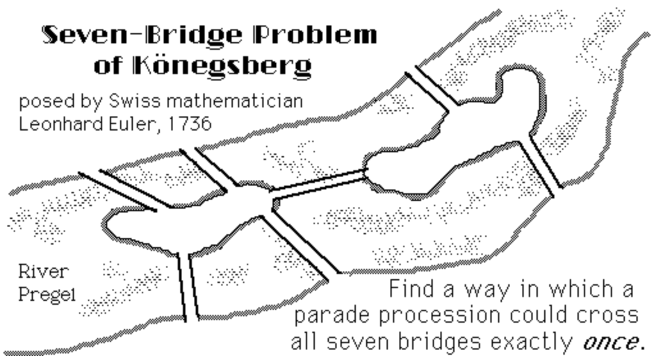
This Hypercard stack was prepared by:
Dennis Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242.
e-mail: dennis-bricker@uiowa.edu

author

Contents

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*.

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.

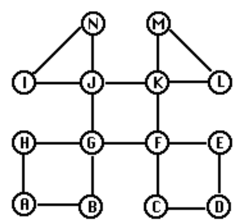
Define:

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.

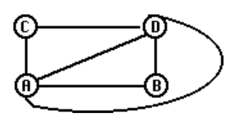
Outdegree of a node in a *directed* graph is the number of edges *from* the node.

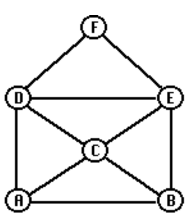
Polarity of a node of a directed graph is the difference: *indegree – outdegree*



Do these graphs possess either Euler tours or Euler paths?

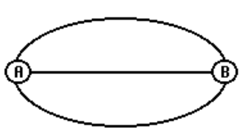
What is the degree of each node?

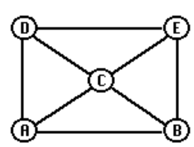




Do these graphs possess either Euler tours or Euler paths?

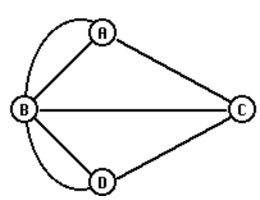
What is the degree of each node?





Do these graphs possess either Euler tours or Euler paths?

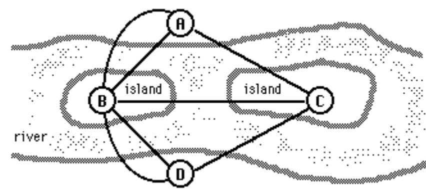
What is the degree of each node?



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

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

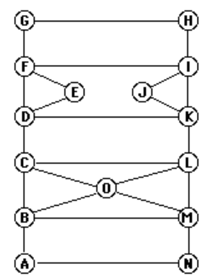
Finding an Euler Tour

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).

Example

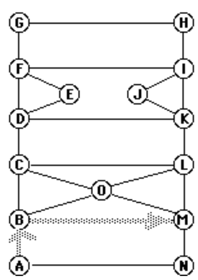
Suppose that you hold a summer job as highway inspector.
You must periodically drive along the highways, checking on debris & the need for repairs.
If you live in town A, is it possible to find a round trip which takes you over each section of highway ***exactly once?***



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:

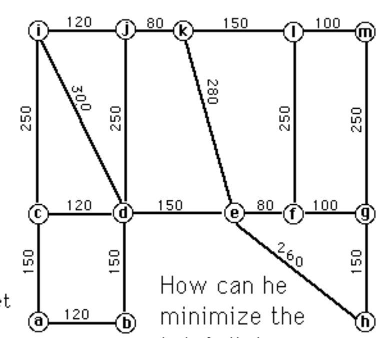
Then in choosing the next edge, we cannot choose edge MN, which has become a "bridge". Either MO or ML must be the next edge included in the tour!



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**.

