Advanced Graph Theory

Saketh Are

October 7, 2011

1. Minimal Spanning Trees

Given a connected graph G with weighted edges, we wish to remove edges so that the graph has minimum total weight possible but remains connected.

More formally, suppose E is the set of edges in G. Then we want to find an acyclic subset of the edges that connects all of the vertices and whose total weight is minimized. This set of edges is acyclic and contains all the vertices, so it must be a tree.

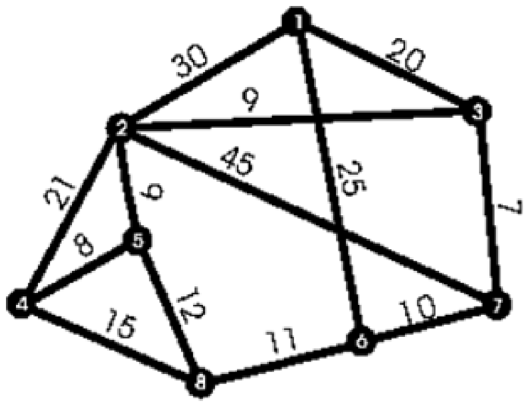

Figure 1: Our original graph G.

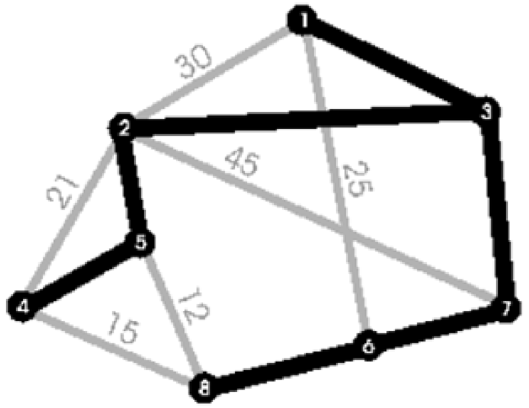

Figure 2: One minimum spanning tree.

1.1 Prim's Algorithm

Prim’s algorithm is one way to find the MST of a graph.

We start with any one node of our graph G. At each step, we find the node not yet included on our tree that will be “cheapest” to connect.

We can implement this using a priority queue that will give us the closest nodes in order. Notice that this algorithm is a very slightly modified form of Dijkstra’s algorithm, implying that our complexity is O(E log V).

1.2 Kruskal's Algorithm

Implementation of Kruskal’s Algorithm involves what are known as disjoint-set data structures, which will be the topic of a future lecture. For now, we will simply be looking at the main idea behind Kruskal’s Algorithm.

We iterate through a sorted list of edges, taking all edges in that order which do not form a cycle. Cycle-checking can be done very quickly with a disjoint-set structure,
leading to an overall runtime of O(E log E) due to the sort.

2. Shortest-Path Algorithms

Most of you are familiar with some of the more common shortest-path algorithms such as Floyd-Warshall or Dijkstra. But a few more exotic ones exist for some specific situations.

3.1 Relaxation

First, let’s review the concept of relaxation. When we’re running a shortest-path algorithm on a graph, we’ll record the shortest path found so far to each node. Relaxing an edge (u; v) simply means that d(v) will be replaced by d(u) + w(u; v) if appropriate.

3.2 Shortest Paths in a DAG

Directed acyclic graphs are nice because the shortest paths from a source are very easy to compute in linear time. First, if this is not already the case, we topologically sort the DAG. Then, we make just one pass over the vertices in the topological order, relaxing the edges that leave the vertex we're on as we go.

3.3 Bellman-Ford

We use the Bellman-Ford algorithm when we want to find shortest paths in a graph where some edges may have negative weight or determine if a graph contains a negative cycle. This algorithm is also somewhat unusual in that we take an edge list for input. Note that a shortest path will not contain more than V - 1 edges in the graph, assuming there is no negative cycle. So, we simply relax every edge V-1 times. This will run in O(EV).

4 Problems

1. Suppose the graph for Prim's algorithm is represented as an adjacency matrix. Give a simple implementation of Prim's algorithm that runs in $O(V^2)$ time.

2. A minimum bottleneck spanning tree is a spanning tree which minimizes the maximum edge weight over all such trees. Find one such minimum bottleneck spanning tree in a graph G.

3. Give an algorithm for finding the spanning tree with the smallest product of edge weights, assuming all edge weights are positive.

4. Given a graph G and spanning tree T. Suppose we decrease the weight of one of the edges not in T. Find the minimum spanning tree in the modified graph.

5. Bessie the cow is walking through some very corrupt cities in Zimbabwe. The cities are connected by a set of roads, and each road has some bandit associated with it. Some bandits are stronger than her, and will require a specific fee $-X_i$ for one safe passage. Other bandits are weaker than Bessie, so she can force them to give her up to X_i amount of money each time she traverses the road. Please help her compute the maximum amount of money she can obtain on her journey from Harare to Kwekwe.

