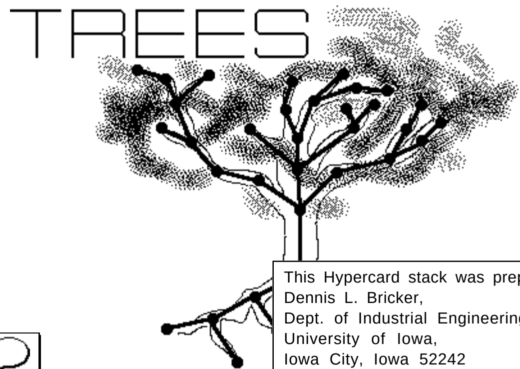
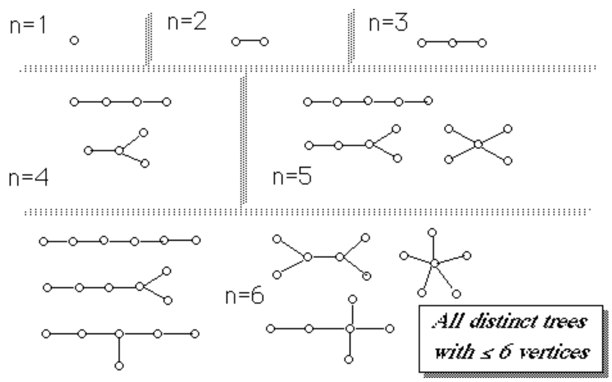
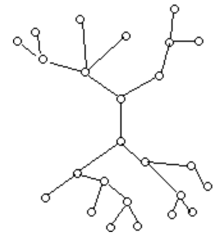


This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dennis-bricker@uiowa.edu

author

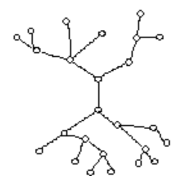


Tree : a connected graph without cycles



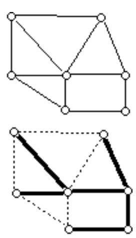
The following statements about a graph G are equivalent:

- G is a tree
- G is connected with *n* vertices and *n-1* edges
- G has *n* vertices, *n-1* edges, and no cycles
- G is such that each pair of vertices is connected by a *unique* elementary chain



Spanning tree

A spanning tree of a connected graph G=(V,A) is a tree with vertex set V and an edge set which is a subset of A



Minimum spanning tree

A minimum spanning tree of a *network* is a spanning tree the sum of whose edge lengths are minimal.

Two algorithms for MST problem:

☞ Prim's Algorithm

Beginning with a single node, at each iteration a tree is obtained by adding an edge & node, until ALL nodes have been included.

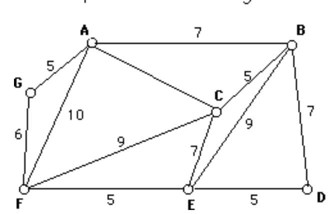
☞ Kruskal's Algorithm

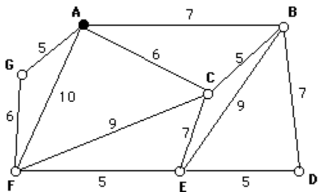
Beginning with N trees, each consisting of a single node, at each iteration two trees are combined by adding an edge, until a single tree is obtained.

Finding a Minimum Spanning Tree (MST) of a Network (Prim's algorithm)

Step 1 (Setup)
Select any node to begin the tree
Step 2 (Addition)
Find a node NOT currently in the tree which is nearest to the set of nodes IN the tree. Add that node and the connecting edge to the tree
Step 3 (Stopping criterion)
If all nodes are in the tree, STOP; otherwise return to step 2

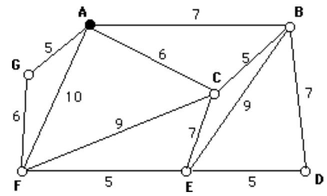
Example: Prim's algorithm for MST





Initially, the tree is empty.

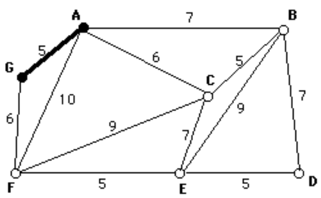
Select (arbitrarily) node A to add to the tree.



