Network Optimization Models

Class 20

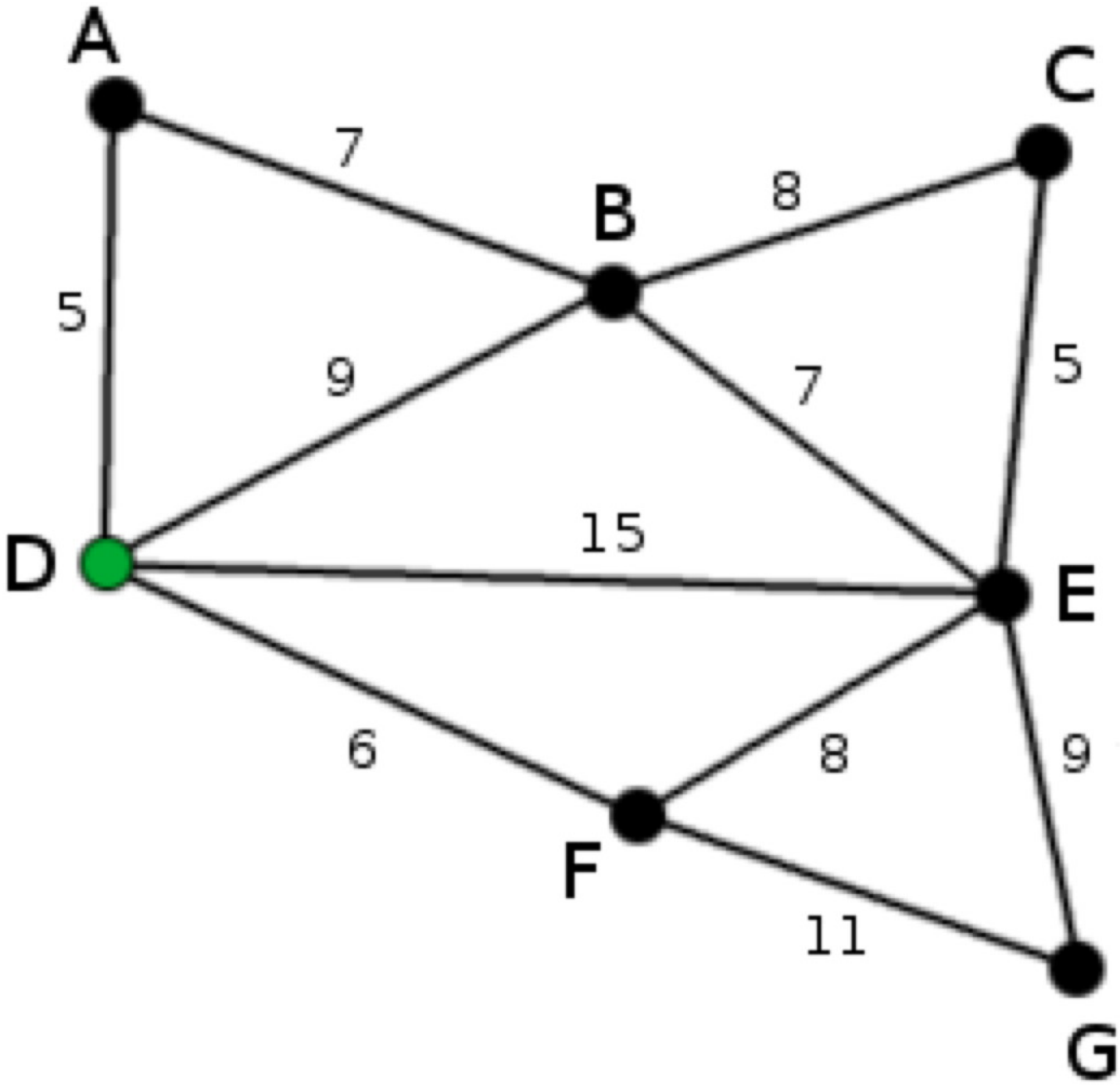
April 5, 2023

Network Optimization Problems

Minimal Spanning Tree

Shortest Path

Maximum Flow


A
C
B
7
8
5
9
7
5
15
D
E
6
8
9
F
11
G

Basic Vocabulary

A **Graph = Network** consists of a set of vertices and a set of edges connecting certain pairs of the vertices.

