UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Worksheet 5(a)

Student Name: Rahul Saxena **UID:** 24MCI10204

Branch: MCA(AI&ML) **Section/Group:** 3-B

Semester: 1st semester **Date of Performance:** 11/10/2024

Subject Name: Design and analysis of Algorithm Lab **Subject Code:** 24CAP-612

AIM:

Implement 0/1 Knapsack problem using dynamic programming.

Task To be Done:

- **Understand the 0/1 Knapsack Problem:** Learn the problem where you have a set of items, each with a weight and a value, and a maximum capacity for the knapsack. The objective is to maximize the total value without exceeding the capacity**.**
- **Implement Dynamic Programming Approach:** Solve the problem by breaking it down into smaller overlapping subproblems and storing solutions to subproblems to avoid redundant calculations.
- **Develop a Program:** Write a Java program that implements the dynamic programming approach to solve the 0/1 Knapsack problem.

Source Code:

```
public class Knapsack {
   static int knapsack(int[] weights, int[] values, int capacity, int n) {
      int[][] dp = new int[n + 1][capacity + 1];
      for (int i = 0; i <= n; i++) {
         for (int w = 0; w <= capacity; w++) {
            if (i == 0 || w == 0) {
               dp[i][w] = 0;
            } else if (weights[i - 1] <= w) {
               dp[i][w] = Math.max(values[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);
            } else {
               dp[i][w] = dp[i - 1][w];
            }
         }
      }
      return dp[n][capacity];
   }
   public static void main(String[] args) {
      int[] weights = { 2, 3, 4, 5 };
      int[] values = { 3, 4, 5, 6 };
      int capacity = 5;
```

```
        int n = weights.length;
        int maxValue = knapsack(weights, values, capacity, n);
        System.out.println("Total Profile: " + maxValue);
    }
}
```

Output:

Learning Outcome:

- Understand the concepts of dynamic programming and how to apply them to solve optimization problems.
- Gain experience with Java programming, especially in working with arrays and loops.
- Learn to break down complex problems into manageable subproblems.
- Develop skills in analysing and optimizing algorithms for better performance.