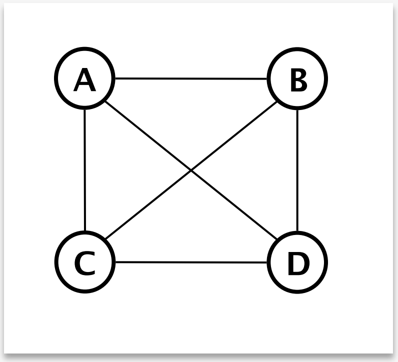
Minimum Spanning Trees

COMP 2210 – Dr. Hendrix

AUBURN
UNIVERSITY

SAMUEL GINN
COLLEGE OF ENGINEERING

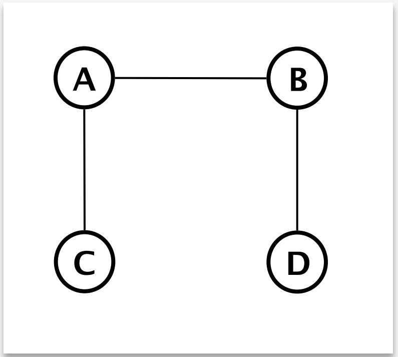
Spanning Trees

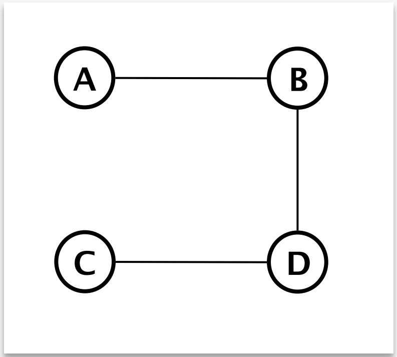


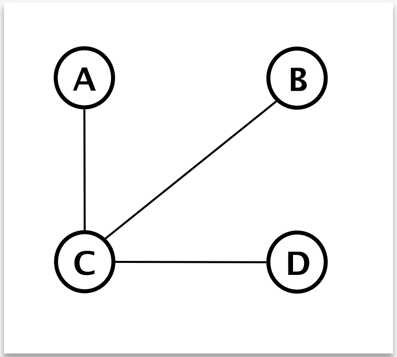
A **spanning tree** of a *connected, undirected* graph is a subgraph that contains

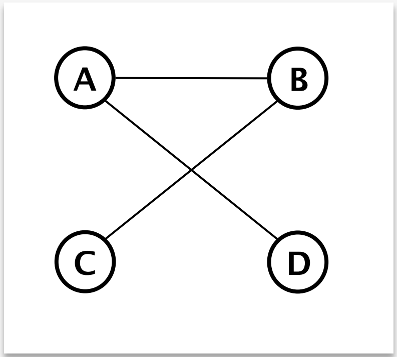
- all the vertices in the graph
- and the fewest number of edges such that the subgraph is connected

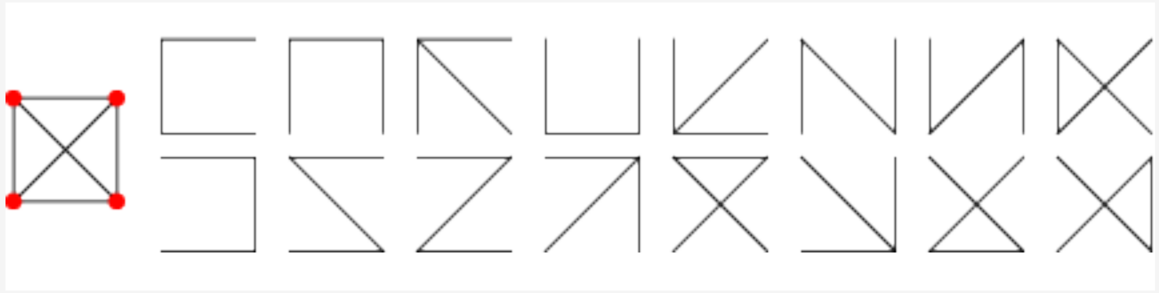
(contains no cycles)



