Syllabus

UNIT I INTRODUCTION: Introduction to Algorithms – Fundamentals of Algorithmic Problem Solving – Fundamentals of the Analysis of Algorithmic Efficiency – Analysis Framework – Asymptotic Notations and Basic Efficiency Classes – Mathematical Analysis of Recursive Algorithms – Mathematical Analysis of Non-recursive Algorithms

UNIT II DIVIDE-AND-CONQUER: Divide and Conquer Methodology – Binary Search – Merge Sort – Quick Sort – Heap Sort – Multiplication of Large Integers – Strassen's Matrix Multiplication

Syllabus

UNIT III **DYNAMIC PROGRAMMING:** Dynamic Programming – Change-making Problem – Computing a Binomial Coefficient – All-pairs Shortest-paths Problem – Warshall's and Floyd's Algorithms – 0/1 Knapsack Problem

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

Syllabus

UNIT V BACKTRACKING AND BRANCH-AND-BOUND: Backtracking – N-Queens Problem – Hamiltonian Circuit Problem – Subset Sum Problem – Branch-and- Bound – Travelling Salesman Problem

UNIT VI LIMITATIONS OF ALGORITHM POWER: P and NP Problems – NP-Complete Problems – Decision Trees – Information Retrieval – Pattern Matching – Data Science Algorithms

UNIT III DYNAMIC PROGRAMMING:

Dynamic Programming **– Change-making Problem –** Computing a Binomial Coefficient – All-pairs Shortest-paths Problem – Warshall's and Floyd's Algorithms – 0/1 Knapsack Problem

Change Making Problem

- The change-making problem addresses the question of finding the minimum number of coins (of certain denominations) that add up to a given amount of money.

- It is a special case of the integer knapsack problem, and has applications wider than just currency.

Change Making Problem

- It is also the most common variation of the coin change problem, a general case of partition in which, given the available denominations of an infinite set of coins, the objective is to find out the number of possible ways of making a change for a specific amount of money, without considering the order of the coins.

- making change is minimization problem, we choose the solution where we have to give minimum no of coins for given amount.

- Methods of solving: 1. Greedy Method 2. Dynamic Method

Change-Making Problem

Given unlimited amounts of coins of denominations

$d_1 > \ldots > d_m$,

give change for amount n with the least number of coins

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

solution: <1, 2, 0, 3>

$F[0] = 0$

n	0	1	2	3	4	5	6
F	0						

$F[1] = \min\{F[1-1]\} + 1 = 1$

n	0	1	2	3	4	5	6
F	0	1					

$F[2] = \min\{F[2-1]\} + 1 = 2$

n	0	1	2	3	4	5	6
F	0	1	2				

$F[3] = \min\{F[3-1], F[3-3]\} + 1 = 1$

n	0	1	2	3	4	5	6
F	0	1	2	1			

$F[4] = \min\{F[4-1], F[4-3], F[4-4]\} + 1 = 1$

n	0	1	2	3	4	5	6
F	0	1	2	1	1		

$F[5] = \min\{F[5-1], F[5-3], F[5-4]\} + 1 = 2$

n	0	1	2	3	4	5	6
F	0	1	2	1	1	2	

$F[6] = \min\{F[6-1], F[6-3], F[6-4]\} + 1 = 2$

n	0	1	2	3	4	5	6
F	0	1	2	1	1	2	**2**

FIGURE 8.2 Application of Algorithm *MinCoinChange* to amount $n = 6$ and coin denominations 1, 3, and 4.

ALGORITHM *ChangeMaking*$(D[1..m], n)$

//Applies dynamic programming to find the minimum number of coins
//of denominations $d_1 < d_2 < \cdots < d_m$ where $d_1 = 1$ that add up to a
//given amount n
//Input: Positive integer n and array $D[1..m]$ of increasing positive
// integers indicating the coin denominations where $D[1] = 1$
//Output: The minimum number of coins that add up to n

$F[0] \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
$temp \leftarrow \infty$; $j \leftarrow 1$
while $j \le m$ **and** $i \ge D[j]$ **do**
$temp \leftarrow \min(F[i - D[j]], temp)$
$j \leftarrow j + 1$
$F[i] \leftarrow temp + 1$
return $F[n]$

6. Find Solution For given Making change problem using Dynamic Programming. W = 12

Coins	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13
2	0	1	1	2	2	3	3	4	4	5	5	6	6	7
3	0	1	1	1	2	2	2	3	4	3	4	5	4	5
7	0	1	1	1	2	2	2	1	2	2	2	3	3	3
10	0	1	1	1	2	2	2	1	2	2	1	2	2	2
20	0	1	1	1	2	2	2	1	2	2	1	2	2	2

Value :- 10, 2, 1

Thank You