UNIT IV GREEDY TECHNIQUE

Greedy Technique – Minimum Spanning Tree – Prim's Algorithm – Kruskal's Algorithm – Single-source-shortest-paths Problem – Dijkstra's Algorithm – Huffman Coding – Fractional Knapsack Problem

Greedy Strategy

- Algorithms for optimization problems typically go through a sequence of steps, with a set of choices at each step.
- For many optimization problems, using dynamic programming to determine the best choices is overkill; simpler, more efficient algorithms will do.
- A greedy algorithm always makes the choice that looks best at the moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.

Greedy Technique

Constructs a solution to an *optimization problem* piece by piece through a sequence of choices that are:

Defined by an objective function and a set of constraints

- ***feasible*, i.e. satisfying the constraints**
- ***locally optimal* (with respect to some neighborhood definition)**
- ***greedy* (in terms of some measure), and irrevocable**

For some problems, it yields a globally optimal solution for every instance. For most, does not but can be useful for fast approximations.

Greedy Technique

- feasible, i.e., it has to satisfy the problem's constraints
- locally optimal, i.e., it has to be the best local choice among all feasible choices available on that step
- irrevocable, i.e., once made, it cannot be changed on subsequent steps of the algorithm
- These requirements explain the technique's name: on each step, it suggests a "greedy" grab of the best alternative available in the hope that a sequence of locally optimal choices will yield a (globally) optimal solution to the entire problem.

Change-Making Problem

Given unlimited amounts of coins of denominations d_1 > d2 > d3 >..… > d_m

give change for amount n with the least number of coins

For example,

the widely used coin denominations in the United States are

d1 = 25 (quarter), d2 = 10 (dime), d3 = 5 (nickel), and d4 = (penny).

How would you give change with coins of these denominations of,

say, 48 cents?

$d_1 = 25$, $d_2 = 10$, $d_3 = 5$, $d_4 = 1$ and $n = 48$

Greedy solution: <1, 2, 0, 3 > = 5 Coins

Change-Making Problem

Greedy solution is

- **optimal for any amount and “normal’’ set of denominations**
- **may not be optimal for arbitrary or imaginary coin denominations**

Example: Prove the greedy algorithm is optimal for the above denominations.

For example, $d1 = 25$, $d2 = 10$, $d3 = 1$, and $n = 30$ [Arbitrary denominations]

Greedy solution: < 1, 0, 0, 4 > = 5 Coins

Optimal solution: < 0, 3, 0, 0 > = 3 Coins

For example, $d1 = 25$, $d2 = 10$, $d3 = 5$, $d4 = 1$ and n = 30 [Normal denominations]

Greedy solution: < 1, 0, 1, 0 > = 2 Coins

Optimal solution: <1, 0, 1, 0 > = 2 Coins

Change-making problem for which the greedy algorithm does not yield an optimal solution.

Example:

For the coin denominations d1 = 7, d2 = 5, d3 = 1 and the amount n = 10.

Greedy solution: < 1, 0, 3 > = 4 coins

Optimal solution: < 0, 2, 0 > = 2 coins

The greedy algorithm yields one coin of denomination 7 and three coins of denomination 1. The actual optimal solution is two coins of denomination 5.

Applications of the Greedy Strategy

- **Optimal solutions:**
  - change making for "normal" coin denominations
  - minimum spanning tree (MST)
  - single-source shortest paths
  - simple scheduling problems
  - Huffman codes

- **Approximations/heuristics:**
  - traveling salesman problem (TSP)
  - knapsack problem
  - other combinatorial optimization problems

Thank You