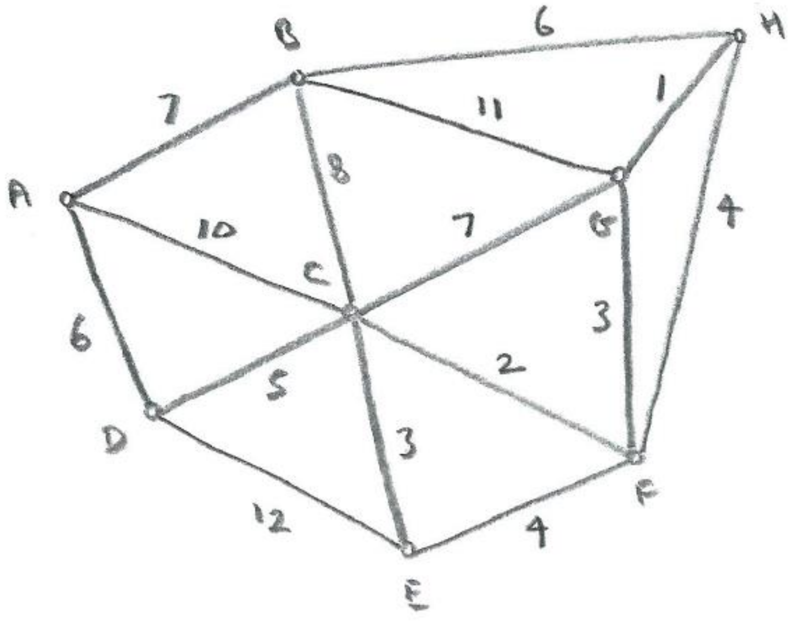
Minimum Connector Problem (6 pages; 21/11/20)

The aim is to find a minimum spanning tree for a given network; ie a tree which connects all the nodes, and which has the smallest possible total weight.

The two standard methods are Kruskal's algorithm and Prim's algorithm. The following network will be used to illustrate them.



Kruskal's algorithm

(1) Start with the shortest arc.

(2) Choose the next shortest arc, provided it doesn't create a cycle.

(3) Repeat until all the nodes have been included.

Notes

(i) If two arcs are of the same length, then either can be chosen. Different minimum spanning trees may result, but their total lengths will be the same.

(ii) It doesn't matter that the arcs that have been chosen are not connected (until the final spanning tree is obtained).

(iii) Because it is awkward for a computer program to look for cycles, Prim's algorithm is usually preferred (especially in the matrix form - see below).

Example

The order in which arcs are added needs to be made clear, and any arcs that are rejected (because they create a cycle) should be mentioned.

GH 1

CF 2

GF 3 [or CE]

CE 3

(FH 4 rejected)

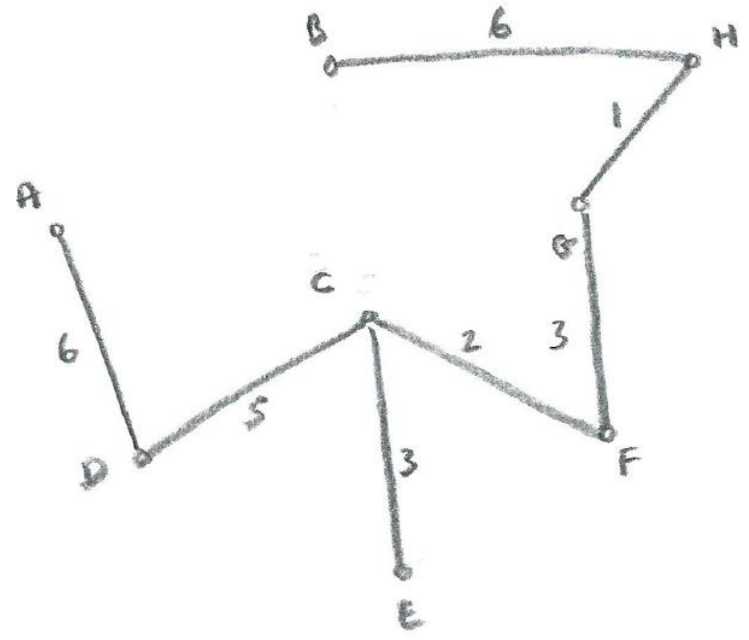
(EF 4 rejected)

CD 5

AD 6 [or BH]

BH 6

Total = 26



Prim's Algorithm

(1) Start with any node (say A). Let T be the tree consisting (initially) of A. Let R be the network remaining after A has been removed from the original network.

(2) Identify the nearest node of R that is joined to T by a direct arc. Add this node and arc to T, and remove them from R.

(3) Repeat until all nodes have been included in T.

Note

If there is more than one nearest node, then either may be chosen. Different minimum spanning trees may result, but their total lengths will be the same.

Example

Once again, the order in which arcs are added needs to be made clear.

