GREEDY ALGORITHMS
CSCE 340 SUPPLEMENTAL LECTURE NOTES

DR. GODFREY C. MUGANDA

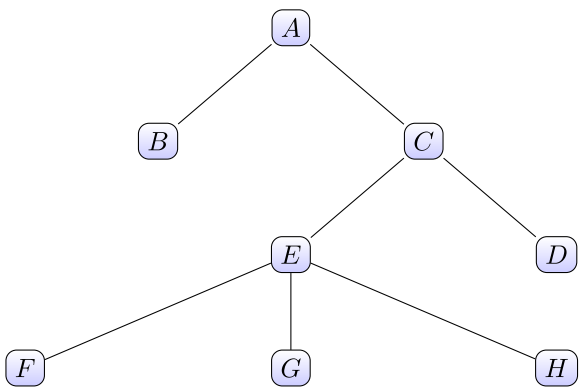
1. Introduction

We want to look at the problem of finding *minimum spanning trees* in a weighted graph, and we want to look at how *greedy algorithms* can be used to solve that type of problem.

2. Trees

To understand what a spanning tree is, we need to understand the concept of a tree.

All of us are familiar with the the concept of binary trees, but here is a more general example of a tree:



Basically, a *tree* is just a connected graph with no cycles.

Two important characteristics of a tree are:

(1) connected: Given any two vertices X and Y, there is a path from X to Y.
(2) no cycles: Given any two vertices X and Y, there is only *one* a path from X to Y.

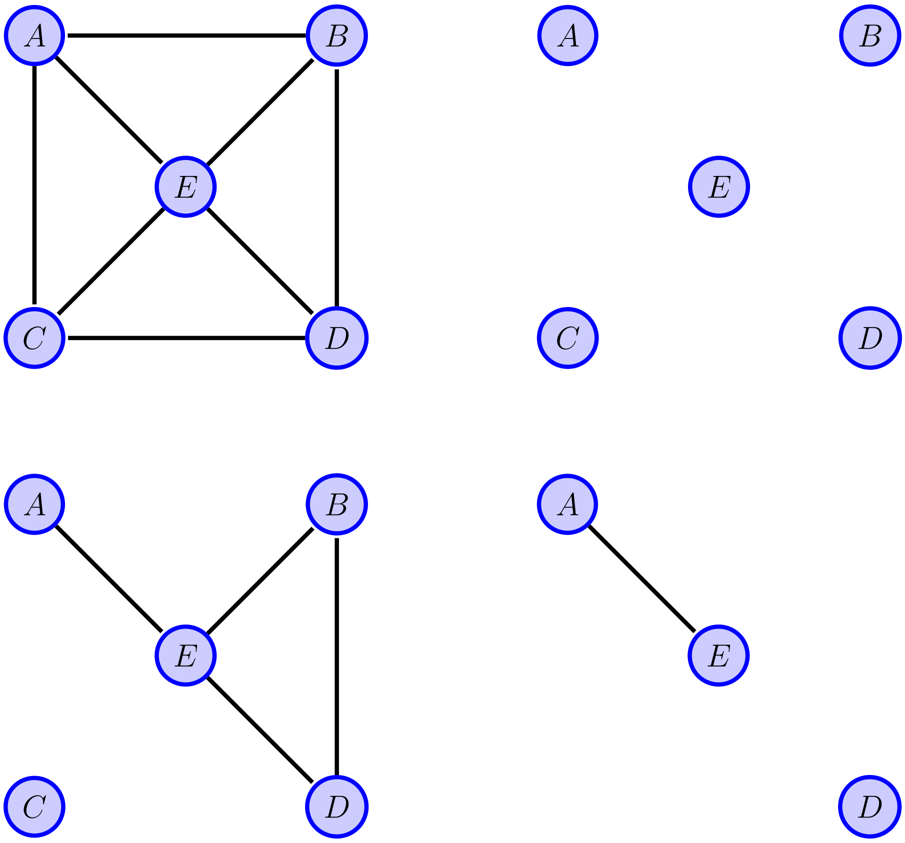
If you have a tree and you add an edge, you get a unique cycle.

