

CSC 301: Analysis of Algorithms

Lecture 10: Greedy Algorithms

William Rebelsky
[rebelskyw]

Grinnell College

Monday 12th February, 2024

Reminders

- Homework 2 is due Wednesday
- Quiz 2 is due Tonight

Table of Contents

1 Greedy Basics

2 Knapsack

Basics of Greedy Algorithms

- A Greedy Algorithm makes the locally optimal choice and hopes it achieves the globally optimal solution
- Greedy solutions can be both recursive and iterative

Basics of Greedy Algorithms

- Designed in 3 steps:
 - ① Cast the optimization problem as one in which you make a choice and are left with one subproblem to solve
 - ② Prove that there is always an optimal solution to the original problem that makes the greedy choice (to show that the greedy choice is safe)
 - ③ Demonstrate the optimal substructure by showing that, having made the greedy choice, what remains is a subproblem with the property that if you combine the optimal solution to the subproblem with the greedy choice made, you arrive at an optimal solution to the problem

Greedy Choice Property

- **Greedy Choice Property:** You can assemble a globally optimal solution by making locally optimal (greedy) choices.
- Generally need to prove this. Start with a globally optimal solution to a sub problem, modify the solution to substitute the greedy choice for another choice resulting in a similar, smaller, subproblem. See Theorem 15.1 for an example

Optimal Substructure

- **Optimal Substructure:** An optimal solution to the problem contains within it optimal solutions to subproblems. We will see later that this is also required for dynamic programming
- Generally need to prove this. See Theorem 15.1 for an example

Table of Contents

1 Greedy Basics

2 Knapsack

Knapsack Problem

- **0-1 Knapsack.** Can hold W pounds, can choose to take any subset of the n items. Each item as a weight (w_i) and value (v_i). Each item must be taken in its entirety or left behind. Goal: Maximize value given the weight restriction.
- **Fractional Knapsack.** Can hold W pounds, can choose to take any subset of the n items. Each item as a weight (w_i) and value (v_i). Each item can be taken in any amount. Goal: Maximize value given the weight restriction.
- Goals:
 - Show that both exhibit the optimal substructure property
 - For each problem: show that it can be solved greedily, or give a counter example
 - Start with the optimal substructure property, after 5ish minutes, we will discuss how to show that for 0-1 Knapsack

0-1 Knapsack Problem optimal substructure

- If the most valuable load weighing at most W pounds includes item j , then the remaining load must be the most valuable load weighing at most $W - w_j$ pounds that can be obtained using the $n-1$ original items that don't include j .

BLANK

This frame intentionally left blank

Knapsack Answer Sketches

- **0-1:** Counterexample. 3 Items, max weight 50 pounds.
Item 1: 10 pounds, \$60. Item 2: 20 pounds, \$100. Item 3:
30 pounds, \$120. Optimal solution: Items 2,3. Greedy
solution: Items 1,2.
- **Fractional:** Take largest density