Design and Analysis of Algorithms

# Knapsack Algorithm

## Discussion on the Data Structure used and Its Effectiveness

### Data Structures Used for Item Management

The code utilizes the following data structures for item management:

- **Vectors**
  The code uses `std::vector` to store the weights, profits, and ratios of the items. Vectors are dynamic arrays that can grow or shrink in size as elements are added or removed. They provide efficient insertion, deletion, and random access operations.
- **Arrays**
  Although not explicitly used, the `std::vector` class is implemented using arrays under the hood. The code uses arrays to store the weights, profits, and ratios of the items.

### Algorithm's Effectiveness in Maximizing Resource Use

The algorithm used in the code is a greedy algorithm, specifically the **Fractional Knapsack Algorithm**. The algorithm's effectiveness in maximizing resource use can be evaluated as follows:

- **Time Complexity**
  The time complexity of the algorithm is O(n^2) due to the nested loops used for sorting the items based on their profit-to-weight ratios. This is not the most efficient time complexity, especially for large inputs.
- **Space Complexity**
  The space complexity of the algorithm is O(n) as it uses vectors to store the weights, profits, and ratios of the items.

- **Optimality**
  The Fractional Knapsack Algorithm is optimal for fractional knapsack problems, where items can be divided into fractions. However, it may not be optimal for 0/1 knapsack problems, where items can only be taken or left behind.
- **Greedy Choice Property**
  The algorithm makes the locally optimal choice at each step, which is to select the item with the highest profit-to-weight ratio. This greedy choice property ensures that the algorithm maximizes the total profit.

## Advantages of the Algorithm

- **Simple to Implement**
  The Fractional Knapsack Algorithm is relatively simple to implement, especially when compared to more complex algorithms like dynamic programming.
- **Fast Execution**
  The algorithm has a relatively fast execution time, especially for small to medium-sized inputs.

## Disadvantages of the Algorithm

- **Limited Applicability**
  The algorithm is only applicable to fractional knapsack problems and may not be suitable for 0/1 knapsack problems.
- **Not Optimal for All Cases**
  The algorithm may not always produce the optimal solution, especially for cases where the items have similar profit-to-weight ratios.