AD 6

DC 5

CF 2

CE 3 [or FG]

FG 3

GH 1

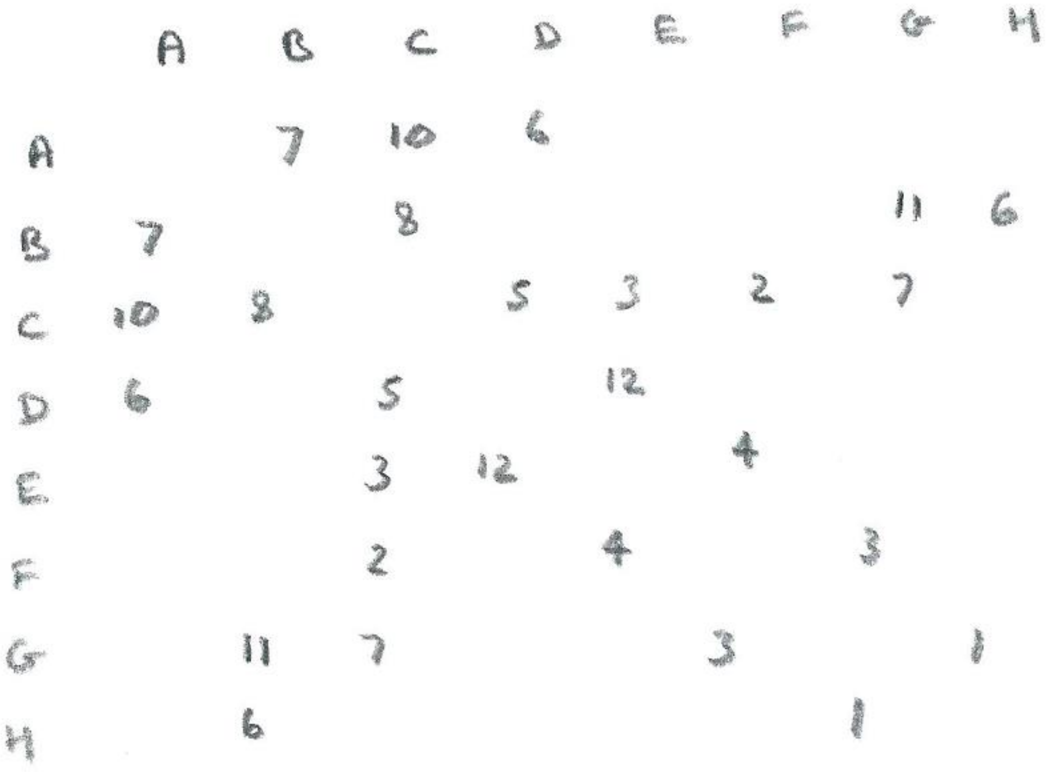
BH 6

Total = 26

[Note that there are no arcs to be rejected for Prim's algorithm.]

The minimum spanning tree is the same as for Kruskal's algorithm (for this example).

Prim's Algorithm using the matrix of distances



	A	B	C	D	E	F	G	H
A		7	10	6				
B	7		8				11	6
C	10	8		5	3	2	7	
D	6		5		12			
E			3	12		4		
F			2		4		3	
G		11	7			3		1
H		6					1	

Referring to the steps above, the first node to be added to T (A in this example) has a '(1)' placed above its column, and its row is crossed out (to remove it from consideration when looking for further nodes of R to transfer to T).

We then look down the column of A, to find the node in R that is nearest to T (T being just A initially). In this example, this is D. So D (together with the arc joining it to T) is added to T and removed from R. In the matrix, a '(2)' is placed above column D, and row D is crossed out (as D is no longer in R). Also, the distance of D from A (ie 6) is ringed, so that the length of the spanning tree can be established. As an ongoing check, it may be useful to keep a record of the arcs being added.

	(1)			(2)				
	A	B	C	D	E	F	G	H
A	~~ ~~	~~7~~	~~10~~	~~6~~	~~ ~~	~~ ~~	~~ ~~	~~ ~~
B	7		8				11	6
C	10	8		5	3	2	7	
D	~~$\textcircled{6}$~~	~~ ~~	~~5~~	~~ ~~	~~12~~	~~ ~~	~~ ~~	~~ ~~
E			3	12		4		
F			2		4		3	
G		11	7			3		1
H		6					1	

AD 6

The position after the 3rd node has been added to T is shown below. The process is then repeated until all nodes have been added to T.

	$\textcircled{1}$		$\textcircled{3}$	$\textcircled{2}$				
	A	B	C	D	E	F	G	H
A	~~—~~	~~7~~	~~10~~	~~6~~				
B	7		8				11	6
C	~~10~~	~~8~~		~~$\textcircled{5}$~~	~~3~~	~~2~~	~~7~~	
D	~~$\textcircled{6}$~~		~~5~~		~~12~~			
E			3	12		4		
F			2		4		3	
G		11	7			3		1
H		6					1	

AD	6
DC	5