Find the node which is nearest to the nodes of the tree (i.e., node A)

This is node G.

Add it (and edge [A,G]) to the tree

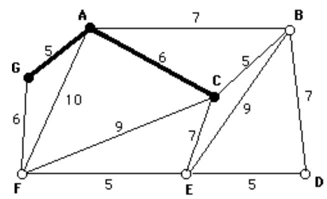


Find the node in the set {B,C,D,E,F} (not in the tree) which is nearest to the nodes {A,G} which are in the tree.

In this case there is a tie!

Break the tie arbitrarily, by selecting node C.

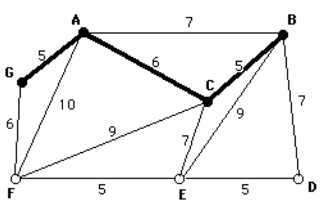
Add node C (and edge [A,C]) to the tree



Find the node from the set {B,D,E,F} (not in the tree) which is nearest to the nodes {A,C,G} (in the tree)

This is node B, a distance 5 from the tree.

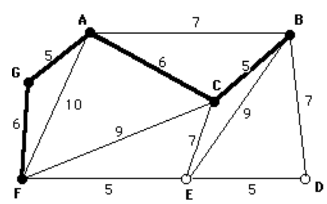
Add the node B (and the edge [B,C]) to the tree



Find the node from the set {D,E,F} which is nearest to the set of nodes in the tree, {A,B,C,G}.

This is node F, a distance of 6 from the tree.

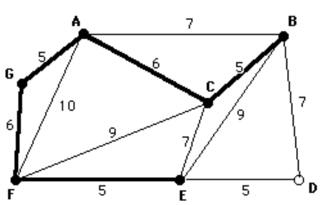
Add node F (and edge [F,G]) to the tree



Find the node from the set {D,E} which is nearest to the nodes in the tree, {A,B,C,F,G}.

This is node E, a distance of 5 from the tree.

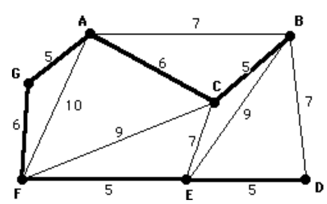
Add node E (and edge [E,F]) to the tree.



Find the node from the set {D} which is nearest to the nodes {A,B,C,E,F,G} in the tree.

This is node D, a distance of 5 from the tree.

Add node D (and edge [D,E]) to the tree.



All nodes are now in the tree, so we stop!

Example: **Alaska Gas Transmission Company** is planning to construct a pipeline to supply gas from Alaska's north slope ("NS") to eight U.S. gas companies, denoted by A through H.

Each mile of "right-of-way which is purchased costs an average of $1000.

How should the pipeline be routed to minimize the total cost of the right-of-way?

distances (x100 miles)

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

Arbitrarily select a node to begin the tree.

Let's choose node NS.

non-TREE

TREE

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

Find the minimum distance from a node NOT in the tree to the node IN the tree.

This is node A.

Add node A (and edge [NS,A]) to the tree.

non-TREE

TREE

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

Find the node NOT in the tree which is nearest to the nodes IN the tree.

This is node B, a distance of 12 from node A.

Add node B (& edge [A,B]) to the tree.

non-TREE

TREE

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

Find the node NOT in the tree which is nearest to the nodes IN the tree.

This is node E, a distance of 11 from node B.

Add node E (& edge [B,E]) to the tree.

non-TREE

TREE

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

Find the node NOT in the tree which is nearest to the nodes IN the tree.

This is node D, which is a distance 5 from node E.

Add node D (& edge [D,E]) to the tree.

non-TREE

TREE

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

Find the node NOT in the tree which is nearest to the nodes IN the tree.

This is node C, a distance of 10 from node D.

Add node C (& edge [C,D]) to the tree.

non-TREE

TREE

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

Find the node NOT in the tree which is nearest to the nodes IN the tree.

This is node H, a distance of 12 from node E.