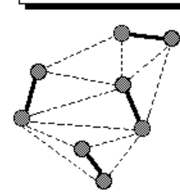


How can he minimize the total distance travelled?

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.)

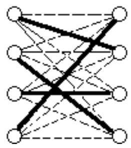
Matching Problem



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}

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.

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.

Matching Problem

Formulation: Define

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

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

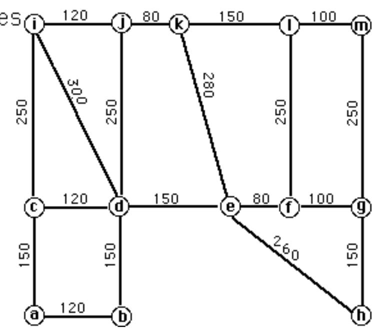
subject to

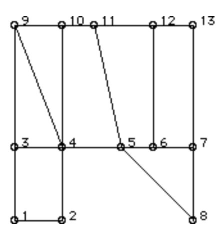
$$\sum_{j \neq i} X_{ij} = 1, \quad i=1,2, \ldots n$$

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

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.





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

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

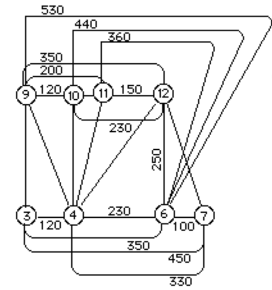
Chinese Postman Problem in a Graph

Odd-degree nodes

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

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



(not all edges are shown in the diagram!)

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)

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

(i) —120— (j) (c) —120— (d)

(k) —150— (l)

(f) —100— (g)

Total length: 490
(length to be traversed twice!)

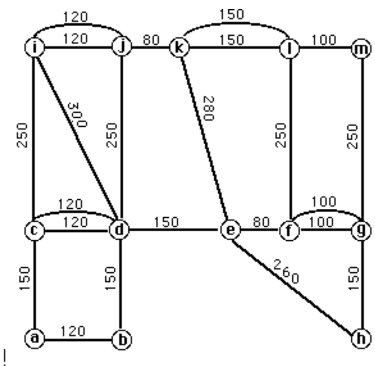
Add paths to the network:

(i) —120— (j) (c) —120— (d)

(k) —150— (l)

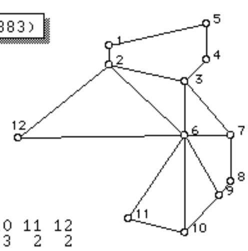
(f) —100— (g)

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



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

Shortest Path Lengths

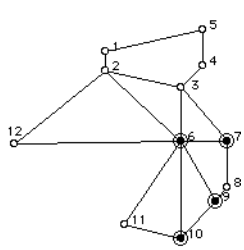
from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	8	37	49	37	48	65	83	76	88	88	53
2	8	0	29	41	45	40	57	75	68	80	80	45
3	37	29	0	12	27	22	28	47	50	62	62	74
4	49	41	12	0	15	34	40	59	62	74	74	86
5	37	45	27	15	0	49	55	74	77	89	89	90
6	48	40	22	34	49	0	17	35	28	40	40	62
7	65	57	28	40	55	17	0	19	26	46	57	79
8	83	75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

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



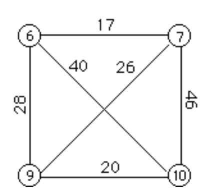
Odd-degree nodes

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

Shortest Paths

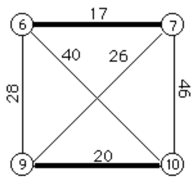
from \ to	6	7	9	10
6	0	17	28	40
7	17	0	26	46
9	28	26	0	20
10	40	46	20	0

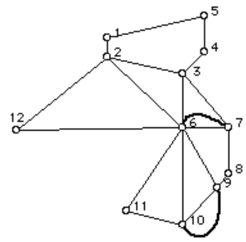
Minimum-weight matching to be solved in this network:



