



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
MUNSHI NAGAR, ANDHERI (WEST), MUMBAI – 400 058.
(Autonomous College Affiliated to University of Mumbai)
MASTER OF COMPUTER APPLICATIONS

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

UCID: 2024510001 BATCH: A NAME: Atharva Vasant Angre

EXPERIMENT NO: 04

EXPERIMENT TITLE: To implement Knapsack Algorithm.

4.1 To Implement Knapsack Algorithm (0/1 Knapsack).

Objective:

1. To Implement Knapsack Algorithm (0/1 Knapsack).



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
MUNSHI NAGAR, ANDHERI (WEST), MUMBAI - 400 058.
(Autonomous College Affiliated to University of Mumbai)
MASTER OF COMPUTER APPLICATIONS

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Program code: -

0/1 Knapsack

```
1 public class zerobyoneknapsack { new *
2 public static void main(String[] args) { new *
3     int[] weights = {2,3,5,7};
4     int[] profits = {3,4,8,6};
5     int capacity = 10;
6     int n = weights.length;
7
8     int maxProfit = knapsack(n, capacity, weights, profits);
9     System.out.println("Maximum Profit: " + maxProfit);
10 }
11
12 public static int knapsack(int n, int capacity, int[] weights, int[] profits) { 1usage new *
13     int[][] dp = new int[n + 1][capacity + 1];
14
15     for (int i = 1; i <= n; i++) {
16         for (int w = 0; w <= capacity; w++) {
17             if (weights[i - 1] <= w) {
18                 dp[i][w] = Math.max(dp[i - 1][w], profits[i - 1] + dp[i - 1][w - weights[i - 1]]);
19             } else {
20                 dp[i][w] = dp[i - 1][w];
21             }
22         }
23     }
24     return dp[n][capacity];
25 }
26 }
27 }
```



BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
MUNSHI NAGAR, ANDHERI (WEST), MUMBAI - 400 058.
(Autonomous College Affiliated to University of Mumbai)
MASTER OF COMPUTER APPLICATIONS

Class : F.Y.MCA Semester : II Academic Year : 2024-25

Course Name : Design and Analysis of Algorithm MC507

Subject Incharge : Prof.Nikhita Mangaonkar

Output:

The screenshot displays an IDE interface with a project named 'knapsack'. The project structure on the left includes 'idea', 'out', 'src', 'Fractional_Knapsack.java', 'Fractional_Knapsack', 'Object', 'Main', 'zerobyoneknapsack', 'gitignore', 'knapsack.iml', 'External Libraries', and 'Scratches and Consoles'. The 'Main.java' file is open, showing the following code:

```
1 public class zerobyoneknapsack { new *
2     public static void main(String[] args) { new *
3         int[] weights = {2,3,5,7};
4         int[] profits = {3,4,8,6};
5         int capacity = 10;
6         int n = weights.length;
7
8         int maxProfit = knapsack(n, capacity, weights, profits);
9         System.out.println("Maximum Profit: " + maxProfit);
10    }
11
12    public static int knapsack(int n, int capacity, int[] weights, int[] profits) { 1usage new *
13        int[][] dp = new int[n + 1][capacity + 1];
14
15        for (int i = 1; i <= n; i++) {
16            for (int w = 0; w <= capacity; w++) {
17                if (weights[i - 1] <= w) {
18                    dp[i][w] = Math.max(dp[i - 1][w], profits[i - 1] + dp[i - 1][w - weights[i - 1]]);
19                } else {
20                    dp[i][w] = dp[i - 1][w];
21                }
22            }
23        }
24        return dp[n][capacity];
25    }
```

The Run console at the bottom shows the command executed: `"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2\lib\idea_rt.jar=50242:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2\bin" -Dfile.encoding=UTF-8`. The output is: `Maximum Profit: 15`. The process finished with exit code 0.

Conclusion:

The 0/1 Knapsack Algorithm helps in selecting items to maximize profit while keeping the total weight within the given capacity. It uses Dynamic Programming to ensure the best selection by considering each item's weight and profit. This method guarantees an optimal solution but requires extra space. Overall, it is an efficient and widely used approach for solving resource allocation problems.