Vertex = Node = Point

Edge = Link = Arc

A **path** between two vertices is a finite sequence of distinct edges that connects the vertices.

Two vertices are **connected** if there is a path between them.

A **connected network** is a network where every pair of vertices is connected

Tree: Connected Graph With No Cycles

Spanning Tree: A Tree Containing All Vertices of a Graph

Minimal Spanning Tree Problem

There are n points in the plane whose mutual distances are given. The problem is to join them with a network in such a way that:

- Any two points are joined to each other either directly or by means of some other points and
- the total length of the network will be minimal.

Otakar Bůruvka (1926) [Electrical Networks]

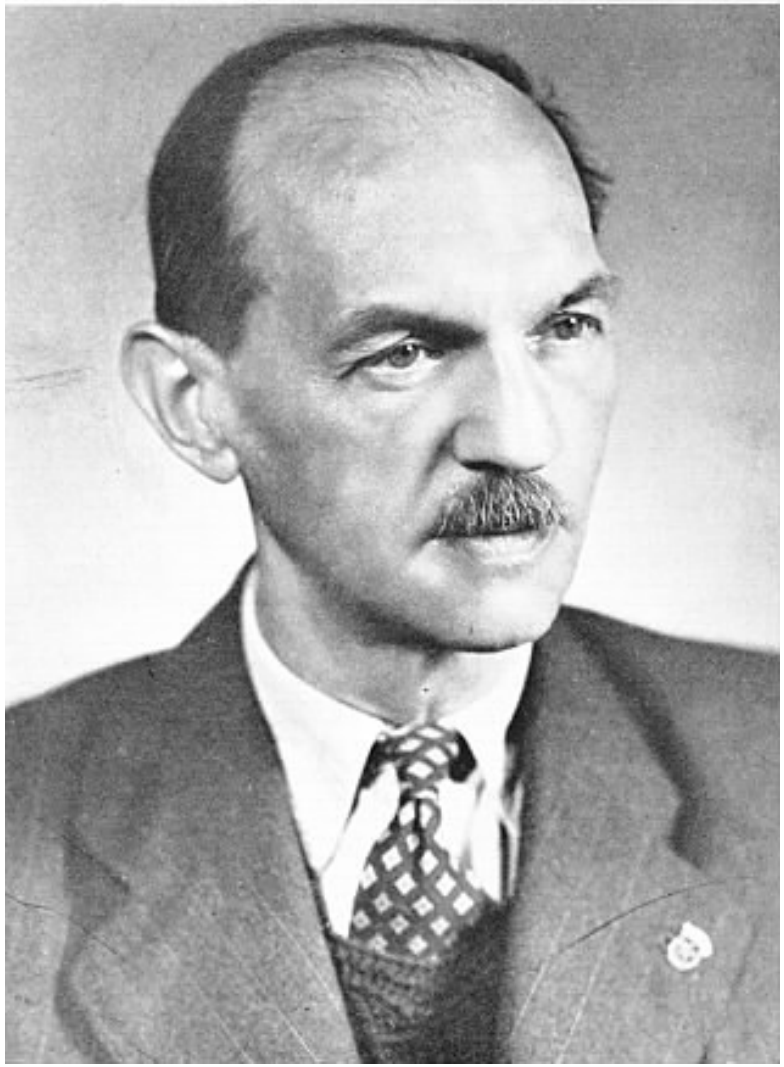
Prim's Algorithm

Vojtech Jarnik (1930)
Robert Prim (1957)
Edsger Dijkstra (1959)

Otakar Bůruvka
May 10, 1899 - July 22, 1995
Click Here for Biography

To many people Bůruvka is best known for his solution of the Minimal Spanning Tree problem which he published in 1926 in two papers *On a certain minimal problem* (Czech) and *Contribution to the solution of a problem of economical construction of electrical networks* (Czech):
There are n points in the plane whose mutual distances are different. The problem is to join them with a network in such a way that:

1. any two points are joined to each other either directly or by means of some other points.
2. the total length of the network will be minimal.



