|  |  |
| --- | --- |
|  | Department of Computer Science and Engineering  Chandpur Science and Technology University |

**LAB-05**

**Course Title**: Algorithm Design and Analysis Sessional

**Course Code**:CSE 2202

**Submitted To-**

**Mostafiz Ahammed**

*Lecturer****Department of Computer Science and Engineering****Chandpur Science and Technology University*

**Submitted By-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl** | Class Roll | Name | Batch | Signature |
| 01 | B210101022 | Md. Rabbi Ahammed | CSE-21 | rabbi |

**Date of Submission: July, 2025**

Experiment 01: Knapsack Problem Using Greedy Approach

# Objective

To implement the 0/1 Knapsack problem using a Greedy approach (Fractional Knapsack) and evaluate its correctness in terms of time complexity, space, and applicability.

# Algorithm (Greedy Approach - Fractional Knapsack)

- Calculate the value-to-weight ratio for each item.  
- Sort the items based on this ratio in descending order.  
- Select items greedily:  
 - If the entire item can be included, add it.  
 - Otherwise, include the fraction that fits and break.

# Theoretical Solution of Given Problem

The greedy approach only works optimally for the fractional knapsack problem. It fails for the 0/1 Knapsack in some cases due to not exploring all combinations. This solution runs in O(n log n) due to sorting.

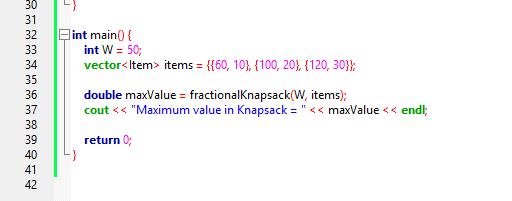
# Practical Work

## a. Pseudocode

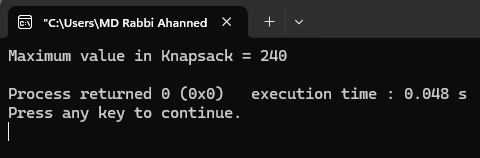
function fractionalKnapsack(W, items):  
 for each item in items:  
 compute value-to-weight ratio  
  
 sort items by ratio in descending order  
  
 totalValue = 0  
 for item in sorted items:  
 if item.weight <= W:  
 W -= item.weight  
 totalValue += item.value  
 else:  
 totalValue += item.value \* (W / item.weight)  
 break  
  
 return totalValue

**b. Source Code (C++) :**

# 

****

**Output :**

****

# Analysis Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Best Case | Worst Case | Avg Case | Space |
| Fractional Knapsack | O(n log n) | O(n log n) | O(n log n) | O(n) |

# Observations

- The greedy method is efficient and fast.  
- Works optimally only for the fractional knapsack problem.  
- Doesn't guarantee optimality for 0/1 knapsack.

# Challenges

- Ensuring accurate floating-point calculations.  
- Understanding the limitation of greedy approach in 0/1 knapsack.  
- Sorting by ratio efficiently and correctly handling fractional parts.

# Conclusion

The greedy approach provides a fast and efficient solution for the fractional knapsack problem, but it is not suitable for the 0/1 knapsack problem. For the latter, dynamic programming is necessary for optimal results. Nonetheless, for approximate or real-time needs, greedy works well.