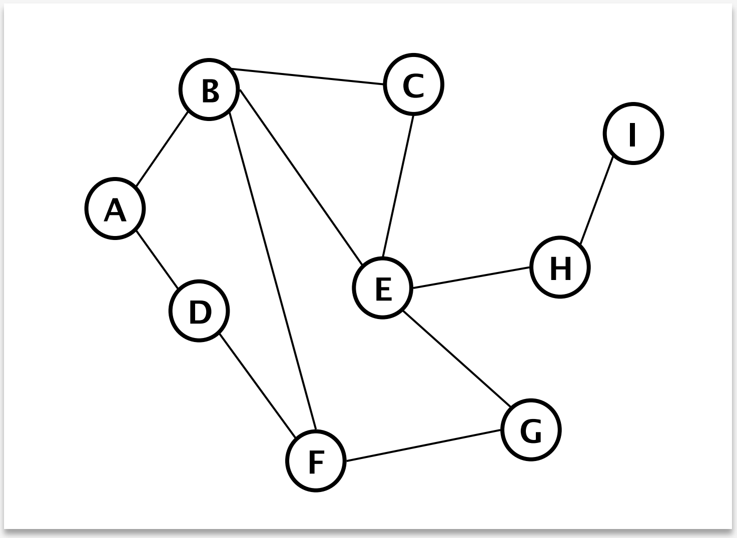








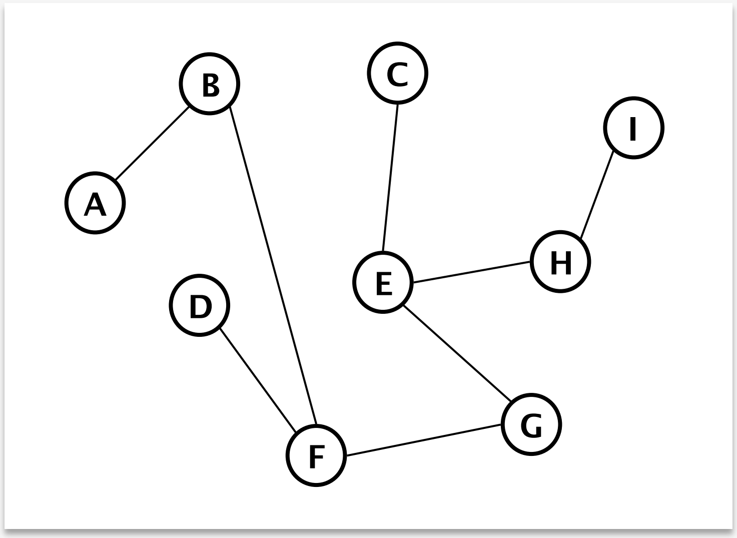
Constructing a spanning tree for a graph



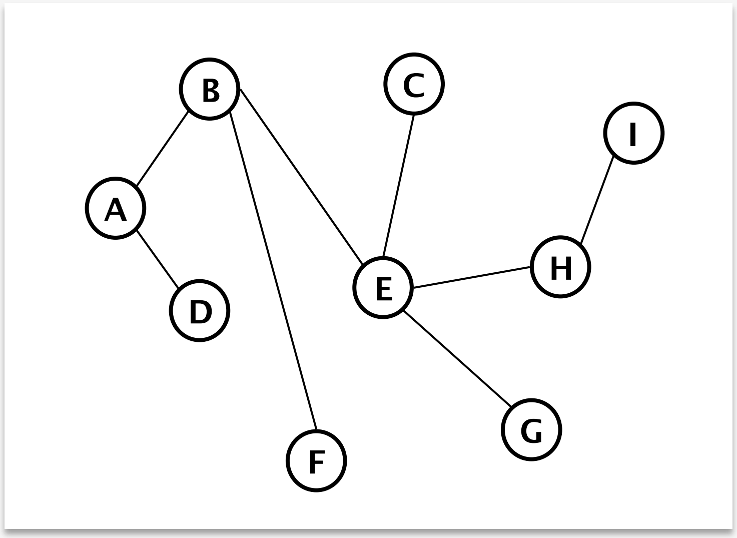
Use DFS (or BFS) and keep track of the edges crossed.

The set of crossed edges form a spanning tree.