Vojtech Jarnik
Dec. 22, 1897 - Sept 22 1970
Click Here for Biography

Edsger Dijkstra
May 11, 1930 - Aug 6, 2002
Click Here for Biography

Robert C. Prim

Robert Clay Prim (born 1921 in Sweetwater, Texas) is an American mathematician and computer scientist. Prim received his B.S. in Electrical Engineering and Ph. D. in mathematics from Princeton University.. During the climax of World War II (1941-1944), Prim worked as an engineer for General Electric. From 1944 until 1949, he was hired by the United States Naval Ordnance Lab as an engineer and later a mathematician. At Bell Laboratories, he served as director of mathematics research. There, Prim developed Prim's algorithm. During his career at Bell Laboratories, Prim along with coworker Joseph Kruskal developed two different algorithms for finding a minimum spanning tree in a weighted graph, a basic stumbling block in computer network design .

Prim's Algorithm

Prim's Algorithm in One Sentence: Select any vertex and label it connected; then proceed to label as connected the unconnected vertex that is closest to a connected vertex.

More formally:

- Create a tree containing a single vertex, chosen arbitrarily from the graph
- Create a set containing all the edges in the graph
- Loop until every edge in the set connects two vertices in the tree
  - Remove from the set an edge with minimum weight that connects a vertex in the tree with a vertex not in the tree
  - Add that edge to the tree

1) This is our original tree. The numbers near the arcs indicate their weight. None of the arcs is highlighted, and vertex D has been arbitrarily chosen as a starting point.

2) The second chosen vertex is the vertex nearest to D: A is 5 away, B is 9, E is 15, and F is 6. Of these, 5 is the smallest, so we highlight the vertex A and the arc DA.

