CS-3250

Section 01

-

Take Home Exam #3

Patrick Pei

Professor Jeremy Spinrad

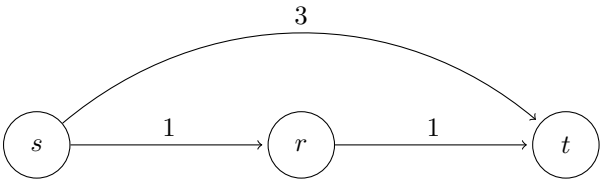
Due 2-15-2017

Exercises

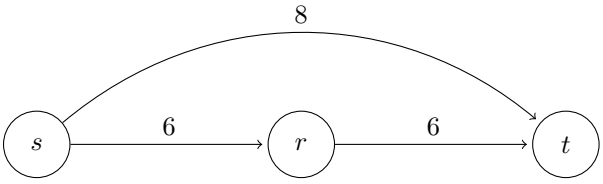
1 You are given a graph G, and you find a shortest path P from s to t in G. Prove or disprove; if G' is formed by adding 5 to the cost of all edges in G, then P is always a shortest path from s to t in G'

No, P will not always be the shortest path from s to t in G'. Proof by contradiction:

G



G'



As is obvious, the shortest path from s to t in G is of cost 2. However, when 5 is added to the cost of all edges, then the shortest path is no longer s→ r→ t but s→ t.

2 You are given a graph G, and you find a minimum spanning tree T for G. If G' is formed by adding 5 to the weights of all edges in G, will G' necessarily be a minimum spanning tree for G?

If G' is formed by adding 5 to the weights of all edges in G, G' is necessarily a minimum spanning tree for G. This is easy to see since the definition of a minimum spanning tree is a subset of edges that connect all vertices with the minimum possible edge weights without cycles. Thus, since 5 is added to every edge in G, G' is necessarily a minimum spanning tree for G because all of the edges changing by the same amount will

not affect the minimum weighted edge between any pair of vertices. Finally, it can also be seen that this is also true in general in that for any graph G with a minimum spanning tree T, if G' is formed by adding or multiplying all edge weights by a positive constant k, T is necessarily still a minimum spanning tree for G'.

3 Design an $O(k(n+m))$ algorithm to find a minimum spanning tree if all edges have integer weights in the range $1...k$.

To start, because the edges are known to have all integer weights in the range $1...k$, extra memory can be used to sort these in linear time either with linear, non-comparison sorting algorithms (radix sort, counting sort, bucket sort) or just by keeping lists of edges by edge weight. From here, a modified version of Prim's algorithm can be implemented to design an $O(k(n+m))$ algorithm to find a minimum spanning tree. Because there is a set constant k number of groups of edges sorted by weights, the time to check for the smallest is clearly k. So for each tracking of the smallest, which is n times for n nodes, the total time it will take is $O(n * k)$. Lastly, because the time it takes for linear sorting of edges is m, the algorithm is $O(k(n) + m)$ which is clearly $O(k(n+m))$. Furthermore, at each step after the first, which starts with randomly selecting a node and adding all the edges by being able to index directly into the position it exists at (ordered by length of the edge) and also by keeping an additional array to track which nodes are already a part of the minimum spanning tree.

4 You are managing a trucking company, and you have trucks of various heights at your warehouse. You are given a graph of highways and intersections, where each highway segment has a maximum allowable height for that segment of road. Design and analyze an algorithm which finds the largest height truck which can be sent to all possible destinations.

In order to find the height of the largest truck which can be sent to all possible destinations, the algorithm must first create a spanning tree such that all vertices of the graph (warehouses) are connected without cycles with the maximum possible total edge weight (maximum allowable height of a highway). This is extremely

similar to creating a minimum spanning tree with the only difference being the goal of maximizing edge weights instead of minimizing edge weights. Thus, Kruskal's algorithm as learned in class can be modified slightly to develop this algorithm. First, all edges will be sorted in $O(mlogm)$ time and one by one will be selected from largest to smallest and added if a cycle is not created; this goes until all vertices are connected. While adding each of these edges, a minimum will simply be kept which represents the lowest maximum allowable height, which is necessarily the bottleneck of the height of the truck that can be sent to all possible destinations. In all, by utilizing the union-find algorithm covered in lecture with data structures to handle these set operations, union operations will take $O(1)$ time each whereas find operations will take $O(logn)$ time each. Thus, the overall algorithm will find the largest height of a truck which can be sent to all possible destinations and will take $O(mlogn)$ time.