DFS tree starting at A:



BFS tree starting at A:

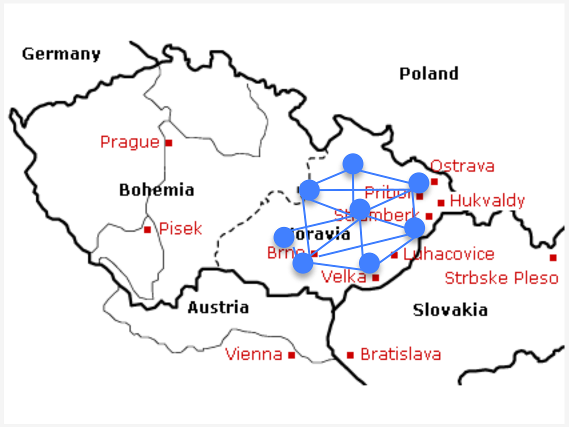


Minimum Spanning Trees

Otakar Borůvka

Czech mathematician

"On a Certain Minimal Problem" -1926

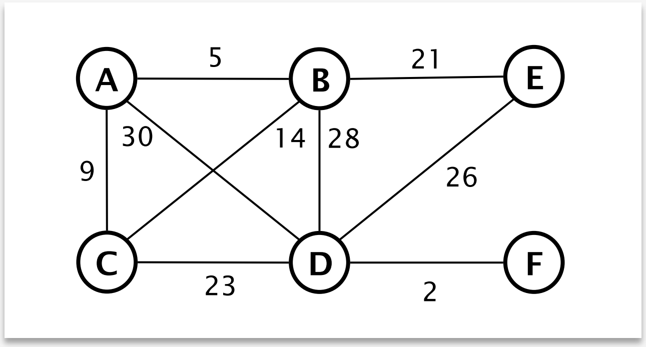


"Soon after the end of World War I, at the beginning of the 1920s, the Electric Power Company of Western Moravia, Brno, was engaged in rural electrification of Southern Moravia. In the framework of my friendly relations with some of their employees, I was asked to solve, from a mathematical standpoint, the question of the most economical construction of an electric power network. I succeeded in finding a construction – as it would be expressed today – of a maximal connected subgraph of minimum length, which I published in 1926."

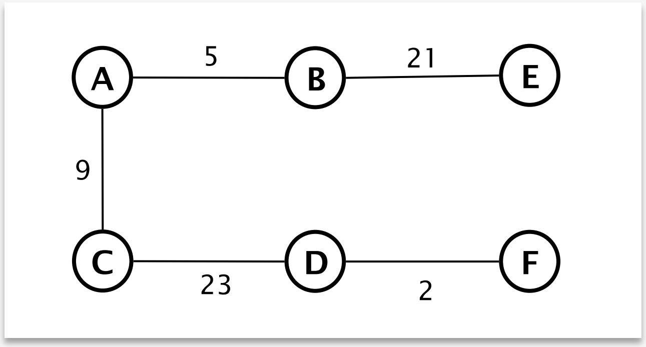
Minimum Spanning Trees

A **minimum spanning tree** of a connected, weighted, undirected graph is a spanning tree for the graph such that the sum of the edge weights is the minimum of all possible spanning trees for the graph.

Graph



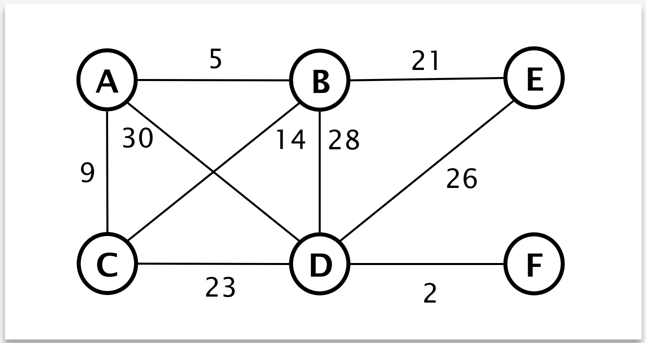
MST



Important Applications…

http://www.ics.uci.edu/~eppstein/gina/mst.html

How to construct a MST?