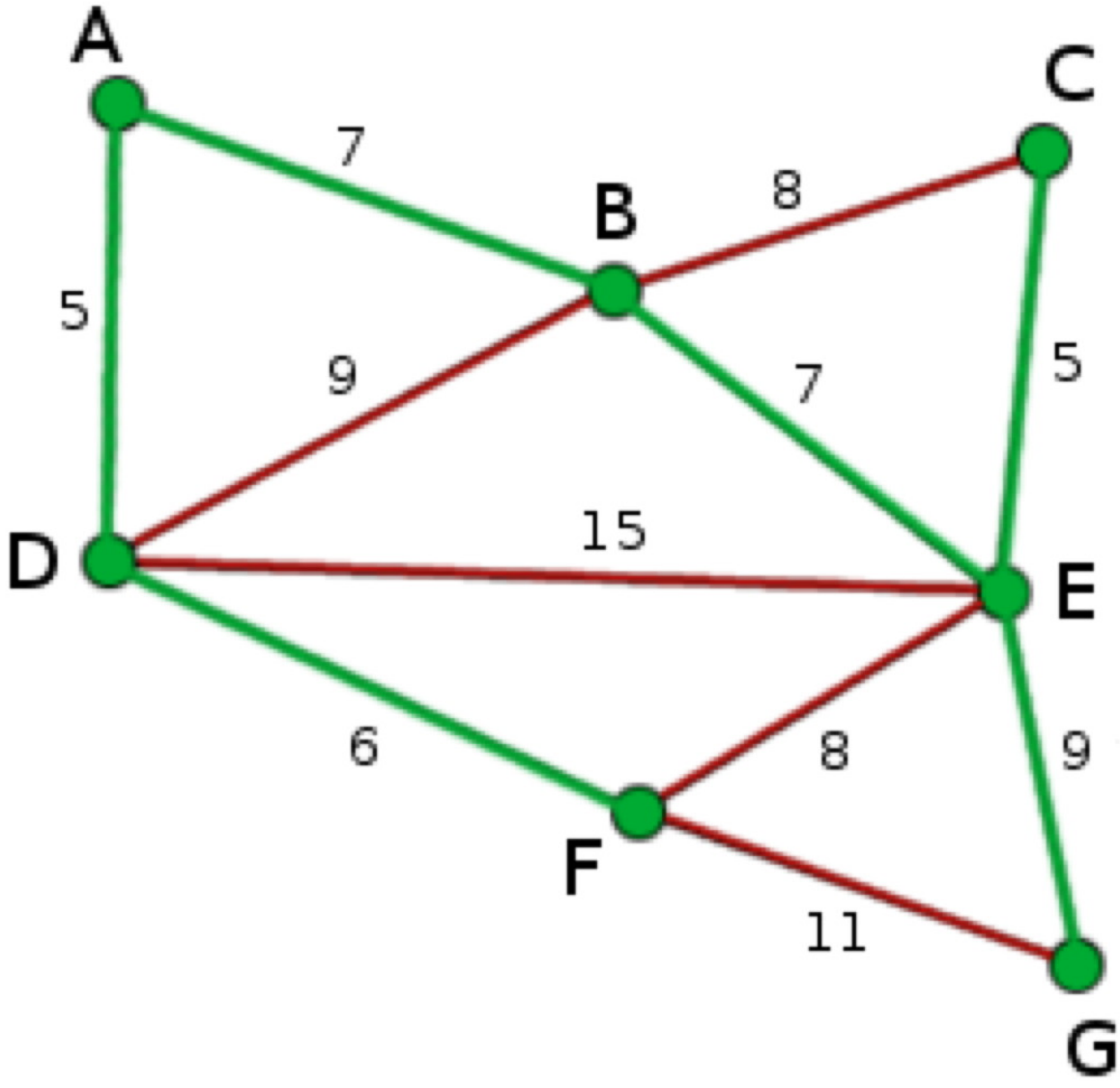
3) The next vertex chosen is the vertex nearest to either D or A. B is 9 away (from D), E is 15, and F is 6. 6 is the smallest, so we highlight the vertex F and the arc DF.

4) The algorithm carries on as above. Vertex B, which is 7 away from A, is highlighted. Here, the arc DB is highlighted in red, because both vertex B and vertex D have been highlighted, so it cannot be used.

