Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058, India
(Autonomous College Affiliated to University of Mumbai)

Experiment No.	6
Aim	To implement a program based on greedy approach
Name	Harsh Mukesh Jain
UID No.	2021300048
Class & Division	SE Comps A (C – Batch)
Date of Performance	27-03-2023
Date of Submission	3-04-2023

Theory/Experiment:

Greedy Alogrithm:

The greedy method is one of the strategies like Divide and conquer used to solve the problems. This method is used for solving optimization problems. An optimization problem is a problem that demands either maximum or minimum results. Let's understand through some terms.

The Greedy method is the simplest and straightforward approach. It is not an algorithm, but it is a technique. The main function of this approach is that the decision is taken on the basis of the currently available information. Whatever the current information is present, the decision is made without worrying about the effect of the current decision in future.

This technique is basically used to determine the feasible solution that may or may not be optimal. The feasible solution is a subset that satisfies the given criteria. The optimal solution is the solution which is the best and the most favorable solution in the subset. In the case of feasible, if more than one solution satisfies the given criteria then those solutions will be considered as the feasible, whereas the optimal solution is the best solution among all the solutions.

For example consider the Fractional Knapsack Problem. The local optimal strategy is to choose the item that has maximum value vs weight ratio. This strategy also leads to a globally optimal solution because we are allowed to take fractions of an item.

Knapsack Problem :

A list of items is given, each item has its own value and weight. Items can be placed in a knapsack whose maximum weight limit is W. The problem is to find the weight that is less than or equal to W, and value is maximized.

There are two types of Knapsack problem :

1.0 – 1 Knapsack
2.Fractional Knapsack

For the 0 – 1 Knapsack, items cannot be divided into smaller pieces, and for fractional knapsack, items can be broken into smaller pieces.

Fractional Knapsack Problem :

Algorithm :

```
Begin
  sort the item list based on the ration of value and weight
  currentWeight := 0
  knapsackVal := 0

  for all items i in the list do
    if currentWeight + weight of item[i] < weight then
      currentWeight := currentWeight + weight of item[i]
      knapsackVal := knapsackVal + value of item[i]
    else
      remaining := weight – currentWeight
      knapsackVal “= knapsackVal + value of item[i] * (remaining/weight of item[i])
      break the loop
  done
End
```

Code :

```
#include <stdio.h>
int main()
{
   int n, maxKnap, i;
   float sum = 0, maxVal = 0, lastItem;
   printf("Enter numer of items:");
   scanf("%d",&n);
   int W[n], V[n], I[n], ans[n];
   for(int i = 0; i < n; i++) {
      printf("For %d item ~\n", i+1);
      printf("Enter value of weight (in kg):");
      scanf("%d",&W[i]);
      printf("Enter value for %d kg weight:",W[i]);
      scanf("%d",&V[i]);
      I[i] = i+1;
   }
   for(int i = 0; i < n - 1; i++) {
      for(int j = 0; j < n - 1; j++) {
         float valToWeight = (float)(V[j])/W[j];
         if(valToWeight < (float)V[j+1]/W[j+1]) {
            int temp = W[j];
            W[j] = W[j+1];
            W[j+1] = temp;

            temp = V[j];
            V[j] = V[j+1];
            V[j+1] = temp;

            temp = I[j];
            I[j] = I[j+1];
            I[j+1] = temp;
         }
      }
   }
   printf("-------------------------------------------\nEnter maximum weight limit:");
   scanf("%d",&maxKnap);
   for(i = 0; i < n; i++) {
      sum += W[i];
      maxVal += V[i];
```

```
        if(maxKnap >= sum) {
            ans[i] = I[i];
        }
        else if(sum > maxKnap) {
            sum = sum - W[i];
            maxVal = maxVal - V[i];
            int req = maxKnap - sum;
            sum = sum + req;
            maxVal = maxVal + (req * V[i])/(float)W[i];
            lastItem = (req)/(float)W[i];
            // printf("%f, %f",sum,maxVal);
            printf("Maximum weight that can be obtain: %0.2f\nMaximum value that can be obtain:
%0.2f",sum,maxVal);
            break;
        }
        else break;
    }

    printf("\nItems in Order:\n");
    for(int j = 0; j < i; j++) {
        printf("%d  ",ans[j]);
    }
    printf("%0.2f of %d",lastItem, ans[i]);

    return 0;
}
```

Output :

```
Enter numer of items:6
For 1 item ~
Enter value of weight (in kg):100
Enter value for 100 kg weight:40
For 2 item ~
Enter value of weight (in kg):50
Enter value for 50 kg weight:35
For 3 item ~
Enter value of weight (in kg):40
Enter value for 40 kg weight:20
For 4 item ~
Enter value of weight (in kg):20
Enter value for 20 kg weight:4
For 5 item ~
Enter value of weight (in kg):10
Enter value for 10 kg weight:10
For 6 item ~
Enter value of weight (in kg):10
Enter value for 10 kg weight:6
-----------------------------------------
Enter maximum weight limit:100
Maximum weight that can be obtain: 100.00
Maximum value that can be obtain: 66.00
Items in Order:
5  2  6  0.75 of 3

...Program finished with exit code 0
Press ENTER to exit console.
```

The time complexity of this algorithm is O(n Log n).

CONCLUSION :

By performing this experiment, I understood the concept of greedy algorithm and based on it solved a problem known as Fractional Knapsack Problem.
The fractional knapsack problem is a classic optimization problem where a set of items with different weights and values are given, and the goal is to select a subset of these items that can fit into a knapsack of a given capacity, such that the total value of the selected items is maximized.