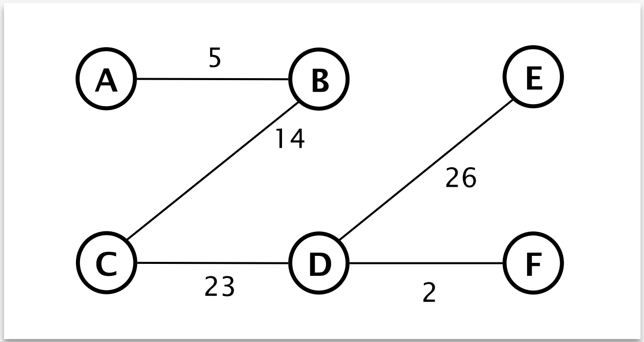


Use DFS and keep track of the edges crossed.

How to choose which neighbor to visit?

Choose the cheapest one that doesn't create a cycle. **The greedy choice**

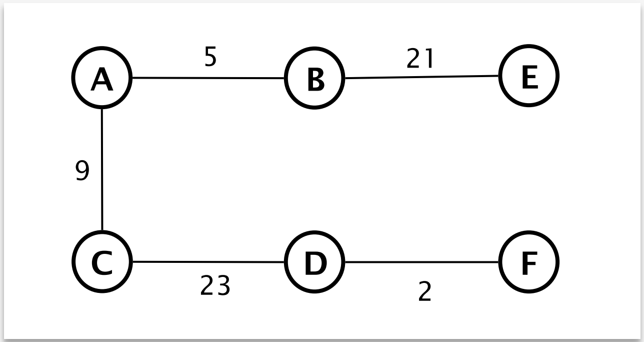
Resulting spanning tree:



Cost = 70

≠

MST (cost = 60)

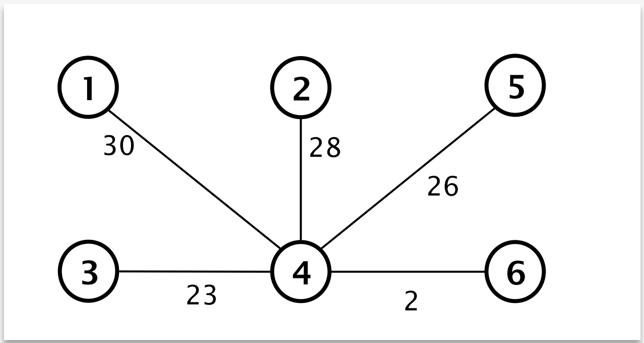


This strategy got us close, but not the optimal solution.

Can the greedy heuristic be used to find the optimal solution on this problem? **Yes…**

≠

Cost = 109



Greedy algorithms

Greedy algorithms are those that apply the following problem-solving heuristic when faced with choices:

Make locally optimal choices in hopes of arriving at a globally optimal solution

This is a heuristic, so it isn't guaranteed to always work. It can even produce the worst possible solution.

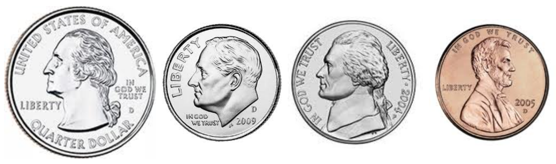
The greedy heuristic is most often applied to **optimization problems**.

Example: Arrive at a given monetary amount using the smallest possible number of coins. (Making change)

