Introduction to Greedy Algorithms

Algorithm Design Paradigm

- No single *"silver bullet"* for solving problems

- Some Design Paradigms
  - Divide and Conquer
  - Randomized Algorithms
  - Greedy Algorithms
  - Dynamic Programming

Optimization Problem

- an **optimization problem** is the **problem** of finding the best solution from all feasible solutions

  - There is some objective for the problem that should be either minimized or maximized
  - Best solution will be the one that minimizes or maximizes the objective of the problem

Greedy Algorithms

- Iteratively make "myopic" decisions and hope everything works out at the end
- A *greedy algorithm* always makes the choice that looks best at the moment
  - Everyday examples:
    - Coin Changing Problem
  - The hope: a locally optimal choice will lead to a globally optimal solution
  - For some problems, it works

Contrast with Divide & Conquer and DP

- Easy to propose many greedy algorithms for many problems
- Easy running time analysis
- Hard to establish correctness
- Danger: Most greedy algorithms are NOT correct
  - (even if your intuition says otherwise)

Minimum Coin Change Problem

- Given a set of coins and a value, we have to find the minimum number of coins which satisfies the value.
- **Example**
- coins[] = {5,10,20,25}
- value = 50

Minimum Coin Change Problem

- **Example**
  - coins[] = {5,10,20,25}
  - value = 50
- **Possible Solutions**
  - {coin * count}
  - {5 * 10} = 50 [10 coins]
  - {5 * 8 + 10 * 1} = 50 [9 coins] goes on.
  - {10 * 5} = 50 [5 coins]
  - {20 * 2 + 10 * 1} = 50 [3 coins]
  - {20 * 2 + 5 * 2} = 50 [4 coins]
  - {25 * 2} = 50 [2 coins]
  - etc etc
- **Best Solution**
  - Two 25 rupees. Total coins two.
  - 25 * 2 = 50

Minimum Coin Change Algorithm

1. Get coin array and a value.
2. Make sure that the array is sorted.
3. Take coin[i] as much we can.
4. Increment the count.
5. If solution found,
    - break it.
6. Otherwise,
    - follow step 3 with the next coin. coin[i+1].
7. Finally, print the count.

Greedy Algorithm for Coin change Example 1

- coin[] = {25,20,10,5}
- value = 50
- Take coin[0] twice. (25+25 = 50).
- Total coins = 2 (25+25)

Greedy Algorithm for Coin change Example 2

- coin[] = {25,20,10,5}
- value = 70
- Take coin[0] twice. (25+25 = 50).
- If we take coin[0] one more time, the end result will exceed the given value. So, change the next coin.
- Take coin[1] once. (50 + 20 = 70).
- Total coins needed = 3 (25+25+20).

Greedy Approach for Coin Change

- In this approach, we are not bothering about the overall result.
- We just pick the best option in each step and hoping that it might produce the best overall result.
- Hence, this method called as the greedy approach.

Greedy Algorithm might not work for some coin denominations

- Example
  - coin[] = {1, 3, 4}
  - value = 6
  - Greedy solution = {4,1,1} = 3 coins
  - Best Solution = {3,3} = 2 coins