Matching

```
from:   6   9
  to:   7  10
```



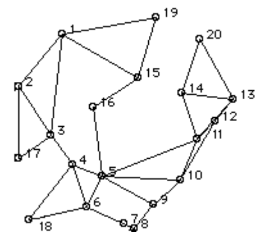



```
length 17     path: 6 7
length 20     path: 9 10
```

Total length of paths added: 37

Example

Random Network (seed= 454621)



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

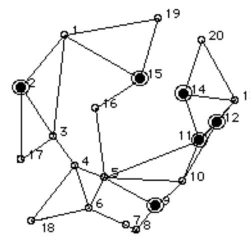
Shortest Path Lengths

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

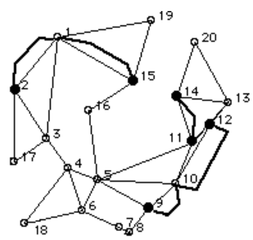


Optimal Matching

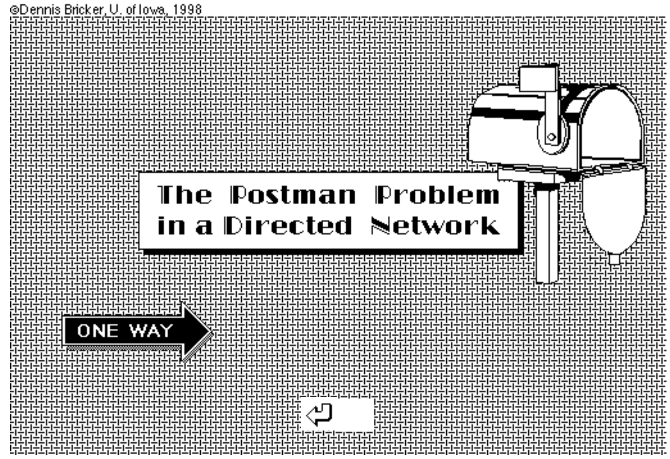
```
from:   2   9  11
  to:  15  12  14
```

```
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!



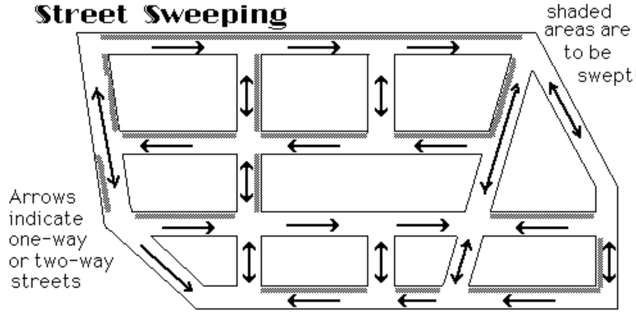
Example: Street Sweeping

ONE WAY

Plan a route for a street-sweeping machine which cleans all curbsides for which "No Parking" is currently in effect.

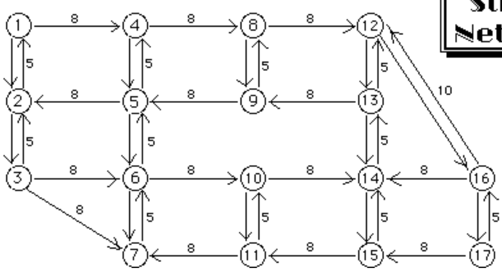
Some streets are one-way streets. Street-sweeping machine must obey any one-way restrictions, and on two-way streets must travel on the right side, with the flow of traffic.

Street Sweeping



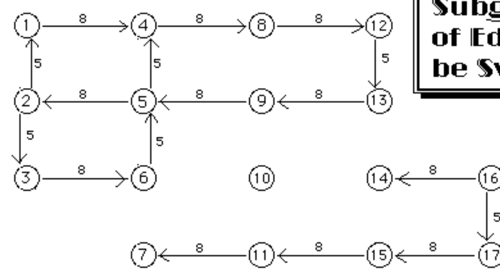
Travel times:
E-W: 8 minutes/block
N-S: 5 minutes/block

Diagonal blocks:
Lower left: 8 minutes
Upper right: 10 minutes



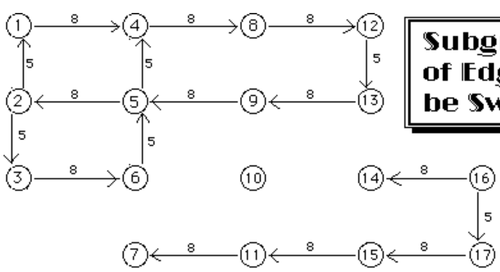
Street Network

Subgraph of Edges to be Swept



Clearly no Euler tour exists for this subgraph (which is not connected)! "Deadheading" will be necessary.

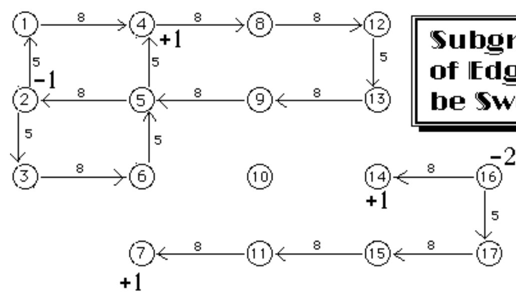
Subgraph of Edges to be Swept



Polarity = indegree - outdegree

Node:	2	4	7	14	16
Polarity:	-1	+1	+1	+1	-2

Subgraph of Edges to be Swept



How should we add an appropriate set of "dead-heading" arcs so that an Euler tour can be constructed?

If a node has positive polarity, then more arcs enter the node than leave it.... so to make the polarity zero, we must add arcs leaving the node.