US dollar, all coins

dollar, quarter, dime, nickel, penny

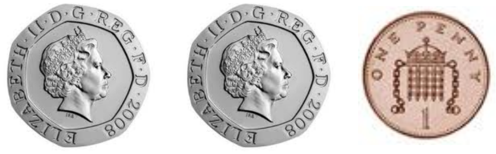
41¢



GBP, all coins

£5, £2, £1, 50p, 20p, 10p, 5p, 2p, 1p

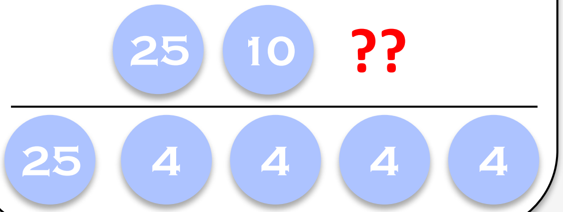
41p



Atlantis $, all coins

25c, 10c, 4c

41c



Greedy algorithms

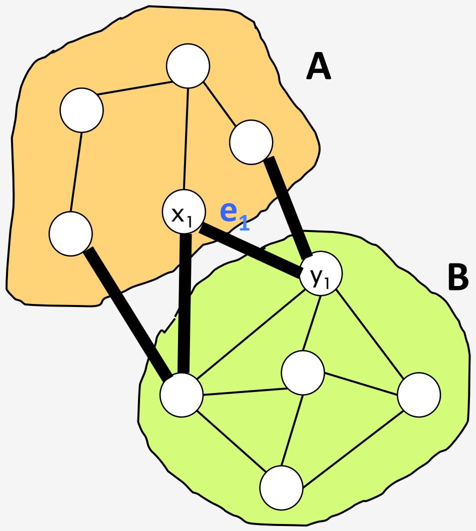
The greedy heuristic isn't always applicable and, even when it is, it isn't guaranteed to find an optimal solution.

For example, if the Atlantis coin set were 10c, 7c, and 1c, and we were trying to make change for 15c, the greedy choice would select one 10c and five 1c coins. But two 7c and one 1c would be better.

There is no general way of knowing whether or not the greedy heuristic is applicable or will lead to an optimal solution, but the problem should exhibit the **greedy choice property** and the **optimal substructure property**. More later …

Important property of the MST problem:

Given any division of the vertices of a graph into two sets, the minimum spanning tree contains the minimum cost edge that connects a vertex in one set to a vertex in the other set.



Proof: Call the minimum cost edge connecting the two sets A and B e1, and assume that e1 is not in the MST. Then consider the graph formed by adding e1 to the purported MST. This graph has a cycle; in that cycle some other edge besides e1 must connect A and B. Deleting this edge and adding e1 gives a lesser cost spanning tree, which contradicts the assumption that e1 is not in the MST.

Prim's Algorithm *(aka Jarnik's Algorithm or the DJP Algorithm)*

Rather than selecting the cheapest neighbor to a certain vertex (e.g., the current vertex on a DFS), this algorithm selects the *cheapest neighbor to any vertex already part of the MST*. **Greedy**

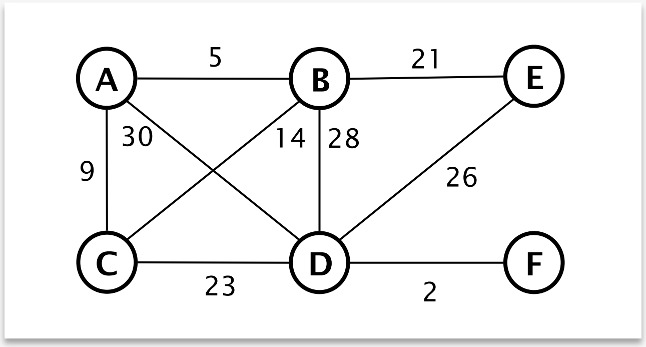
```
Initialize the MST as an empty graph.
Select any vertex and add it to the MST.
Add all the edges in the graph to a collection E.
while (still more edges in E) &&
      (have not added n-1 edges to the MST)
{
     Remove the edge e (x, y) from E such that
          - x is already in the MST.
          - e has minimum cost
     Add e and y to the MST if this doesn't create a cycle.
     }
}
```

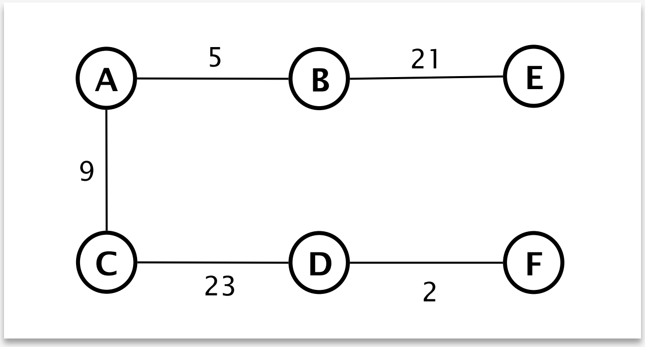
Min heap

Makes handling the min heap tricky. Better to build the min heap incrementally.

