LAB 8

Fractional Knapsack Problem

Name : Rahul Thapar ID : 1410110321 Date : 22 Feb, 2017

SCREENSHOT:

```
rahthap@rahthap-Inspiron-3521:~/Desktop/Lab8$ subl lab8.c
rahthap@rahthap-Inspiron-3521:~/Desktop/Lab8$ gcc lab8.c -o lab8
rahthap@rahthap-Inspiron-3521:~/Desktop/Lab8$ ./lab8
        Implementation of Fractional Knapsack Problem
        ______
         Number of objects : 6
         COST of each object - 12 1 2 1 4 10
         PROFIT on each corresponding object - 4 2 2 1 10 15
        Total Weight of the BAG: 15
                                VALUE : 10
       Object Added : 5
                                            PROFIT : 1
                                                PROFIT: 4
                                                                Space Left: 11
        Object Added : 2
                                                                Space Left: 10
                               VALUE : 2
       Object Added : 6
                                VALUE : 15
                                               PROFIT : 10
                                                                Space Left: 0
Final VALUE in the bag : 27.00.
rahthap@rahthap-Inspiron-3521:~/Desktop/Lab8$
```

CODE:

/*

@author: Rahul Thapar

ID: 1410110321

Problem: Implement Fractional Knapsack Problem

Algorithm

1. Calculate DENSITY(PROFIT) : value per weight for each

item

2. Sort the items as per the value density in descending

order 3. Take as much item as possible not already taken in the knapsack */ #include <stdio.h> int n; int cost[50]; int value[50]; int W; int q; void knapsack_fill() { int current_weight; float total_value; int i, maximum_i; int used[10]; for (i = 0; i < n; ++i)used[i] = 0;current weight = W; while (current_weight > 0) { // While the bag is NOT full : add // Find the suitable object to ADD maximum i = -1; for (i = 0; i < n; ++i)if $((used[i] == 0) &&((maximum_i == -1) ||$ ((float)value[i]/cost[i] > (float)value[maximum_i]/cost[maximum_i]))) $maximum_i = i$; used[maximum i] = 1; // Maximum value used current_weight -= cost[maximum_i];

total_value += value[maximum_i];

if (current_weight >= 0)

```
printf("\tObject Added : %d\t VALUE : %d\t
PROFIT: %d\t Space Left: %d\n", maximum i + 1,
value[maximum i], cost[maximum i], current weight);
          else {
             printf("\tObject Added : %d\t VALUE : %d\t
PROFIT
             %d\t
                            Left :
                                      %d\n",
                    Space
                                               (int)((1
(float)current_weight/cost[maximum_i]) * 100), value[maximum_i],
cost[maximum_i], maximum_i + 1);
             total value -= value[maximum i];
             total value
                                             (1
                                                        +
(float)current_weight/cost[maximum_i]) * value[maximum_i];
          }
      }
      printf("\n\n");
      printf("Final VALUE in the bag : %.2f.\n\n", total value);
   }
   int main(){
      printf("\n\n");
      printf("\tlmplementation of
                                     Fractional
                                                  Knapsack
Problem\n");
=========\n\n");
      printf("\t Number of objects : ");
      scanf("%d",&n);
      printf("\t COST of each object - ");
      for(q=0;q< n;q++){
          scanf("%d",&cost[q]);
      }
      printf("\t PROFIT on each corresponding object - ");
      for(q=0;q< n;q++){
          scanf("%d",&value[q]);
      }
      printf("\tTotal Weight of the BAG : ");
      scanf("%d",&W);
```

```
printf("\n\n");
knapsack_fill();
return 0;
}
```