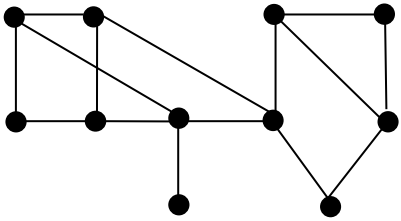
8. GRAPH ALGORITHMS

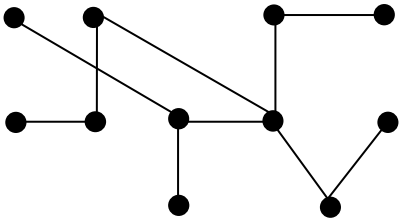
§8.1. Minimal Spanning Trees

Starting with any connected undirected graph we can obtain a tree by removing edges so as to remove cycles. If the resulting tree includes all the vertices we say that it is a **spanning tree**.

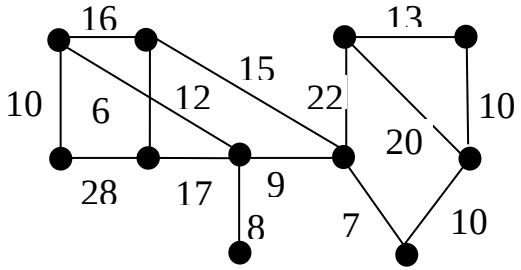
Example 5: The following undirected graph G is connected.



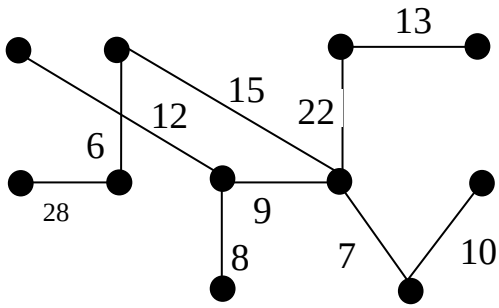
The following is a spanning tree for G.



Let's now make G into a weighted graph.



We transfer these weights to our spanning tree.



The total weight of this spanning tree is 130. Now the vertices in our original weighted graph might represent towns, with the weights being the distances between them. If we want to build roads connecting these towns, as cheaply as possible, we might choose to base the network on a spanning tree. This particular spanning tree would require a total of 130 kilometres of road. But perhaps there is one whose total weight is less than 130.

The **weight** of a spanning tree is the sum of the weights of its edges. A **minimal spanning tree** in a weighted undirected graph is a spanning tree of lowest weight. How

might we systematically go about finding such a minimal spanning tree?

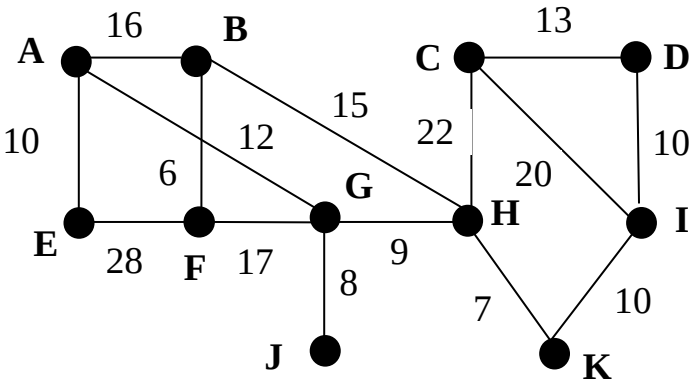
§8.2. Kruskal's Algorithm

Kruskal's Algorithm (J.B. Kruskal) finds a minimal spanning tree in a weighted undirected graph. We start with an edge of lowest weight. At each stage we examine all the edges which could be added without producing a cycle, and of these edges we choose one of lowest weight. While this seems a good strategy it's not clear that it will guarantee a minimal spanning tree at the end.

KRUSKAL'S ALGORITHM for finding a minimal spanning tree in a connected, weighted undirected graph.

(1) Create a list of the edges and their weights, in ascending order of weights. Call this list A.
(2) Create an empty list of edges, and their weights, called list B.
(3) Delete any edges from list A which would create a cycle if added to the graph in list B.
(4) If list A is empty END.
(5) Transfer the first edge from list A to list B.
(6) Go to step (3).
List B will now contain the edges of a minimal spanning tree.