Conversely, if the node has negative polarity, then arcs must be added which enter the node in order to make the polarity zero.

Consider the positive polarities as "sources" of shipments to be made, and the negative polarities as "demands".

The "shipping cost" is the cost of adding a path from a "source" node to a "demand" node, i.e., the length of the shortest path between the nodes.

SOURCE	DEMAND #2	DEMAND #16	supply
#4	13	26	1
#7	18	41	1
#14	29	20	1
rqmt	1	2	

Transportation Model

Minimize $13X_{4,2} + 26X_{4,16} + 18X_{7,2} + 41X_{7,16} + 29X_{14,2} + 20X_{14,16}$

subject to

$$X_{4,2} + X_{4,16} = 1$$
$$X_{7,2} + X_{7,16} = 1$$
$$X_{14,2} + X_{14,16} = 1$$
$$X_{4,2} + X_{7,2} + X_{14,2} = 1$$
$$X_{4,16} + X_{7,16} + X_{14,16} = 2$$

$X_{4,2} \geq 0, X_{7,2} \geq 0, X_{14,2} \geq 0,$
$X_{4,16} \geq 0, X_{7,16} \geq 0, X_{14,16} \geq 0$

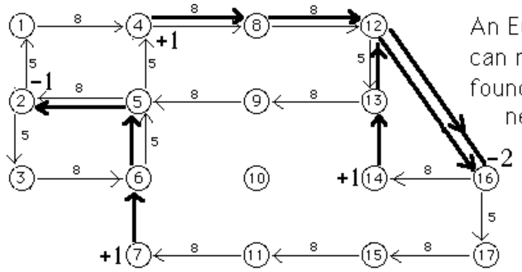
Optimal Solution:

SOURCE	DEMAND #2	DEMAND #16	supply
#4	13	1 | 26	1
#7	1 | 18	41	1
#14	29	1 | 20	1
rqmt	1	2	

Transportation Model

That is, add paths from:

node 4 to node 16
node 7 to node 2
node 14 to node 16



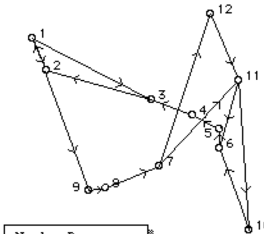
An Euler tour can now be found of the new network

The **bold** arcs indicate the "deadheading" travel, i.e., travel to be done while not sweeping.

Paths added:
node 4 to node 16
node 7 to node 2
node 14 to node 16

Another Example

Random Network (seed= 433760)



Node Degrees

node #	1	2	3	4	5	6	7	8	9	10	11	12
in-degree	1	2	2	1	1	2	1	1	1	1	2	1
out-degree	3	1	1	1	1	1	2	1	1	1	2	1
polarity	-2	1	1	0	0	1	-1	0	0	0	0	0

Distances (edge lengths)

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	13	54	999	999	999	999	999	61	999	999	999
2	13	0	999	999	999	999	999	999	999	999	999	999
3	999	44	0	999	999	999	999	999	999	999	999	999
4	999	999	18	0	999	999	999	999	999	999	999	999
5	999	999	999	12	0	999	999	999	999	999	999	999
6	999	999	999	999	7	0	999	999	999	999	999	999
