CU CHANDIGARH UNIVERSITY
DEPARTMENT OF ACADEMIC AFFAIRS
Discover. Learn. Empower.

NAAC GRADE A+
ACCREDITED UNIVERSITY

Project Report

ON

Knapsack Problem Visualizer (JavaScript)

Submitted by

KASHISH (24MCA20042)
SECTION (6 "B")
(Subject Code: 24CAP- 612)

in partial fulfilment for the award of the degree of

Master of Computer Application

CU CHANDIGARH UNIVERSITY
CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Chandigarh University
Jan 2025 – May 2025

Submitted to

DR. Maajid Bashir

Index

1. Introduction

This mini project is a visual implementation of the classic 0/1 Knapsack Problem. The knapsack problem is like packing a bag for a trip - you have different items with different weights and values, but you can only carry a limited weight. This tool helps users understand how computers solve this problem using special techniques called dynamic programming or greedy algorithms.

2. Objective of the Project

To build a user-friendly tool that:

- Lets users input items with their weights and values
- Allows users to set how much weight the knapsack can hold
- Shows which items should be selected to get the highest total value
- Visualizes the solution process so users can learn how these algorithms work

3. Technologies Used

For the JavaScript version:

- HTML: Creates the structure of the web page
- CSS: Makes the page look nice and user-friendly
- JavaScript: Powers the calculations and interactions

For advanced users (C++ version):

- C++ programming language
- Standard Template Library (STL)

4. System Requirements

To use this tool, you need:

- A computer with Windows, Mac, or Linux operating system
- For JavaScript version: Any modern web browser (Chrome, Firefox, Safari, or Edge)
- For C++ version: A text editor like VS Code or Code::Blocks and a C++ compiler

No special hardware is needed - any computer made in the last 10 years should work fine.

5. System Design

The system works in three simple steps:

1. **Input**: Users enter items (name, weight, value) and the maximum capacity
2. **Processing**: The system uses smart algorithms to find the best combination of items
3. **Output**: Shows which items to pick and their total value

The design is simple but effective, making it easy for beginners to understand how optimization algorithms work.

6. Module Description

The project has three main parts:

Input Module:

- Collects item details (name, weight, value)
- Gets the knapsack capacity from the user
- Has error checking to make sure inputs make sense

Algorithm Module:

- JavaScript version uses a greedy approach (picks items with highest value-to-weight ratio first)
- C++ version uses dynamic programming (checks all possible combinations efficiently)
- Shows step-by-step how the solution is found

Output Module:

- Shows which items were selected
- Displays the total weight and value
- Includes visual elements like colored bars to represent items

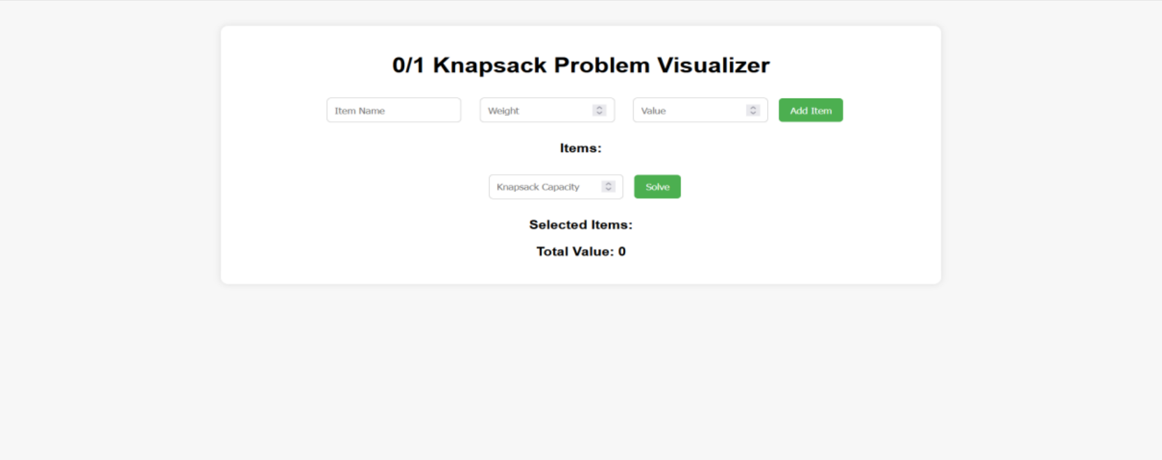
7. Database Design

This project doesn't use a database. All information is:

- Entered by the user when they run the program
- Stored temporarily in the computer's memory
- Lost when the program closes (unless saved manually)

This makes the project simple to run without installation or setup.

