মেট্রোপলিটন ইউনিভার্সিটি

Metropolitan University

Project Titel: 0/1 Knapsack Problem Visualizer using Dynamic Programming

Submitted By

Fahmida Akter Nitu , Sumaiya Islam Sumi
Student ID: 232-115-042 , 232-115-075
Department of Computer Science and Engineering

.

Submitted To

Nasif Istiak, Remon
Lecturer, Department of CSE
Metropolitan University

Submission Date 11/04/2025

1. Introduction

This report presents an interactive visualizer for solving the 0/1 Knapsack problem using dynamic programming (DP) in C++. The visualizer helps users understand the DP process step by step. It also highlights the decision-making involved in filling the DP table, making it

2. Problem Description

The 0/1 Knapsack problem asks us to maximize the total value of items in a knapsack without exceeding a given weight capacity. Each item has a weight and value. The goal is to find the optimal combination of items to pack into the knapsack, ensuring the maximum value is achieved while staying within the weight limit.

3. Solution Approach

We use dynamic programming (DP) to solve the problem. The process involves:

- **DP Table Construction**: We create a 2D table where the rows represent items and the columns represent capacities. The table is filled based on the following:
    - If an item fits within the current capacity, we check whether including it gives a higher value than excluding it.
    - If it doesn't fit, we simply carry forward the previous best value.
- **Backtracking**: After filling the DP table, we backtrack to determine which items are included in the optimal solution.

4. Code Structure

The code is divided into several parts:

- **Input Handling**: The user enters the number of items, their weights, values, and the knapsack's capacity.
- **DP Table Initialization and Update**: The DP table is initialized and updated inside a nested loop.
- **Display Functions**: The table is printed in a readable format, and a delay is introduced to let users observe the progress.
- **Backtracking**: After the table is filled, we backtrack to find the items that form the optimal solution.

5. Time Complexity

The time complexity of the algorithm is $O(n \times W)$, where n is the number of items and W is the knapsack's capacity. This is due to the nested loops that fill the DP table. The space complexity is $O(n \times W)$ because of the storage required for the table.

6. Conclusion

This program offers a clear and interactive way to understand the 0/1 Knapsack problem using dynamic programming. The visualization process, with step-by-step updates and decision-making explanations, makes it an educational tool that helps learners grasp the logic behind the algorithm effectively.