How do we know if an edge (x, y) creates a cycle?

y in MST => cycle
y not in MST => no cycle





Kruskal's Algorithm

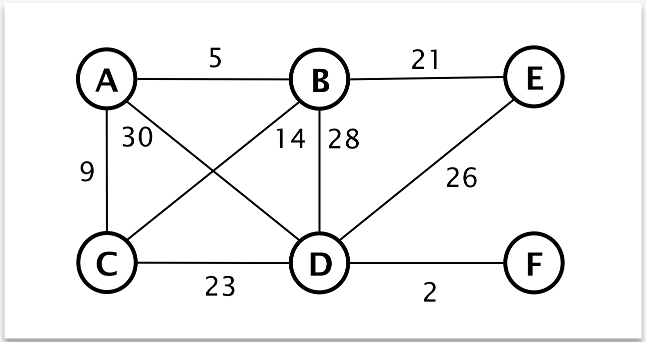
Sort the edges in ascending order of weight. Starting with the cheapest edge, add each edge to the MST unless it would create a cycle. Stop when n-1 edges have been added.

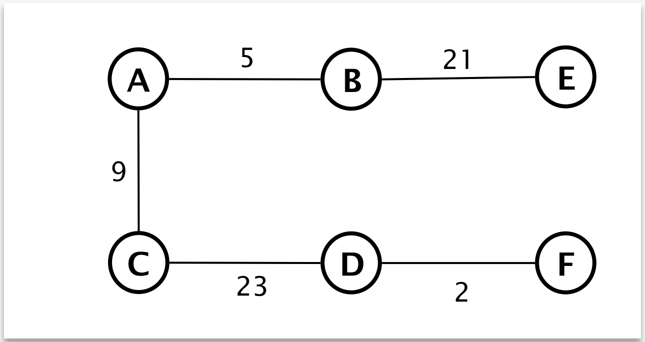
Greedy

```
Initialize MST with all vertices but no edges
Add all the edges to a collection E
while (still more edges in E)
     &&
     (have not added n-1 edges to the MST)
{
     Remove the edge with minimum cost from E.
     Add it to the MST if it does not create a cycle.
}
```