Add node H (& edge [E,H]) to the tree.

non-TREE

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

TREE

Find the node NOT in the tree which is nearest to the nodes IN the tree.

This is node G, a distance of 8 from node H.

Add node G (& edge [G,H]) to the tree.

non-TREE

TREE

	NS	A	B	C	D	E	F	G	H
NS	0	32	43	41	44	45	53	56	61
A	32	0	12	15	16	17	31	25	32
B	43	12	0	18	12	11	32	26	28
C	41	15	18	0	10	14	23	15	18
D	44	16	12	10	0	5	22	13	16
E	45	17	11	14	5	0	23	15	12
F	53	31	32	23	22	23	0	7	14
G	56	25	26	15	13	15	7	0	8
H	61	32	28	18	16	12	14	8	0

Find the node NOT in the tree which is nearest to the nodes IN the tree.

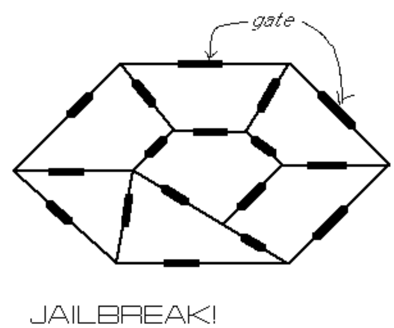
This is node F, a distance of 7 from node G.

Add node F (& edge [F,G]) to the tree.

The tree now spans all nine nodes, and is the Minimum Spanning Tree.

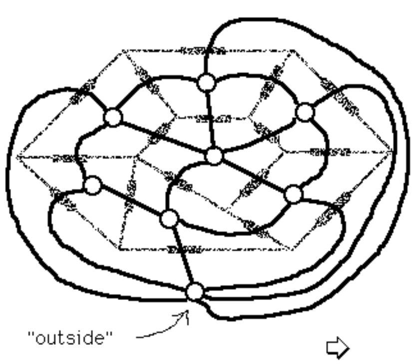
APL code for Prim's MST algorithm

```
    ∇TREE←MST C;IN;OUT;K;L;ROWMIN;MIN;J
     ⍝
     ⍝  Compute Minimum Spanning Tree of a graph
     ⍝
     IN←,1                             ⍝ List of nodes in tree
     OUT←1+⍳¯1+1↑⍴C                    ⍝ List of nodes not yet in
     TREE←(⍴C)⍴0
     LENGTH←0
     ⍝          Find shortest arc joining IN & OUT nodes
     NEXT:ROWMIN←⌊/C[IN;OUT]
     MIN←⌊/ROWMIN
     J←ROWMIN⍳MIN
     ⍝          Add arc from IN node (K) to OUT node (L)
     K←IN[J]
     L←OUT[C[K;OUT]⍳MIN]
     TREE[K;L]←1
     OUT←(L≠OUT)/OUT
     IN←IN,L
     LENGTH←LENGTH+MIN
     →NEXT IF (⍴IN)<1↑⍴C
    ∇
```



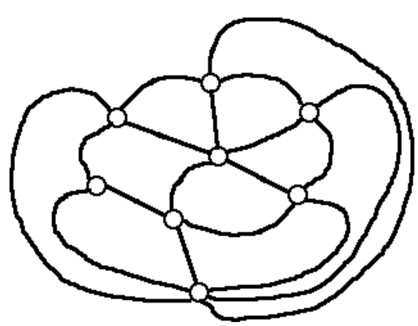
- Prisoners have been divided into seven groups by walls
- An outside accomplice plans to help them to escape by blowing up some of the gates, using explosives

HOW CAN HE DO THIS, DESTROYING AS FEW GATES AS POSSIBLE?



Represent each room, together with the "outside world", by a node, and each gate by an edge.

The problem is to find a spanning tree with the fewest edges!



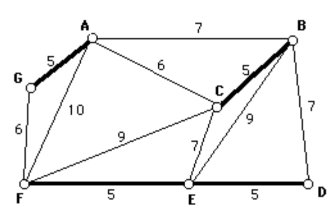
The number of nodes is 8.