If you subsequently delete any edge on that unique cycle, you get a tree(possibly a different tree).

3. Subgraphs

Given a graph $G = (V, E)$ with vertex set V and edge set E, a subgraph H of G is another graph whose vertex set is a subset of V, and whose edge set is a subset of E.

Here is an example of a graph G and three of its subgraphs.



The first two subgraphs are *spanning subgraphs*, meaning they have the same vertex set as the original graph.

The last subgraph is not spanning, because it leaves out some of the original vertices.

4. Spanning Trees

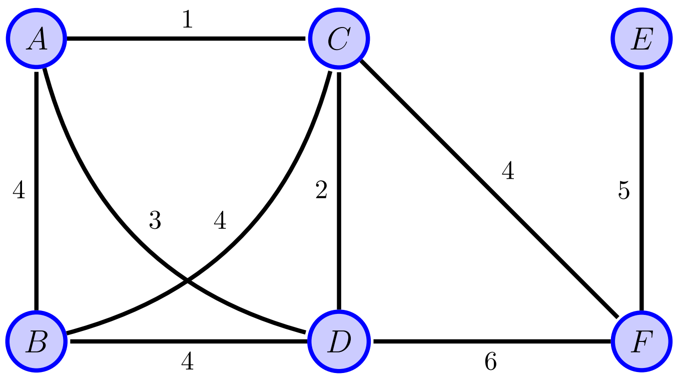
A spanning subgraph that is also a tree is called a *spanning tree*.

Thus a spanning tree is a connected spanning subgraph that has no cycles.

5. Weighted Graphs

Now suppose that we have a weighted graph. Recall that a weighted graph is just a graph where each edge is assigned a real number called the *weight* of the edge.

Here is an example of a weighted graph from page 127 of the class text:



The weights of the edges on the graph are costs associated with the maintenance of those edges. So for example, the graph you see above could be

(1) a transportation network connecting cities A through F in a state such as Illinois
(2) edges are highways connecting the cities,
(3) the weight of each edge in the annual cost of keeping that highway free of potholes, say in millions of dollars.

If the cost of the maintaining all the edges is too high, you might want to drop some of the edges, that is, stop maintaining some of the highways.

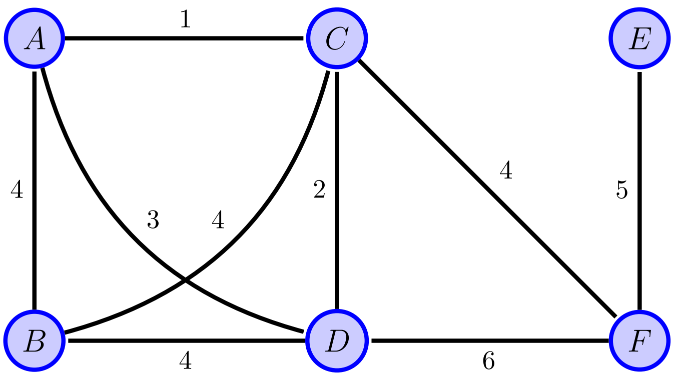
By dropping some edges, you get a subgraph. You do not want to drop any cities, so you want to a spanning subgraph of minimum weight.

But a spanning subgraph of minimum weight cannot have a cycle, so you really want a spanning tree of minimum weight.

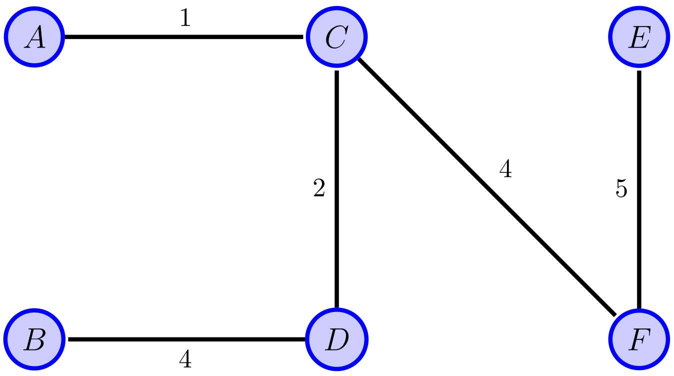
This is the problem of finding a minimum weight spanning tree in a weighted graph.

