**CS430 Lecture 16 Activities**

Opening Questions

1. Briefly explain what two properties a problem must have so that a greedy algorithm approach will work.

2. A good cashier gives change using a greedy algorithm to minimize the number of coins they give back. Explain this greedy algorithm.

3. For what types of optimization problems does optimal substructure fail?

Activity Selector Problem

Given a set S of n activities each with start Si, Finish Fi, find the maximum set of Compatible Activities (non-overlapping).

(or could be jobs scheduled with fixed start and end times instead of meetings)

1. The brute force approach would be to find all possible subsets of n activities, eliminate the ones with non-compatible meetings, and find the largest subset. How many subsets are there?

2. Prove the Activity Selector Problem has optimal substructure (using similar proof by contradiction approach that we used for dynamic programming: assume you have an optimal answer, remove something to get to the largest sub-problem, show that the sub-problem must also be solved optimally).

Proving a Greedy Choice Property

To prove a Greedy Choice Property for a problem (that local optimal choice leads to global optimal solution) use the following “cut and paste” proof.

1. Assume you have an optimal answer that does not contain the greedy choice you are trying to prove.
2. Show that you can “cut” something out of that optimal answer and “paste” in the greedy choice you are trying to prove. Therefore either the assumed optimal answer already contained the greedy choice, or you can make the assumed optimal contain the greedy choice.
3. Therefore there is always an optimal answer that contains the greedy choice (can be continued like an inductive proof).

3. Try various “common sense” greedy approaches that divide the problem into a sub-problem(s) and try to come up with counter-examples or prove the greedy choice is correct.

Does every Optimization Problem exhibit Optimal Substructure?

Consider the following two problems in which we are given a directed graph *G* = (*V, E*) and vertices *u*, *v* ∈ *V*.

1. Un-weighted shortest path: Find a path from *u* to *v* consisting of the fewest edges.
2. Un-weighted longest simple path: Find a simple path from *u* to *v* consisting of the most edges.

4. Try to prove Optimal Substructure for the above two problems.

5. Why does optimal substructure fail for some optimization problems?