Text Book: Introduction to the Design and Analysis of Algorithms Author: Anany Levitin 2 nd Edition

REFERENCE BOOK: "Fundamentals of Computer Algorithms", Horowitz, Sahni, Rajasekaran, Universities Press, 2/e, 2007

Unit-4

5. Greedy Approach

Greedy Approach is a general design technique and it is applicable to optimization problems only. The greedy approach suggests constructing a solution through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution to the problem is reached. On each step—and this is the central point of this technique—the choice made must be:

- **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

Greedy is the most straight forward design technique. Most of the problems have n inputs and require us to obtain a subset that satisfies some constraints. Any subset that satisfies these constraints is called a feasible solution. We need to find a feasible solution that either maximizes or minimizes the objective function. A feasible solution that does this is called an optimal solution.

CONTROL ABSTRACTION

```
Algorithm Greedy (a, n)
// a(1 : n) contains the 'n' inputs
{
solution :={} ;  // initialize the solution to empty
        for i:=1 to n do
```

```
            {
            x := select (a);
            // initialize the solution to empty
            if feasible (solution, x) then
            solution := Union (Solution, x);
            }

    return solution;

    }
```

Examples of Greedy Algorithms:

1. Coin-change problem
2. Minimum Spanning Tree (MST)
    a. Prim's Algorithm
    b. Kruskal's Algorithm
3. Single-source shortest paths
    a. Dijkstra's Algorithm
4. Huffman codes

1. Coin Change Problem

A greedy algorithm to find the minimum number of coins for making the change of a given amount of money. Usually, this problem is referred to as the change-making problem.

- In the change-making problem, we're provided with an array, D = { d1, d2, d3,…..dm} of m distinct coin denominations.
- Now we need to find an array(subset) s having minimum number of coins that add up to a given amount of money n, provided that there exists a viable solution.

- Let's consider a real-life example for a better understanding of the change-making problem.
- Let's assume that we're working at a cash counter and have an infinite supply of D ={1,2,5,10, 50,100} valued coins.
- **A person buys things worth Rs. 72 and gives a Rs. 100 bill. How does the cashier give change for Rs. 28?**

Option	Choosen Coins
28-10 = 18	10
18-10 =8	10,10
8-5 =3	10,10,5
3-2 = 1	10,10,5,2
1-1 =0	10,10,5,2,1