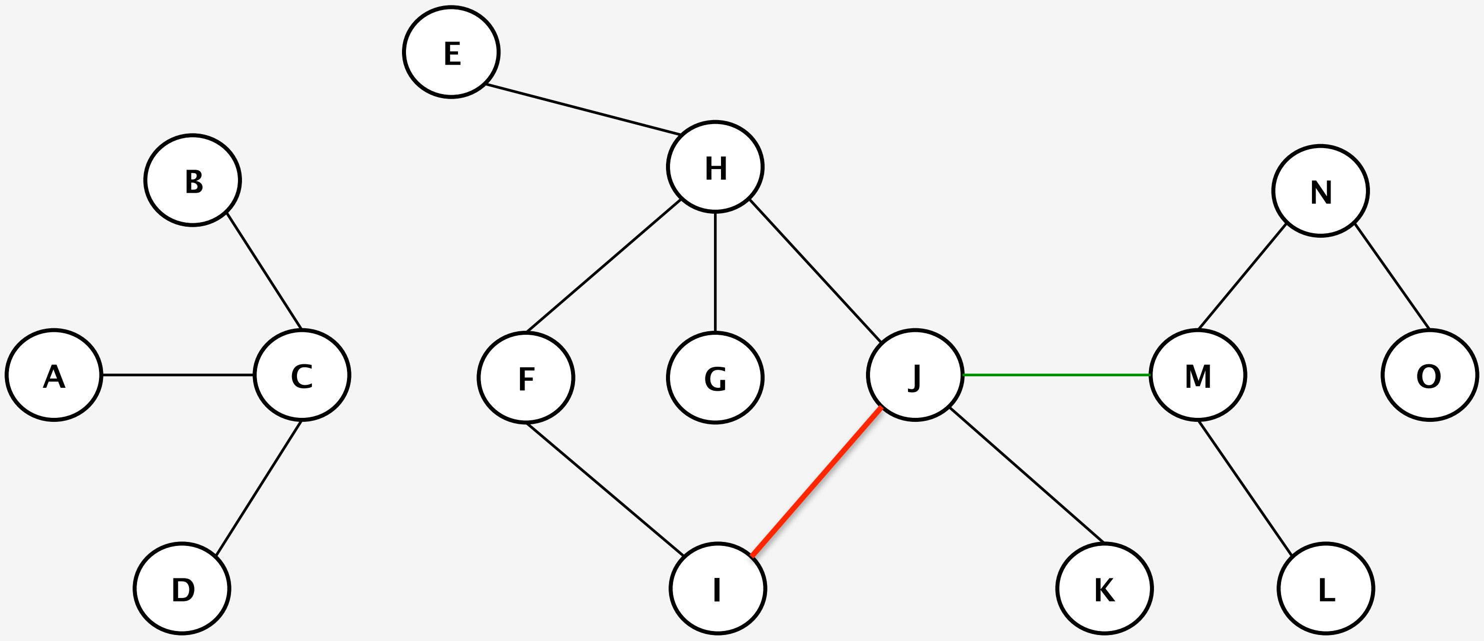
Min heap

How do we know if an edge (x, y) creates a cycle?





Connected components and cycles



```
Add edge (x, y) if doesn't create a cycle
c1 = findComponent(x);
c2 = findComponent(y);
if (x != y)
{
   add (x,y) to graph;
   union(c1, c2);
}
```

Add (I, J)? **No**

c1 = green
c2 = green

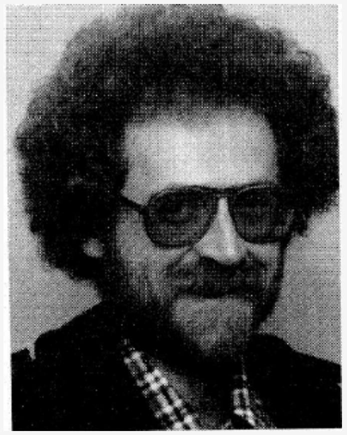
Add (M, J)? **Yes**

c1 = blue
c2 = green

References

1. http://en.wikipedia.org/wiki/Spanning_tree
2. http://mathworld.wolfram.com/SpanningTree.html
3. http://mathworld.wolfram.com/SpanningTree.html (image use)
4. http://algs4.cs.princeton.edu/43mst/

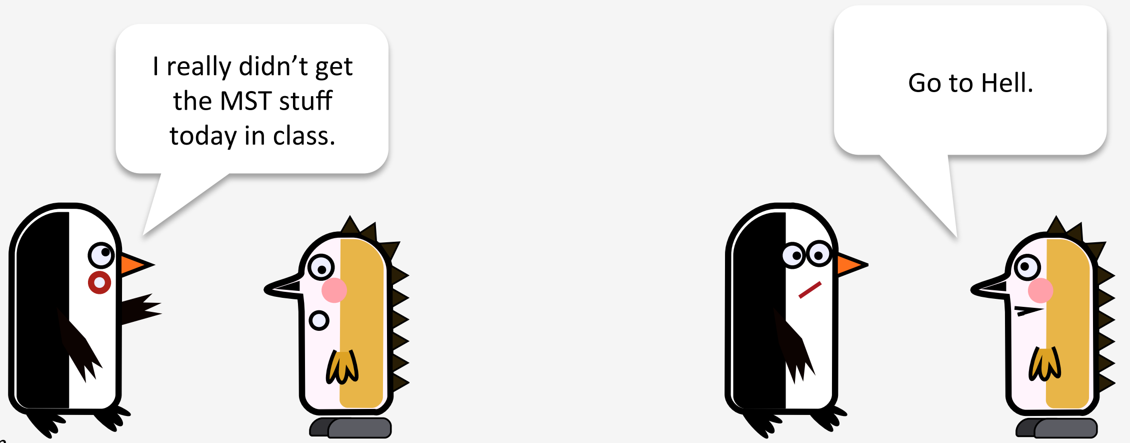
and …



Dr. Hell

http://www.cs.sfu.ca/~pavol/

On the history of the minimum spanning tree problem (Pavol Hell, with Ron L. Graham), *Annals of the History of Computing* 7 (1985) 43 – 57.



Original character art by Esa Holopainen.