Review

Final Exam

- 6/13 E11 Terman Hall 9AM-11:45AM
  - Bring your student ID
- Greedy algorithm (after minimum spanning trees)
  - Huffman encoding
  - Scheduling problem
- Dynamic programming
  - Longest increasing subsequences, edit distance, knapsack
  - Chain matrix multiplication, optimal binary search trees
  - Shortest reliable paths, Floyd-Warshall, independent set in trees
  - TSP, coin change
- NP-completeness
  - P, NP, co-NP, NP-hard, NP-complete
  - Reductions
- Approximation algorithm

Greedy algorithm vs. dynamic programming

- Both solve optimization problems using “optimal substructure” – optimal solution contains optimal solutions to subproblems.
- Greedy algorithms works only for limited problems.
- Dynamic programming are more widely applicable.

Hallmarks of optimization problems

Greedy algorithms	Dynamic Programming

1. Optimal substructure
An optimal solution to a problem (instance) contains optimal solutions to subproblems.

2. Overlapping subproblems
A recursive solution contains a "small" number of distinct subproblems repeated many times.

Greedy algorithms	Dynamic Programming
3. Greedy choice property *Locally optimal choices lead to globally optimal solution*	***Greedy Choice is not possible*** *Globally optimal solution requires trace back through many choices*

Proving optimal substructure

- Consider the problem of making change for *n* cents using the fewest number of coins. Assume that each coin's value is an integer.
- Show that this problem has *optimal substructure*.
- Suppose we have an optimal solution for a problem of making change for *n* cents, and we know that this optimal solution uses a coin whose value is *c* cents; let this optimal solution use *k* coins.
- We claim that this optimal solution for the problem of *n* cents must contain within it an optimal solution for the problem of $n-c$ cents.
- We use the usual *cut-and-paste argument.*
- There are $k-1$ coins in the solution to the $n-c$ cents problem used within our optimal solution to the *n* cents problem.
- If we had a solution to the $n-c$ cents problem that used fewer than $k-1$ coins, then we could use this solution to produce a solution to the *n* cents problem that uses fewer than *k* coins, which contradicts the optimality of our solution.

Proving greedy-choice property (1/2)

- If {25, 10, 5, 1} are available coins, greedy algorithm works.
- We’ll prove greedy-choice property for this case.
- Suppose an optimal solution to making change for *n* cents includes one coin of value *c*, where *c* is the largest coin value such that $c \leq n$.
- Consider some optimal solution. If this optimal solution includes a coin of value *c*, then we are done.
- Otherwise, this optimal solution does not include a coin of value *c*.
- We have four cases to consider:

Proving greedy-choice property (2/2)

1. If $1 \leq n < 5$, then $c=1$. A solution may consist only of pennies, and so it must contain the greedy choice.
2. If $5 \leq n < 10$, then $c=5$. By supposition, this optimal solution does not contain a nickel, and so it consists of only pennies. Replace five pennies by one nickel to give a solution with four fewer coins.
3. If $10 \leq n < 25$, then $c=10$. By supposition, this optimal solution does not contain a dime, and so it contains only nickels and pennies. Some subset of the nickels and pennies in this solution adds up to 10 cents, and so we can replace these nickels and pennies by a dime to give a solution with fewer coins.
4. If $25 \leq n$, then $c=25$. By supposition, this optimal solution does not contain a quarter, and so it contains only dimes, nickels, and pennies. If it contains three dimes, we can replace these three dimes by a quarter and a nickel, giving a solution with one fewer coin. If it contains at most two dimes, then some subset of the dimes, nickels, and pennies adds up to 25 cents, and so we can replace these coins by one quarter to give a solution with fewer coins.

Proving correctness of greedy algorithm

- Thus, we have shown that there is always an optimal solution that includes the greedy choice.
- We can combine the greedy choice with an optimal solution to the remaining subproblem to produce an optimal solution to our original problem. (by optimal substructure)
- Therefore, the greedy algorithm produces an optimal solution.

NP-completeness

- P, NP, co-NP, NP-hard, NP-complete
- Reductions
- NP-complete proofs
- CIRCUIT-SAT, SAT, 3-SAT
- IND-SET, CLIQUE, VERTEX-COVER
- HAM, TSP

Approximation algorithm

- 2 approximation algorithm for vertex-cover
- 2 approximation algorithm for TSP with triangle inequality