All spanning trees will have seven edges!

Kruskal's Algorithm for MST

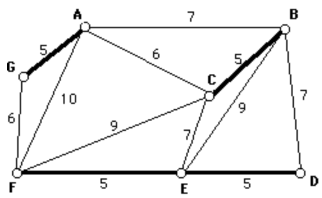
Step 1: Setup
  Let $G_0 = (V, \varnothing)$ and i=0
Step 2: Addition of Edge
  Find (x,y) which minimizes w(x,y), and set $w(x,y) = +\infty$
Step 3: Test for cycle
  If the addition of edge (x,y) to the graph G_i would form a cycle, then go to step 2;
  Otherwise, add edge (x,y) to graph G_i and increment i.
Step 4: Test for termination
  If $i < n-1$, then return to step 2.
  Otherwise, stop with G_{n-1} = MST

Example (Kruskal's MST Algorithm)



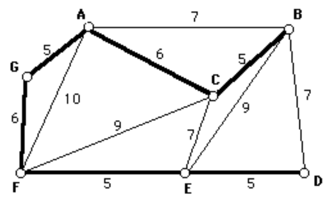
i	Edges in G_i
0	none
1	AG
2	AG, BC
3	AG, BC, DE
4	AG, BC, DE, EF

In each of the first 4 iterations, there is a tie for the minimum-length edge to be added



i	Edges in G_i
0	none
1	AG
2	AG, BC
3	AG, BC, DE
4	AG, BC, DE, EF

Next, there is a tie between edges FG and AC

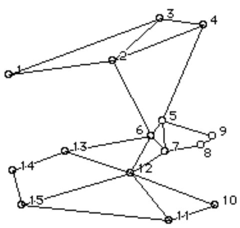


i	Edges in G_i
0	none
1	AG
2	AG, BC
3	AG, BC, DE
4	AG, BC, DE, EF
5	AG, BC, DE, EF, FG
6	AG, BC, DE, EF, FG, AC

Since i=6 = n-1, we terminate.

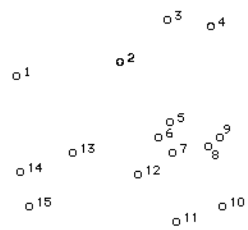
Example (Kruskal's MST Algorithm)

A network with 15 nodes:



i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

We begin with 15 "trees", each consisting of a single node:



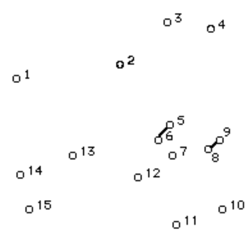
edges sorted according to length

i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

The two trees consisting of nodes 8 & 9 are joined, so that we now have 14 trees:

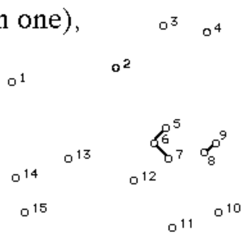
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Next, edge (5,6) is added, which joins two trees, resulting in only 13 trees:



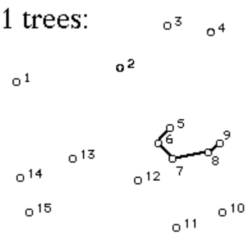
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Edge (6,7) is next added, combining two trees (one with 2 nodes, the other with one), giving us 12 trees:



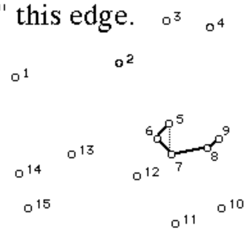
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Edge (7,8) is added, combining trees {5,6,7} and {8,9}, giving us only 11 trees:



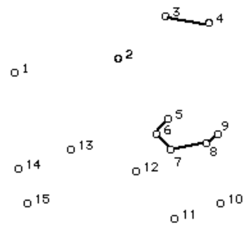
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

If edge (5,7) were added, a cycle 5-7-6-5 would be formed, and so we "skip" this edge.



