Greedy Algorithms: Intro

Textbook: Chapter 9

Making change

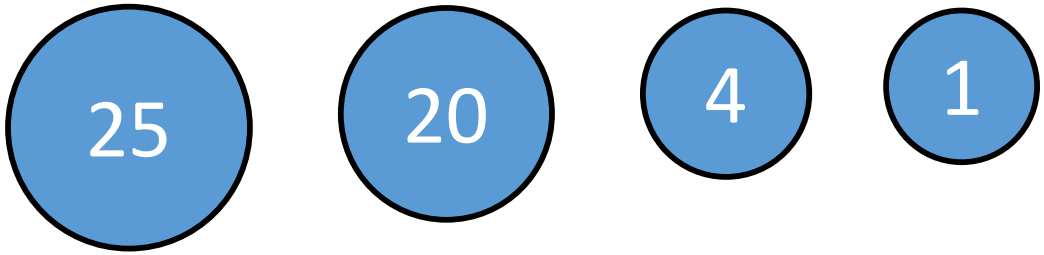
- Imagine you're a shop clerk giving change and you want to use the fewest coins

- Strategy idea:
  - Always select the biggest feasible coin
- Example: 37 cents
  - 1 quarter (need 12 more cents)
  - 1 dime (need 2 more cents)
  - 2 pennies (2 *whats*?!)
- This is a "greedy" algorithm

Does this algorithm always give the best answer?

- For US/Canadian coins, yes
  - Even without pennies

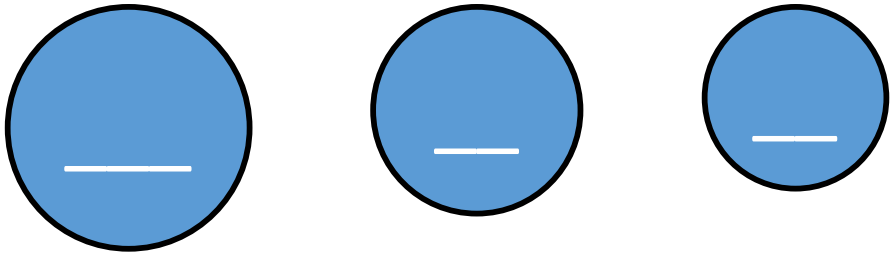
- But what if your coins were:



- And you had to give 28 cents?
  - Greedy algorithm result: 25, 1, 1, 1 → 4 coins
  - But there is a 3-coin answer

Puzzle – just for fun!

- Fill in the blanks with numbers that work
  - I.e. make a “smaller” counterexample than the one I used on my previous slide
- What if your coins were:



- And you had to give ____ cents?
  - Greedy algorithm result: ___ , ___ , ___ → 3 coins
  - But there is a 2-coin answer

*PFG for first person to give a correct answer

Optimization/decision problems

- An optimization problem is one in which you want to find not just *a* solution, but the *best* solution
  - As opposed to decision problem – "does a solution exist?"
  - Decision problem has a yes/no answer
  - Optimization problem is about minimizing or maximizing

- Greedy algorithms attempt to solve *optimization problems*

Greedy algorithms

- Construct a solution to an optimization problem through a sequence of choices
- Always choose the best option available “right now”
  - The “best” choice is the one that gets us closest to an optimal solution (e.g. take the biggest feasible coin)
- You hope that by choosing a *local* optimum at each step, you will end up at a *global* optimum

Greedy algorithms

- Greedy choice properties:
  - *Feasible: it has to satisfy the problem's constraints*
    - If you are making change for 17 cents, you don't pick a quarter
  - *Locally optimal: it has to be the best local choice among all feasible choices available on that step*
  - *Irrevocable: Once made, it cannot be changed during subsequent steps of the algorithm*

Greedy algorithms

- We will examine greedy algorithms for the following problems:
  - Finding a minimum spanning tree (MST) of a graph
    - Prim's algorithm
    - Kruskal's algorithm
  - Finding shortest paths in a graph
    - Dijkstra's algorithm
  - Coloring a graph

Practice problems

1. Chapter 9.1, page 324, question 9
2. Chapter 9.2, page 331, questions 1,2
3. Chapter 9.3, page 337, questions 1,2,4