6. Example Minimum Spanning Tree

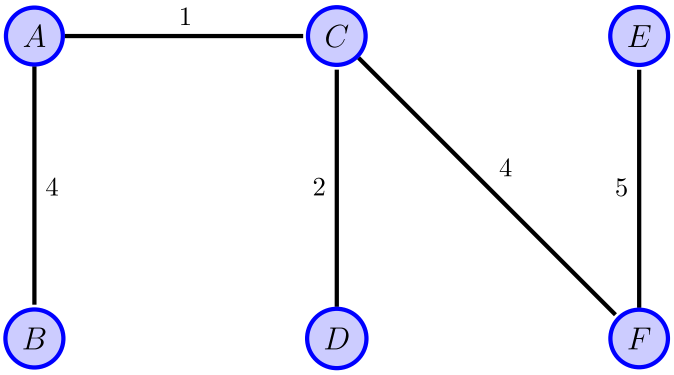
Given the weighted graph



Here is a minimum spanning tree with weight 16:



Here is another minimum spanning tree also with weight 16:



For a given weighted graph, there can be many minimum spanning trees.

7. Greedy Algorithms for Minimimum Spanning Trees

Greedy algorithms are used to solve *optimization* problems.

An optimization problem is a problem where you want to build a solution that either minimizes some cost or maximizes some profit.

Greedy algorithms are applicable where each solution is made up of parts, and each part contributes to the total cost or total profit.

For example, in the case of a minimum spanning tree (MST), each MST is made up of edges and each edge contributes a cost to the spanning tree.

In a minimization problem, the greedy method builds the solution be adding one part at a time, at each stage always adding the part that is consistent with parts already added, and contributes the least to the cost.

Greedy algorithms do not always work: they require proof that they work.

8. Prim's algorithm for MSTs

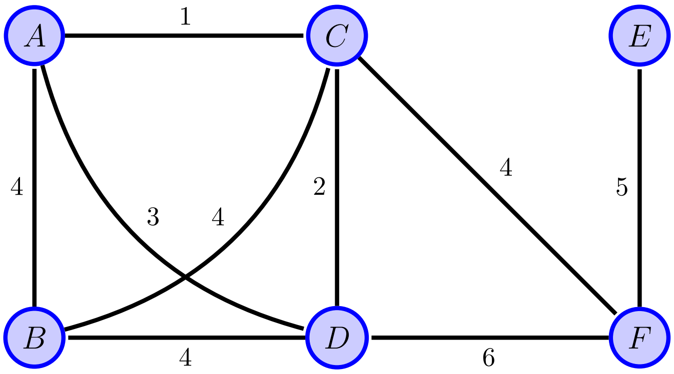
Prim's algorithm is a greedy algorithm for finding a minimum spanning tree in a weighted graph.

Main ideas behing Prim's algorithm:

(1) Every vertex is a part of of every MST, so you can start building an MST at any vertex. That single vertex constitutes a tree T that is included in some MST.
(2) If you already have a tree T that is included in some MST, extend T by adding an edge e that has exactly one endpoint in T. That means that $T + e$ is connected and has no cycle, so $T + e$ is still a tree.
(3) The greedy step: in picking the edge e, pick the edge of minimum weight among all edges that have exactly one end point in T.

Here is an example

Given the weighted graph

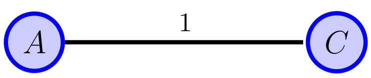


Start with a tree T consisting of just the vertex A:

The greedy step: Look at all edges that that have exactly one endpoint in $T = A$ and pick the edge of minimum weight. Choices are

(1) AB with weight 4
(2) AD with weight 3
(3) AC with weight 1.