8. Screenshots

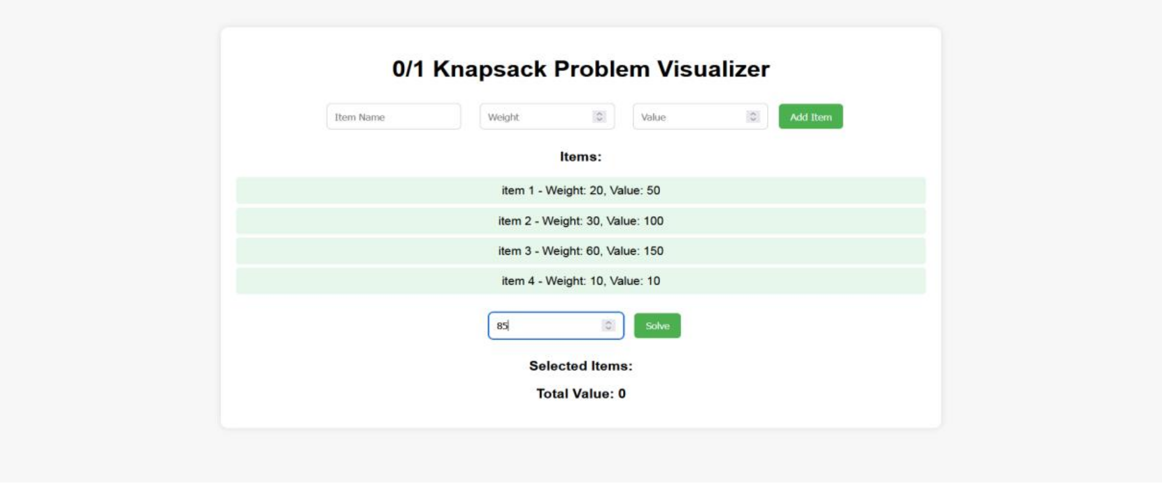


0/1 Knapsack Problem Visualizer

Items:

Selected Items:

Total Value: 0



0/1 Knapsack Problem Visualizer

Items:

item 1 - Weight: 20, Value: 50

item 2 - Weight: 30, Value: 100

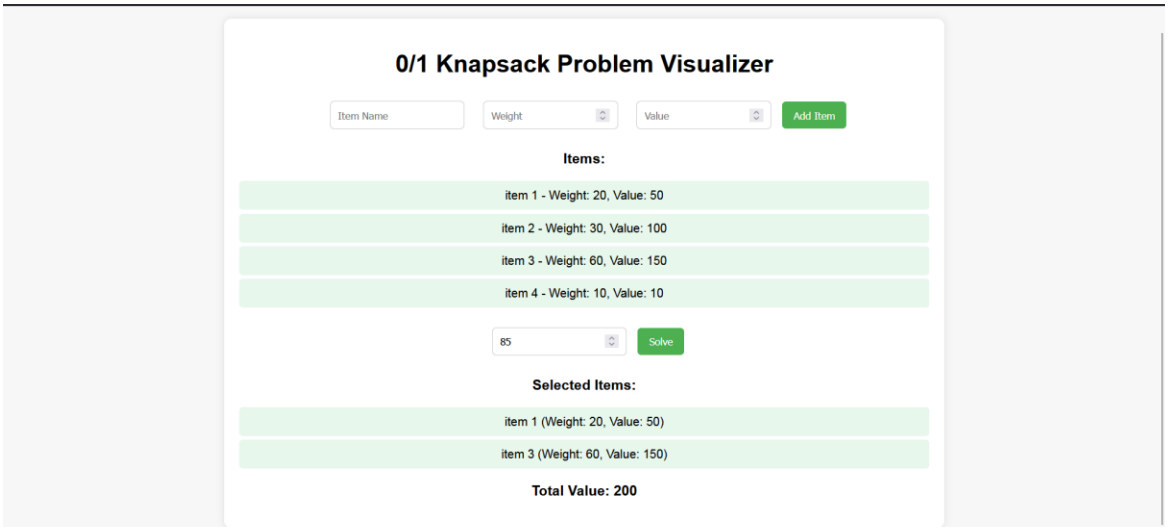
item 3 - Weight: 60, Value: 150

item 4 - Weight: 10, Value: 10

85

Selected Items:

Total Value: 0



0/1 Knapsack Problem Visualizer

Items:

item 1 - Weight: 20, Value: 50

item 2 - Weight: 30, Value: 100

item 3 - Weight: 60, Value: 150

item 4 - Weight: 10, Value: 10

85

Selected Items:

item 1 (Weight: 20, Value: 50)

item 3 (Weight: 60, Value: 150)

Total Value: 200

9. Conclusion

This project successfully demonstrates how computers solve the knapsack problem - a classic optimization challenge. By providing a visual and interactive tool, it helps people understand complex algorithms in a simple way. The project shows that even complicated math problems can be solved step-by-step with the right approach.

The implementation meets all the original goals:

- It's easy to use
- It clearly shows which items should be selected
- It helps users understand how optimization algorithms work

10. Future Scope

This project could be improved in several ways:

Enhanced User Interface:

- Add colorful graphs and charts
- Create animations showing how the algorithm makes decisions
- Make the interface responsive for mobile devices

Added Features:

- Save and load different item sets
- Compare solutions from different algorithms
- Add the fractional knapsack problem (where you can take parts of items)

Learning Tools:

- Add tutorial mode with explanations of each step
- Include practice problems with solutions
- Show complexity analysis of different approaches

Technical Improvements:

- Optimize code for better performance with large item sets
- Add local storage to remember user inputs
- Create a shareable link feature to send problems to others