

# Marwadi University Faculty of Technology

## **Department of Information and Communication Technology**

Subject: DAA (01CT0512) AIM: 1/0 Knapsack using Greedy Approach

Experiment No: 14 Date: 12/9/2023 Enrolment No: 92100133020

## 1/0 Knapsack using Greedy Approach:

Greedy approach does not guarantee the optimal solution for 0/1 knapsack as it chooses items based on immediate benefit without considering the overall weight constraints.

### Algorithm:

Select items in decreasing order of their values and include them in the knapsack if they fit, until the knapsack is full.

#### Code:

```
#include <iostream>
#include <algorithm>
using namespace std;
struct Item {
  int weight, value;
};
bool comparison(Item a, Item b) {
  return (a.value > b.value);
}
int knapsackGreedy(Item items[], int n, int capacity) {
  sort(items, items + n, comparison);
  int totalValue = 0, currentWeight = 0;
  for (int i = 0; i < n; i++) {
     if (currentWeight + items[i].weight <= capacity) {
       currentWeight += items[i].weight;
       totalValue += items[i].value;
    }
  }
  return totalValue;
int main() {
  Item items[] = \{\{10, 60\}, \{20, 100\}, \{30, 120\}\};
  int capacity = 50;
  int n = sizeof(items) / sizeof(items[0]);
  cout << "Maximum value in Knapsack: " << knapsackGreedy(items, n, capacity);</pre>
  return 0;
}
```



# Marwadi University Faculty of Technology

# **Department of Information and Communication Technology**

Subject: DAA (01CT0512) AIM: 1/0 Knapsack using Greedy Approach

Maximum value in Knapsack: 220

Justification:\_\_\_\_\_

Experiment No: 14 Date: 12/9/2023 Enrolment No: 92100133020

## **Output:**

Space complexity:	
Justification:	
Time complexity:	
Best case time complexity:	
Justification:	
Worst case time complexity:	