|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **Analysis of Algorithms** | **Semester:** | **IV** |
| **Date of Performance:** | **20 / 02 / 2024** | **Batch No:** | **A – 2** |
| **Faculty Name:** | **Dr. Aarti Phadke** | **Roll No.:** | **16014022050** |
| **Faculty Sign & Date:** |  | **Grade / Marks:** | **\_\_\_ / 25** |

**Experiment No.: 5**

**Title: Knapsack Problem**

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Implementation of Knapsack Problem using Greedy Problem. |

|  |
| --- |
| **COs to be achieved:** |
| **CO2:** Describe various algorithm design strategies to solve different problems. |

|  |
| --- |
| **Apparatus / Software Tools Used:** |
| 1. VS Code 2. Microsoft Excel |

|  |
| --- |
| **Theory:** |
| The knapsack problem represents constraint satisfaction optimization problems’ family. Based on the nature of constraints, the knapsack problem can be solved with various problem-solving strategies.  Typically, these problems represent resource optimization solutions. Given a set of n inputs.   * Find a subset, called feasible solution, of the n inputs subject to some constraints, and satisfying a given objective function. * If the objective function is maximized or minimized, the feasible solution is optimal. * It is a locally optimal method.   New Concepts to be learned:  Application of algorithmic design strategy to any problem, Greedy method of problem-solving Vs other methods of problem solving, optimality of the solution, knapsack problem and their applications. |

|  |
| --- |
| **Stepwise-Procedure/Algorithm:** |
|  |

|  |
| --- |
| **Upload the code/Output:** |
| Code:  def knapsack\_greedy(m, n, profits, weights):      x = [0.0] \* n      U = m        ratios = [(profits[i] / weights[i], i) for i in range(n)]      ratios.sort(reverse = True)        for \_, i in ratios:          if weights[i] > U:              break            x[i] = 1.0          U -= weights[i]        if i < n:          x[i] = U / weights[i]          total\_profit = sum(profits[i] \* x[i] for i in range(n))          print("\nsolution vector: ", x)          return total\_profit    m = int(input("\nenter knapsack size: "))  n = int(input("\nenter number of objects: "))  print("\n")  profits = [int(input(f"enter the profit of object {i + 1}: ")) for i in range(n)]  print("\n")  weights = [int(input(f"enter the weight of object {i + 1}: ")) for i in range(n)]    total\_profit = knapsack\_greedy(m, n, profits, weights)  print("\ntotal Profit: ", total\_profit)  Output:    Solving Problem: |

|  |
| --- |
| **Post Lab Subjective / Objective Type Questions:** |
| **Solve the following knapsack instance by greedy programming technique.**  **Capacity = 8, N = 4**  **Let Pi and Wi are as shown in table.**   |  |  |  | | --- | --- | --- | | i | Pi | Wi | | 1 | 1 | 2 | | 2 | 2 | 3 | | 3 | 5 | 4 | | 4 | 6 | 5 |   Output:    Solving Problem: |

|  |
| --- |
| **Conclusion:** |
| Implementing the Knapsack Problem with Greedy strategy helps understand practical algorithm design for optimization. It introduces students to the trade-offs between greedy methods and other problem-solving approaches, offering insights into resource optimization and locally optimal solutions. |

**Signature of faculty in-charge with Date:**