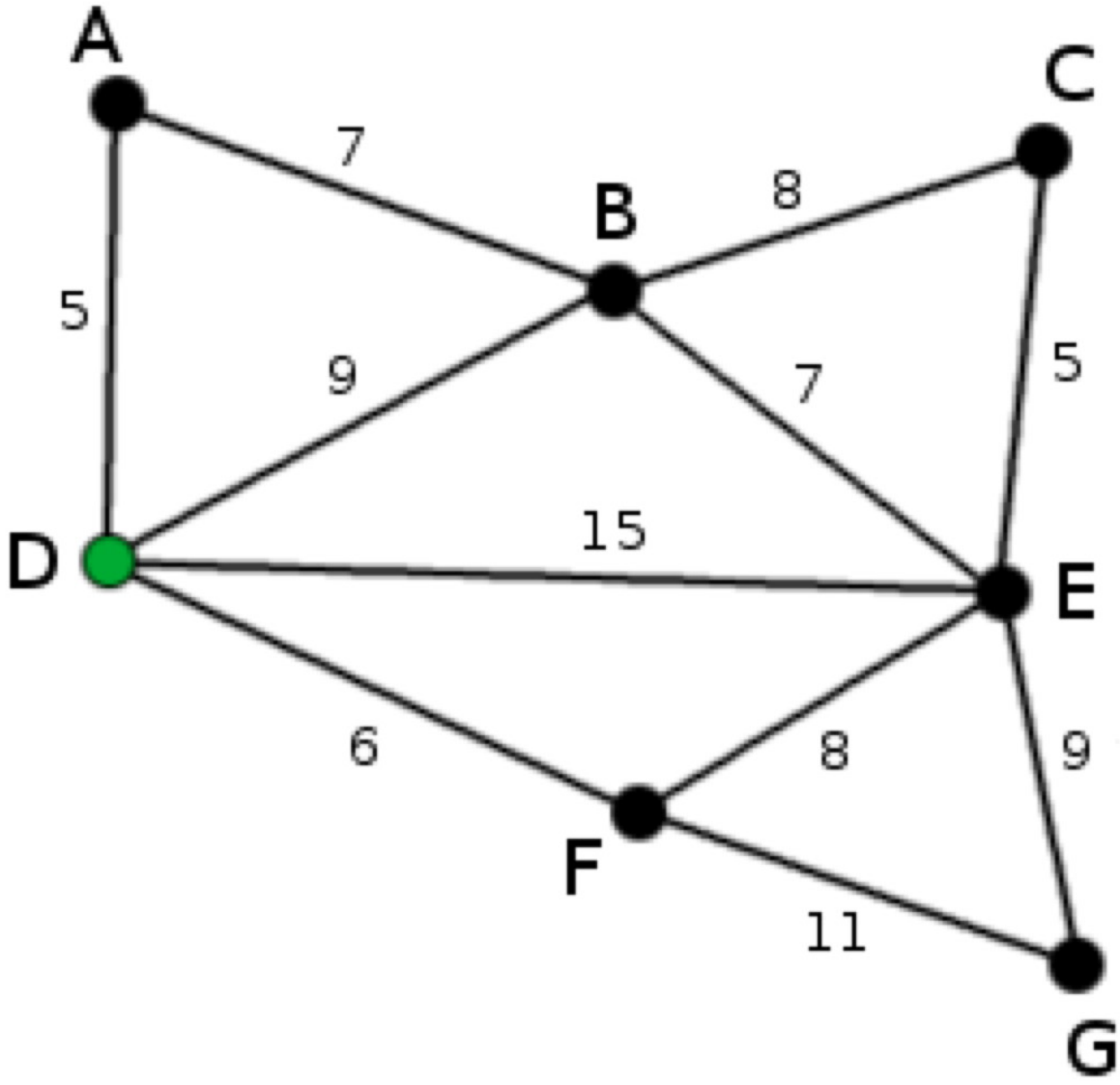
5) In this case, we can choose between C, E, and G. C is 8 away from B, E is 7 away from B, and G is 11 away from F. E is nearest, so we highlight the vertex E and the arc EB. Two other arcs have been highlighted in red, as both their joining vertices have been used.

6) Here, the only vertices available are C and G. C is 5 away from E, and G is 9 away from E. C is chosen, so it is highlighted along with the arc EC. The arc BC is also highlighted in red.

7) Vertex G is the only remaining vertex. It is 11 away from F, and 9 away from E. E is nearer, so we highlight it and the arc EG. Now all the vertices have been highlighted, the minimum spanning tree is shown in green. In this case, it has weight 39.



Vertices were added in the order D, A, F, B, E, C, G



Vertices are added in the order D, A, F, B, E, C, G

Proof that Prim's Algorithm Works

Here's a quick proof.
Suppose we have a connected, weighted graph; that is, there is a weight or cost $W(a, b)$ assigned to each edge (a, b). Let E^* be a subset of the edges in a Minimum Spanning Tree T. Let V^* be the vertices incident with edges in E^*.
If (x, y) is an edge of minimum weight such that x is in V^* and y is not in V^*, then we claim that adding the edge (x, y) to E^* gives a subset of a minimum spanning tree.
Why? If the edge is in T, this is trivial.
Suppose (x, y) is not in T.
Then there must be a path in T from x to y since T is connected.
If (v, w) is the first edge on this path with one vertex in V^*, if we delete it and replace it with (x, y) we get a spanning tree.
This tree must have smaller weight than T, since $W(v, w) > W(x, y)$. Thus T could not have been the Minimum Spanning Tree.