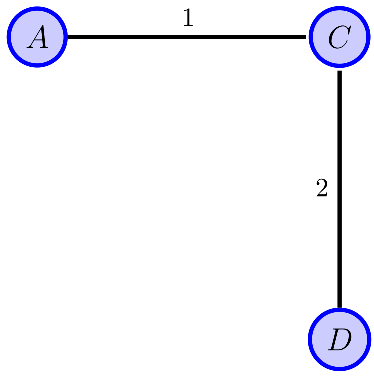
The greedy step selects AC.



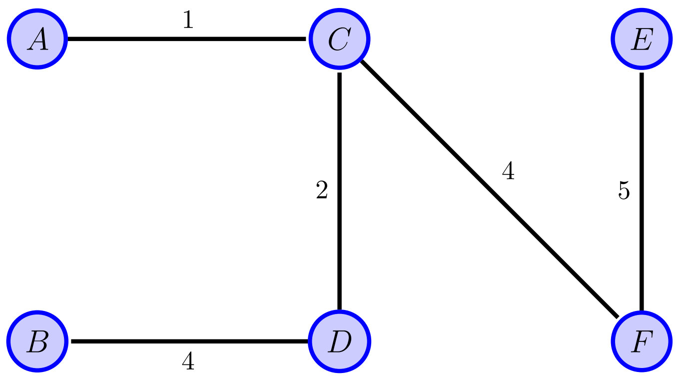
Next, you look at all edges that have exactly one endpoint in the tree consisting of the single edge AC. The choices are

(1) AB with weight 4
(2) AD with weight 3
(3) CB with weight 4
(4) CD with weight 2
(5) CF with weight 4

The greedy step selects CD, and the new T becomes $AC + CD$:



So the algorithm continues in this manner until it has a minimum spanning tree that reaches every vertex.



9. Correctness of Prim's Algorithm

How do we know Prim's algorithm is correct?

Suppose that at any stage, we have a tree T that is included in some MST M. This is certainly the case at the beginning when T consists of only one vertex.

So we know

$$T \subseteq M.$$

The greedy step: among all edges that that have exactly one endpoint in T, pick an edge e of minimum weight and add it to T. Then

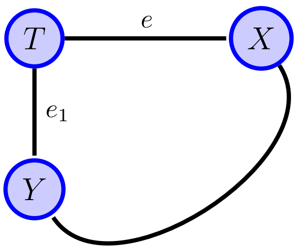
$$T + e \subseteq M + e.$$

Now $T + e$ is still a tree, but M was already a mimimum spanning tree, so now

$$M + e$$

has a unique cycle that contains the edge e.

Suppose that the endpoint of the edge e that is not in T is X. The unique cycle in $M + e$ goes from T across e to X, and eventually must come back to connect with the endpoint of e that is in T:



To do this, the cycle must go through another edge e_1 that also has just one endpoint in T. By the greedy step,

$$\text{weight}(e) \leq \text{weight}(e_1)$$

By deleting e_1 from $M + e$, we eliminate the unique cycle so that $M + e - e_1$ is now a spanning tree again, and

$$T + e \subseteq M + e - e_1.$$

But the weight of $M + e - e_1$ is less or equal to the weight of M, which means that $M + e - e_1$ is also an minimum spanning tree.

Thus we have established that if T is included in some MST, then after the greedy step, $T + e$ is also included in some MST.

Now each greedy step adds one new vertex to T, so eventually, T will contain all vertices and itself be a spanning tree that is contained in a MST. That makes T itself a minimum spanning tree.

10. Prim's algorithm

Let G be a weighted graph of n vertices. Here is Prim's algorithm for building a minimum spanning tree.

let T consist of any single vertex of the graph
repeat $n - 1$ times
    let e be an edge with exactly one endpoint in T whose weight is minimum
    $T = T + e$
end repeat

11. Kruskal's algorithm

There is another greedy algorithm for finding a MST.

The two algorithms can be compared to the two divide and conquer algorithms Quicksort and Mergesort in the similarity and differences.

It is the same idea of greed, it is just approached differently.