

|  |  |
| --- | --- |
| **Name** | Vedant Deshmukh |
| **UID No.** | 2021300025 |
| **Class** | COMPS A (B batch) |
| **Experiment No.** | 5 |

**Aim:** Fractional Knapsack problem

# Theory:

* **Greedy Alogrithms**

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.

The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.

# Algorithm:

knap\_sack\_greedy(W, n, weight[], profit[]):

io[] = new float[n] // array to store the fraction of items taken ratio[] = new RatioIndex[n] // array of RatioIndex pointers

// calculate ratio of profit to weight for each item for i = 0 to n-1:

ratio[i] = new RatioIndex() ratio[i]->r = profit[i] / weight[i] ratio[i]->index = i

// sort items by ratio in descending order sort\_descending(ratio, n)

try\_w = 0 // current weight of items included in the knapsack total\_profit = 0 // total profit obtained from included items



// iterate over the sorted items for i = 0 to n-1:

index = ratio[i]->index

try\_w = try\_w + weight[index] io[index] = 1 // take the entire item

// if the knapsack is now overfilled, take a fraction of the item instead if try\_w > W:

try\_w = try\_w - weight[index] space\_needed = W - try\_w

portion\_taken = space\_needed / weight[index] io[index] = portion\_taken

total\_profit = total\_profit + (portion\_taken \* profit[index]) else:

total\_profit = total\_profit + profit[index]

// create a new Result object to store the output res = new Result

res->io\_array = io

res->total\_profit = total\_profit res->total\_weight = try\_w

return res

# Code:



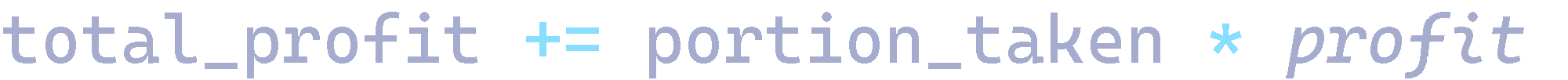




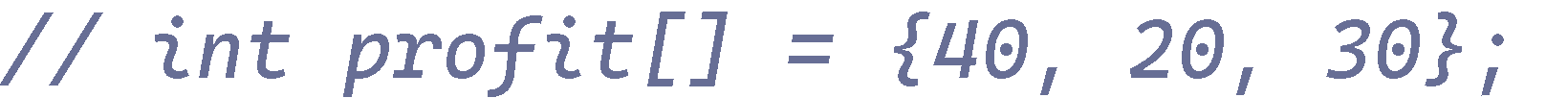
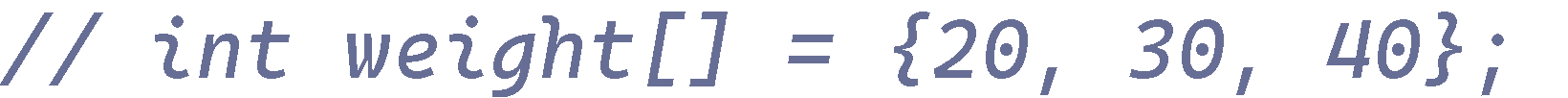






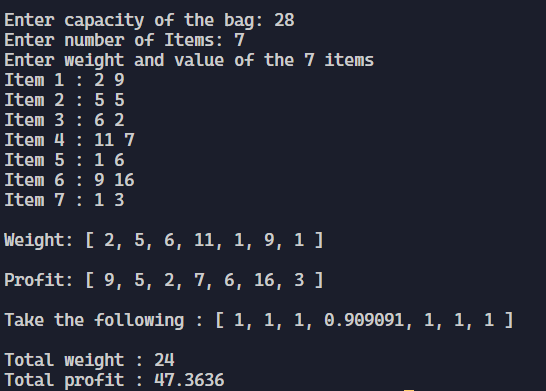






**Output:**





# Observation:

The greedy algorithm can easily solve fractional knapsack problem in O(n log n) time

# CONCLUSION:

After conducting this experiment, I have learnt how to use greedy approach to solve the fractional knapsack problem.