Algorithms & Complexity 2/24/2017

0145-344-001

Note Taker: Jai Punjwani

ANNOUNCEMENTS

Topic: Greedy Algorithms & Minimum Spanning Tree

PowerPoint: <http://home.adelphi.edu/~siegfried/cs344/344l6.pdf>

* Optimization problems try to find the best solution from all feasible ones. (ex: shortest path).
* Greedy algorithms seek to take the best option it sees right away in hope of optimizing some outcome. (going for the best **local** option in hope of optimizing the **global** outcome). However, they only work in some cases.

Example Where Greedy Algorithm does **not work**:

Given coins of 4 cent, 3 cent, and 1 cent coin. find the minimum # of coins that make up 6 cents.

The greedy algorithm will choose the highest (4-cent coin) and then 2 1-cent coins, totaling 6 cents. However, two 3-cen coins would be a more optimal choice.

**Minimum Spanning Trees (MST)**

A **spanning tree** is a subset of a graph, G, which connects all vertices of G with the minimum possible number of edges.

A **minimum spanning tree** is a spanning tree that minimizes the weights of the edges. Below are two **greedy** algorithms that help us find a MST.

**Kruskal’s Algorithm**

<https://www.tutorialspoint.com/data_structures_algorithms/kruskals_spanning_tree_algorithm.htm>

* remove any loops and parallel edges (keep the lowest cost edge). We will have **n** vertices.
* make a table of the edges and their costs from least to greatest
* Start with lowest edge in out MST
* Add the next lowest edge that keeps our MST connected but does not create a circuit. repeat this till we have **n-1** edges

**Prim’s Algorithm** <https://www.tutorialspoint.com/data_structures_algorithms/prims_spanning_tree_algorithm.htm>

* remove any loops and parallel edges (keep the lowest cost edge).
* Choose an arbitrary vertex as our root. Add it to our list of visited
* consider all edges from all vertices in our MST so far. Add the least cost edge that leads to an unvisited vertex, and then add that vertex to our list of visited.

**Scheduling Problem**

**Knapsack Problem**