Example 5 (continued): We name the vertices.

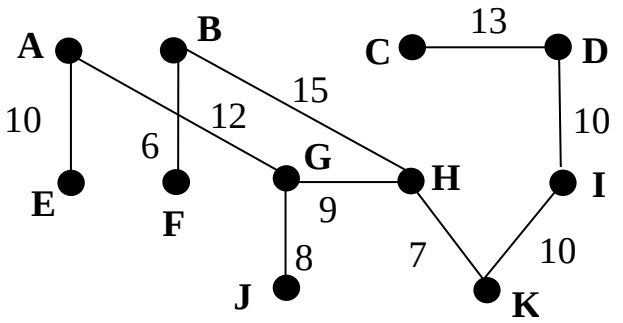


STEP	LIST A	LIST B
(1), (2)	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	empty
(3), (4), (5)	HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6
(3), (4), (5)	GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7
(3), (4), (5)	GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8

STEP	LIST A	LIST B
(3), (4), (5)	AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9
(3), (4), (5)	DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10
(3), (4), (5)	IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10 DI10
(3), (4), (5)	AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10 DI10 IK10
(3), (4), (5)	CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12
(3), (4), (5)	BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13
(3), (4)	BH15 AB16 FG17 EF28	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13
(5)	AB16 FG17 EF28	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15

STEP	LIST A	LIST B
END	empty	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15

This minimal spanning tree is



and its weight is 100, a big improvement on the previous spanning tree. Moreover, by Kruskal's Theorem, we'll never be able to do any better.

An important stage in the algorithm is to delete those edges which, when added to the tree in list B, would create a cycle. If we have a diagram to look at we can do this 'by inspection'.

But if we have a graph with many thousand vertices there's no way we'd attempt to draw it. The graph will only ever exist as a list of edges, together with their

weights. So how would our Kruskal program deal with this?

We can do this by keeping a list of the components in our graph **B**. Then, as we scan through the edges in list **A**, we delete those edges whose two vertices lie in the same component.

If vertices P, Q were in the same component there would be a path connecting them. Adding the edge PQ would provide a second path from P to Q and hence there would be a cycle. In this case we'd delete that edge from list A. Otherwise that edge remains, unless it is chosen to be transferred to list B.

EXPANDED KRUSKAL'S ALGORITHM for finding a minimal spanning tree in a connected, weighted undirected graph.

(1) Create a list of the edges and their weights, in ascending order of weights. Call this list A.
(2) Create list B, an empty list of edges and their weights.
(3) Create an empty list of components. Call it list C.
(4) Delete any edges from list A where both endpoints lie in the same component.
(5) If list A is empty END.
(6) Transfer the first edge PQ from list A to list B.
(7) If neither P nor Q is in a component create a new component PQ.

(8) If P is in a component and Q is not add Q to that component.
(9) If P, Q are in different components combine these components into one.
(10) Go to step (4).

Example 5 (again):

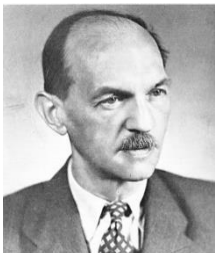
STEP	LIST A	LIST B	LIST C
(1), (2), (3)	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	empty	empty
(4), (5), (6)	HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6	empty
(7)	HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6	BF
(8), (9), (4) – (6)	GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7	BF

STEP	LIST A	LIST B	LIST C
(7)	GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7	BF HK
(8), (9), (4) – (7)	GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8	BF HK GJ
(8), (9), (4) – (7)	AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9	BF GHJK
(8), (9), (4) – (7)	DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10	BF GHJK AE
(8), (9), (4) – (7)	IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10 DI10	BF GHJK AE DI
(8), (9), (4) – (7)	AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10 DI10 IK10	BF DGHIJK AE

STEP	LIST A	LIST B	LIST C
(8), (9), (4) – (7)	CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12	BF ADEGHIJ K
(8), (9), (4) – (7)	BH15 AB16 FG17 CI20 CH22 EF28	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13	BF ACDEGHI JK
(8), (9), (4) – (7)	AB16 FG17 EF28	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15	ABCDEFG HIJK

STEP	LIST A	LIST B	LIST C
END	empty	BF6 HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15	ABCDEFG HIJK