Here's a bit longer proof: Let P be a connected, weighted graph with $W(e)$ indicating the weight (or cost) of edge e. At every iteration of Prim's algorithm, an edge must be found that connects a vertex in a subgraph to a vertex outside the subgraph.
Since P is connected, there will always be a path to every vertex.
The output Y of Prim's algorithm is a tree, because the edge and vertex added to Y are connected to other vertices and edges of Y and at no iteration is a circuit created since each edge added connects two vertices in two disconnected sets.
Also, Y includes all vertices from P because Y is a tree with n vertices, same as P. Therefore, Y is a spanning tree for P.

Let Y1 be any minimal spanning tree for P. If Y = Y1, then the proof is complete. If not, there is an edge in Y that is not in Y1.

Let e be the first edge that was added when Y was constructed. Let V be the set of vertices of Y - e. Then one endpoint of e is in Y and another is not. Since Y1 is a spanning tree of P, there is a path in Y1 joining the two endpoints. As one travels along the path, one must encounter an edge f joining a vertex in V to one that is not in V. Now, at the iteration when e was added to Y, f could also have been added and it would be added instead of e if its weight was less than e. Since f was not added, we conclude that

$$W(f) \geq W(e).$$

Let Y2 be the tree obtained by removing f and adding e from Y1. Then Y2 is a tree that has more common with Y than with Y1. If Y2 equals Y, QED.

If not, we can find a tree, Y3, with one more edge in common with Y than Y2 and so forth.
Continuing this way produces a tree that has more in common with Y than with the preceding tree.
Since there are finite number of edges in Y, the sequence is finite, so there will eventually be a tree, Yh, which is identical to Y.
This shows Y is a minimal spanning tree.


ON AIR
And now a word for our sponsors...

Mathematics Courses
FALL 2023

201	Statistics and Data Science	MWF 9:45	
211	Regression	TuThF 8:40	Becky Tang
218	Statistical Learning	MWF 1:10	Alex Lyford
223A	Multivariable Calculus	MWF 8:40	Mike Olinick
223B	Multivariable Calculus	MWF 9:45	Mike Olinick
226	Differential Equations	MWF 1:10	Michaela Kubacki
302	Abstract Algebra	MWF 11:15	David Dorman
310	Probability	MWF 11:15	
323	Real Analysis	MWF 1:10	Steve Abbott
326	Partial Differential Equations	MW 9:45	Michaela Kubacki
332	Topology (CW)	MW 2:15	Mike Olinick
335	Differential Geometry	MWF 1:10	Emily Proctor
412	Bayesian Statistics	TuThF 11:15	Becky Tang
741	Number Theory Seminar	TuTh 9:45	Pete Schumer
746	Linear Algebra Methods	TuTh 11:15	John Schmitt