6. (IOI 2003) Farmer John's cows wish to travel freely among the $N (1 \leq N \leq 200)$ fields (numbered $1..N$) on the farm, even though the fields are separated by forest. The cows wish to maintain trails between pairs of fields so that they can travel from any field to any other field using the maintained trails. Cows may travel along a maintained trail in either direction. The cows do not build trails. Instead, they maintain wild animal trails that they have discovered. On any week, they can choose to maintain any or all of the wild animal trails they know about. Always curious, the cows discover one new wild animal trail at the beginning of each week. They must then decide the set of trails to maintain for that week so that they can travel from any field to any other field. Cows can only use trails which they are currently maintaining. The cows always want to minimize the total length of trail they must maintain. The cows can choose to maintain any subset of the wild animal trails they know about, regardless of which trails were maintained the previous week. Wild animal trails (even when maintained) are never straight. Two trails that connect the same two fields might have different lengths. While two trails might cross, cows are so focused, they refuse to switch trails except when they are in a field. At the beginning of each week, the cows will describe the wild animal trail they discovered. Your program must then output the minimum total length of trail the cows must maintain that week so that they can travel from any field to any other field, if there exists such a set of trails.

7. (APIO 2008) The Kingdom of New Asia contains N villages connected by M roads. Some roads are made of cobblestones, and others are made of concrete. Keeping roads free-of-charge needs a lot of money, and it seems impossible for the Kingdom to maintain every road. *A* new road maintaining plan is needed. The King has decided that the Kingdom will keep as few free roads as possible, but every two distinct villages must be connected by one and only one path of free roads. Also, although concrete roads are more suitable for modern traffic, the King thinks walking on cobblestones is interesting. As a result, he decides that exactly K cobblestone roads will be kept free. Given a description of roads in New Asia and the number of cobblestone roads that the King wants to keep free, write a program to determine if there is a road maintaining plan that satisfies the King's criteria, and output a valid plan if there is one.

8. (David Benjamin and Jacob Steinhardt, 2008) Farmer John has grown so lazy that he no longer wants to continue maintaining the cow paths that currently provide a way to visit each of his $N\,(5 \leq N \leq 10,000)$ pastures (conveniently numbered $1..N$). Each and every pasture is home to one cow. FJ plans to remove as many of the $P\,(N-1 \leq P \leq 100,000)$ paths as possible while keeping the pastures connected. You must determine which $N-1$ paths to keep. Bidirectional path j connects pastures S_j and $E_j\,(1 \leq S_j \leq N; 1 \leq E_j \leq N; S_j \neq E_j)$ and requires $L_j\,(0 \leq L_j \leq 1,000)$ time to traverse. No pair of pastures is directly connected by more than one path. The cows are sad that their transportation system is being reduced. You must visit each cow at least once every day to cheer her up. Every time you visit pasture i (even if you're just traveling through), you must talk to the cow for time $C_i\,(1 \leq C_i \leq 1,000)$. You will spend each night in the same pasture (which you will choose) until the cows have recovered from their sadness. You will end up talking to the cow in the sleeping pasture at least in the morning when you wake up and in the evening after you have returned to sleep. Assuming that Farmer John follows your suggestions of which paths to keep and you pick the optimal pasture to sleep in, determine the minimal amount of time it will take you to visit each cow at least once in a day.

9. (Andre Kessler, 2009) Farmer John's cows are living on $N\,(2 \leq N \leq 200,000)$ different pastures conveniently numbered $1..N$. Exactly $N-1$ bidirectional cow paths (of unit length) connect these pastures in various ways, and each pasture is reachable from any other cow pasture by traversing one or more of these paths. The cows have organized $K\,(1 \leq K \leq N/2)$ different political parties conveniently numbered $1..K$. Each cow identifies with a single political party; cow i identifies with political party $A_i\,(1 \leq A_i \leq K)$. Each political party includes at least two cows. The political parties are feuding and would like to know how much 'range' each party covers. The range of a party is the largest distance between any two cows within that party (over cow paths). Please help the cows determine party ranges.

10. (IOI 2002) Yong-In city plans to build a bus network with N bus stops. Each bus stop is at a street corner. Yong-In is a modern city, so its map is a grid of square blocks of equal size. Two of these bus stops are to be selected as hubs H_1 and H_2. The hubs will be connected to each other by a direct bus line and each of the remaining $N-2$ bus stops will be connected directly to either H_1 or H_2 (but not to both), but not to any other bus stop. The distance between any two bus stops is the length of the shortest possible route following the streets. That is, if a bus stop is represented as (x, y) with x-coordinate x and y-coordinate y, then the distance between two bus stops (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$. If bus stops A and B are connected to the same hub H_1, then the length of the route from A to B is the sum of the distances from A to H_1 and from H_1 to B. If bus stops A and B are connected to different hubs, e.g., A to H_1 and B to H_2, then the length of the route from A to B is the sum of the distances from A to H_1, from H_1 to H_2, and from H_2 to B. The planning authority of Yong-In city would like to make sure that every citizen can reach every point within the city as quickly as possible. Therefore, city planners want to choose two bus stops to be hubs in such a way that in the resulting bus network the length of the longest route between any two bus stops is as short as possible. One choice P of two hubs and assignments of bus stops to those hubs is better than another choice Q if the length of the longest bus route is shorter in P than in Q. Your task is to write a program to compute the length of this longest route for the best choice P.