7	999	999	999	999	999	999	0	999	999	999	46	61
8	999	999	999	999	999	999	23	0	999	999	999	999
9	999	999	999	999	999	999	999	7	0	999	999	999
10	999	999	999	999	999	32	999	999	999	0	999	999
11	999	999	999	999	999	26	999	999	999	55	0	999
12	999	999	999	999	999	999	999	999	999	999	28	0

999 <=> infinity, i.e., absence of arc

Path Lengths

from \ to	1	2	3	4	5	6	7	8	9	10	11	12
1	0	13	54	182	170	163	91	68	61	192	137	152
2	13	0	67	195	183	176	104	81	74	205	150	165
3	57	44	0	239	227	220	148	125	118	249	194	209
4	75	62	18	0	245	238	166	143	136	267	212	227
5	87	74	30	12	0	250	178	155	148	279	224	239
6	94	81	37	19	7	0	185	162	155	286	231	246
7	166	153	109	91	79	72	0	234	227	101	46	61
8	189	176	132	114	102	95	23	0	250	124	69	84
9	196	183	139	121	109	102	30	7	0	131	76	91
10	126	113	69	51	39	32	217	194	187	0	263	278
11	120	107	63	45	33	26	211	188	181	55	0	272
12	148	135	91	73	61	54	239	216	209	83	28	0

Predecessors

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

Chinese Postman Problem in a Digraph

i:	1	2	3	6	7
Polarity:	-2	1	1	1	-1

Shortest Paths

from \ to	1	7
2	13	104
3	57	148
6	94	185

Solving transportation problem

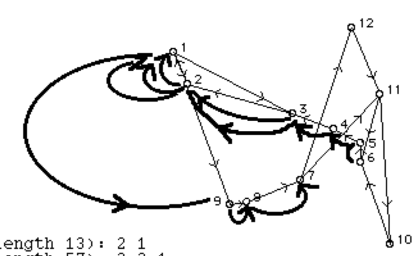
Solution of the transportation problem:

Paths added

from:	2	3	6
to:	1	1	7

1 × path (length 13): 2 1
1 × path (length 57): 3 2 1
1 × path (length 185): 6 5 4 3 2 1 9 8 7

Total length of paths added: 255



1 × path (length 13): 2 1
1 × path (length 57): 3 2 1
1 × path (length 185): 6 5 4 3 2 1 9 8 7

Total length of paths added: 255

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.

REFERENCE

H.A. Eiselt, Michel Gendreau, & Gilbert Laporte,

"Arc Routing Problems, Part I: The Chinese Postman Problem", Operations Research, volume 43 (March/April 1995), pp. 231-242.

"Arc Routing Problems, Part II: The Chinese Postman Problem", Operations Research, volume 43 (May/June 1995), pp. 399-414.