COURSES

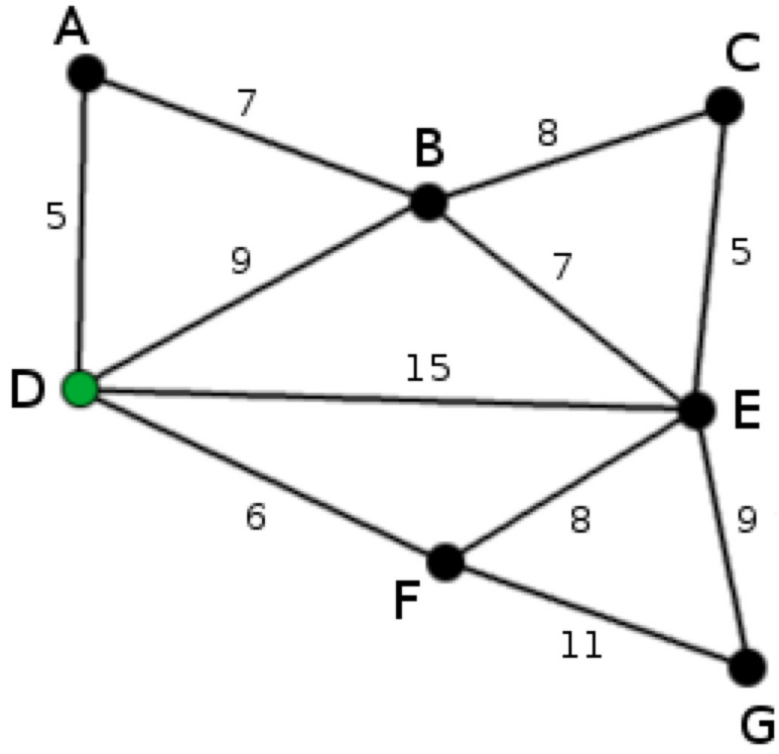
Kruskal's Algorithm

Kruskal's algorithm is an algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. Kruskal's algorithm is an example of a greedy algorithm.

Kruskal's Algorithm in One Sentence: At each stage, add the cheapest edge that connects two nodes not already connected

More formally,

- Create a forest F (a set of trees), where each vertex in the graph is a separate tree
- Create a set S containing all the edges in the graph
- while S is nonempty:
  - Remove an edge with minimum weight from S
  - If that edge connects two different trees, then add it to the forest, combining two trees into a single tree
  - Otherwise discard that edge



Kruskal adds edges in the order, CE, AD, DF, AB, BE, EG

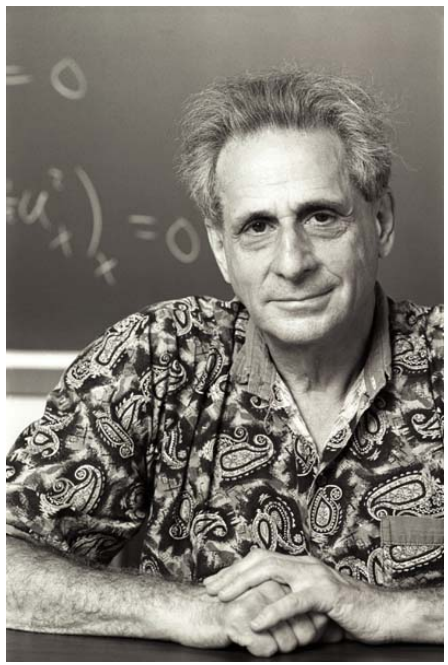
With the use of a suitable data structure, we can show that Kruskal's algorithm runs in $O(m \log n)$ time, where m is the number of edges in the graph and n is the number of vertices.

(January 29, 1928 - September 19, 2010)

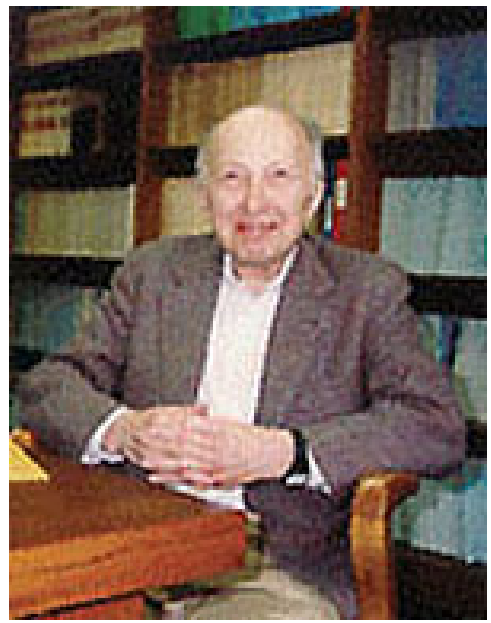
Joseph Bernard Kruskal (born in New York) was an American mathematician, statistician, and psychometrician. He was a student at the University of Chicago and at Princeton University, where he completed his Ph.D. in 1954.
Kruskal worked on well-quasi-orderings and multidimensional scaling. He was a Fellow of the American Statistical Association, former president of the Psychometric Society, and former president of the Classification Society of North America.