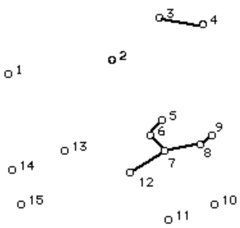
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Edge (3,4) is added next, reducing the number of trees to only 10:



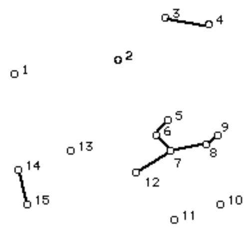
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Edge (7,12) is added, reducing the number of trees to nine:



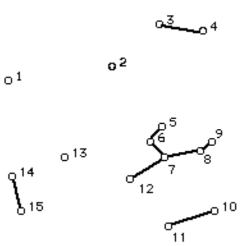
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Edge (14,15) is added, reducing the number of trees to eight:



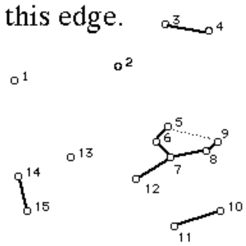
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Adding edge (10,11) reduces the number of trees to seven:



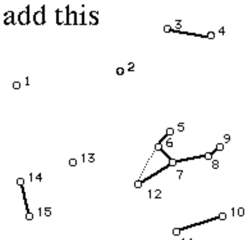
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Adding edge (5,9) would create a cycle (5-9-8-7-6-5) and so we don't add this edge.



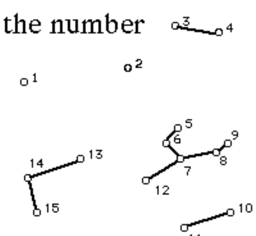
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Adding edge (6,12) also would create a cycle (6-12-7-6), and so we don't add this edge:



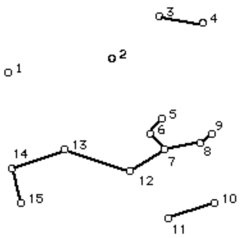
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Adding edge (13,14) doesn't create a cycle, and so we add this edge, reducing the number of trees to only six:



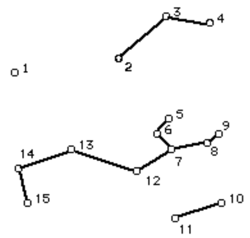
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Edge (12,13) is added, reducing the number of trees to five:



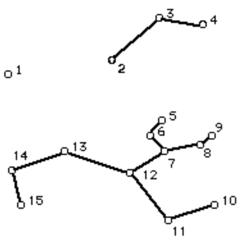
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Edge (2,3) is added next, reducing the number of trees to four:



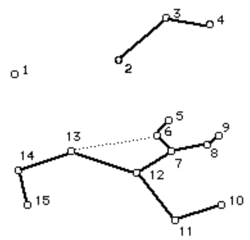
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Next we add edge (11,12), to obtain three trees:



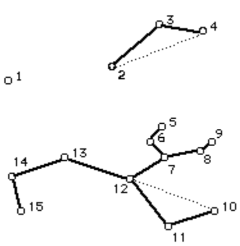
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Adding edge (6,13) would form a cycle, so we skip it:



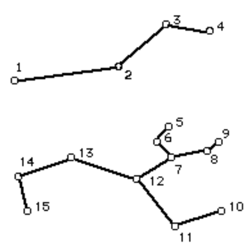
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Adding edge (10,12) would form a cycle, as would edge (2,4):



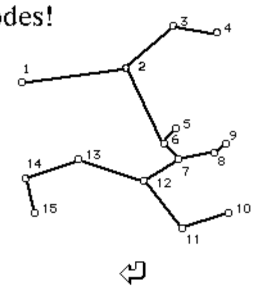
i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

The next edge to be added is (1,2), which leaves us with only two trees!



i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57

Finally, adding edge (2,6) leaves us with a single tree, spanning all of the nodes!



i	j	d_{ij}
9	8	6
5	6	8
7	6	9
8	7	12
5	7	14
3	4	15
7	12	16
15	14	16
10	11	17
5	9	18
12	6	18
14	13	20
12	13	24
3	2	25
12	11	26
13	6	30
12	10	33
4	2	35
1	2	36
6	2	37
12	15	40
4	5	46
11	15	50
1	3	57