§8.3. Prim's Algorithm



JARNIK

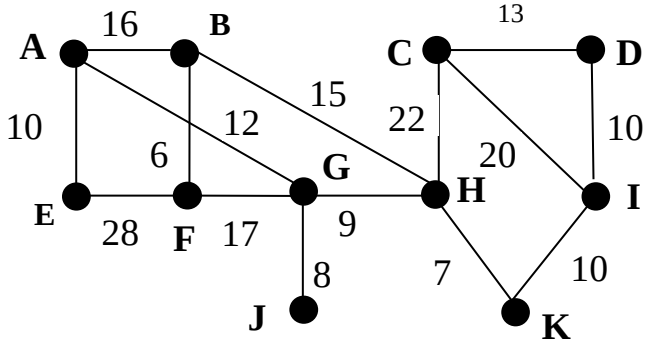
PRIM

Another algorithm to find a minimal spanning tree was first developed by Vojtech Jarnik in 1930 but was independently discovered by Robert Prim in 1957 and is known as Prim's algorithm.

Often different spanning trees are found by Kuskal's and Prim's algorithms, but of course the total weight will be the same.

PRIM'S ALGORITHM:
(1) Create a list, called list A, containing all the edges except one of lowest weight.
(2) Create a list, called list B, consisting of this edge.
(3) Create a list, called list C, consisting of the endpoints of this edge.
(4) If list A is empty END.
(5) Remove any edges in list A that have both endpoints in list C.
(6) Of all the edges in list A that have exactly one endpoint in list B, choose one of lowest weight, transfer it to list B and add the other endpoint to list C.
(7) Go to step (4).
List B will consist of the edges in a minimal spanning tree.

Example 5 (continued):



LIST A	LIST B	LIST C
HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 BH15 AB16 FG17 CI20 CH22 EF28	BF6	BF
HK7 GJ8 GH9 AE10 DI10 IK10 AG12 CD13 AB16 FG17 CI20 CH22 EF28	BF6 BH15	BFH
GJ8 GH9 AE10 DI10 IK10 AG12 CD13 AB16 FG17 CI20 CH22 EF28	BF6 BH15 HK7	BFHK
GJ8 AE10 DI10 IK10 AG12 CD13 AB16 FG17 CI20 CH22 EF28	BF6 BH15 HK7 GH9	BFGHK
AE10 DI10 IK10 AG12 CD13 AB16 CI20 CH22 EF28	BF6 BH15 HK7 GH9 GJ8	BFGHJK
AE10 DI10 AG12 CD13 AB16 FG17 CI20 CH22 EF28	BF6 BH15 HK7 GH9 GJ8 IK10	BFGHIJK
AE10 AG12 CD13 AB16 FG17 CI20 CH22 EF28	BF6 BH15 HK7 GH9 GJ8 IK10 DI10	BDFGHIJK

LIST A	LIST B	LIST C
AE10 CD13 AB16 CI20 CH22 EF28	BF6 BH15 HK7 GH9 GJ8 IK10 DI10 AG12	ABDFGHIJK
CD13 AB16 CI20 CH22 EF28	BF6 BH15 HK7 GH9 GJ8 IK10 DI10 AG12 AE10	ABDEFGHIJK
CI20 CH22	BF6 BH15 HK7 GH9 GJ8 IK10 DI10 AG12 AE10 CD13	ABCDEFGHIJK
	BF6 BH15 HK7 GH9 GJ8 IK10 DI10 AG12 AE10 CD13	ABCDEFGHIJK

This is the same spanning tree as was found by Kruskal's Algorithm.

coopersnotes.net
LIST OF TITLES

GENERAL
- The Mathematics At The Edge Of The Rational Universe

ELEMENTARY
- Basic Mathematics
- Concepts of Algebra
- Concepts of Calculus
- Elementary Algebra
- Elementary Calculus

1st YEAR UNI
- Techniques of Algebra
- Techniques of Calculus
- Matrices

2nd YEAR UNI
- Linear Algebra
- Languages & Machines
- Discrete Mathematics

3rd YEAR UNI
- Group Theory volume 1
- Group Theory volume 2
- Galois Theory
- Graph Theory
- Number Theory
- Geometry
- Topology
- Set Theory

POSTGRADUATE
- Ring Theory
- Representation Theory
- Quadratic Forms
- Group Tables vol 1
- Group Tables vol 2