Kruskal also initiated and was first president of the Fair Housing Council of South Orange and Maplewood in 1963, and actively supported civil rights in several other organizations.

Joseph should not be confused with his two brothers Martin David Kruskal (September 28, 1925 - December 26, 2006) , co-inventor of solitons and of surreal numbers) and William Henry Kruskal (October 10, 1919 - April 21, 2005) who developed the Kruskal-Wallis one-way analysis of variance).



Martin Kruskal



William Kruskal

Proof That Kruskal's Algorithm Works

Let P be a connected, weighted graph and let Y be the subgraph of P produced by the algorithm. Y cannot have a cycle, since the last edge added to that cycle would have been within one subtree and not between two different trees. Y cannot be disconnected, since the first encountered edge that joins two components of Y would have been added by the algorithm. Thus, Y is a spanning tree of P.

Let Y1 be a minimum spanning tree for P which has the greatest number of edges in common with Y. If Y1=Y then Y is a minimum spanning tree. Otherwise, let e be the first edge considered by the algorithm that is in Y but not in Y1. Let C1 and C2 be the components of F which e lies between at the stage when e is considered. Since Y1 is a tree, Y1+e has a cycle and there is some different edge f of that cycle that also lies between C1 and C2. Then Y2=Y1+e-f is also a spanning tree. Since e is considered by the algorithm before f, the weight of e is at most equal to the weight of f, and since Y1 is a minimum spanning tree the weights of these two edges must in fact be equal. Therefore, Y2 is a minimum spanning tree having more edges in common with Y than Y1 does, contradicting our assumption about Y1. This proves that Y must be a minimum spanning tree.

Some References

J. B. Kruskal: On the shortest spanning subtree and the traveling salesman problem. In: Proceedings of the American Mathematical Society. 7 (1956), pp. 48-50

R. C. Prim: Shortest connection networks and some generalisations. In: Bell System Technical Journal, 36 (1957), pp. 1389-1401

D. Cherition and R. E. Tarjan: Finding minimum spanning trees. In: SIAM Journal of Computing, 5 (Dec. 1976), pp. 724-741

For Biography of Jarnik: http://www-groups.dcs.st-and.ac.uk/ history/Mathematicians/Jarnik.html

For more about Dijkstra: http://www.cs.utexas.edu/users/EWD/

Kruskal's recollection: www.emis.de/journals/AM/97-12/kruskal.ps

Shortest Path Problem

An Algorithm For The Shortest-Path Problem

We have a connected network with two special nodes designated the **Origin** and the **Destination** and a non-negative distance assigned to each link

.

Objective of the *n*th iteration: Find the nth nearest node to the **Origin**. Repeat for $n = 1,2,3,..$ until the nth nearest node is the **Destination**.

Input for the *n*th iteration: $n-1$ nearest nodes to the **Origin** solved for at the previous iterations, including their shortest path and distance from the **Origin**. These nodes, together with the **Origin**, will be called **solved nodes**; the others are **unsolved nodes.**

Candidates for the *n*th nearest node: Each solved node that is directly connected by a link to one or more unsolved nodes provides one candidate: the unsolved node with the **shortest** connecting link. (Ties provide additional candidates)

Calculation of the *n*th nearest node: For each solved node and its candidate, add the distance between them and the distance from the **Origin** to this solved node. The candidate with the smallest such total distance is the *n*th nearest node (ties provide additional solved nodes) and its shortest path is the one generating this distance.