CSI2003 – Advanced Algorithm

Problem in Greedy Algorithm

Knapsack Problem

NAME : ARAVIND S

REG. NO. : 21MID0229

SLOT : A1 / L49+L50

DATE : 05.01.2023

Problem: (To be solved using Greedy algorithm)

Object/n	1	2	3	4	5	6	7
Profit	10	15	7	8	9	5	4
Weight	1	3	5	4	1	3	2
Pro/weigh	10	5	1.4	2	9	1.66	2

Find the objects that can be placed in a bag that has a maximum capacity of 15 units with the maximum profit that can be gained out of it.

Manual Solution: (optimal solution with max. profit)

We take the object with max profit per weight first.

Order of object with max profit/weight ratio : **{1, 5, 2, 4, 7, 6, 3}**
Their weights : **{1 +1 +3 +4 +2 +3 +1}**
Respective Profits : **{10+9+15+8+4+5+1.4}**

(only one unit weight of the 3rd object is taken because it overhead the max capacity),
(7/5=1.4 for the 3rd object)

Table form:

Order	1	5	2	4	7	6	3
Ratio	10	9	5	2	2	1.66	1.4
Weight	1	1	3	4	2	3	1*
Profit	10	9	15	8	4	5	1.4*

Total Profit = 10+9+15+8+4+5+1.4 = **52.4**

Code: (implemented in C)

```c
#include<stdio.h>
int main()
{
    float profit[7]={10,15,7,8,9,5,4}; //profit array
    float weight[7]={1,3,5,4,1,3,2}; //respective weight array
    float prowei[7]; //array to store the profit/weight ratio
    //profit and weight can also be taken through input
    int w = 0; //to store the weight
    int a,b; //assisting variables
    float pro=0; //total profit

    for(int i=0;i<7;i++)
    {
        prowei[i]=profit[i]/weight[i]; //the array containing the profit/weight ratios
        //printf("%f ",prowei[i]);
    }
    printf("Order               : ");
    for(int i=0;i<7;i++)
    {
        float temp=0; //temporary variable
        int max=0; //to find which object has the highest pro/weight ratio
        a=0; //to find the overflow in weight

        for(int j=0;j<7;j++)
        {
            if(prowei[j]>temp) //comparing which obj has the highest ratio
            {                  //similar to sorting
                max=j;         //index of the highest ratio is stored
                temp=prowei[j]; //temp stores the value of highest ratio
            }
        }

        printf("%d ",max+1); //prints the sequence of pro/weig in descending order
        w = w + weight[max]; /*calculates the total weight,
                              respective weights are added sequentially*/
        if(w>15) //if weight goes beyond 15 kgs
        {
            w=w-weight[max]; //we subtract the last added weight
            a=15-w; //to know how much excess has been added
            b=max; //obj number which causes overload
        }
        pro=pro+profit[max]; //the profit gained by each obj is added sequentially
        prowei[max]=0;
```

```
        //the element in pro/weig array is assigned zero so next largest number is selected
    }
    //printf("%f", profit[b] / weight[b]); //to find the pro/weig of the overloading object
        pro = pro + (profit[b] / weight[b])-profit[b];
        // the last obj is added partially to the total profit
        printf("\nweight distribution : %d + %d \ntotal profit        : %f",w,a,pro);
       //print final result
}
```

Clean code without comments:

```
#include<stdio.h>
int main()
{

    float profit[7]={10,15,7,8,9,5,4};
    float weight[7]={1,3,5,4,1,3,2};
    float prowei[7];

    int w = 0;
    int a,b;
    float pro=0;

    for(int i=0;i<7;i++)
    {
        prowei[i]=profit[i]/weight[i];
    }

    printf("Order    : ");

    for(int i=0;i<7;i++){
        float temp=0;
        int max=0;
        a=0;

        for(int j=0;j<7;j++)
        {
            if(prowei[j]>temp)
           {
                max=j;
                temp=prowei[j];
            }
        }

        printf("%d ",max+1);
        w = w + weight[max];
```

```
        if(w>15)
        {
            w=w-weight[max];
            a=15-w;
            b=max;
        }
        pro=pro+profit[max];
        prowei[max]=0;
    }
        pro = pro + (profit[b] / weight[b])-profit[b];
        printf("\nweight distribution : %d + %d \ntotal profit: %.2f",w,a,pro);
}
```

Output Screenshot:

```
PS C:\Users\Aravind\OneDrive\Documents\VScode\sem 4\AA>
 .\greedy }
Order               : 1 5 2 4 7 6 3
weight distribution : 14 + 1
total profit        : 52.40
```

(∴ same result)

Bonus Question:

Using your learnt knowledge in the course CSI2003, answer the following:
Name any one unique (one and only) difference between greedy algorithm with any other algorithm.

- A greedy algorithm is an algorithmic paradigm that follows the problem-solving heuristic of **making the locally optimal choice at each stage** with the hope of finding a global optimum.
- One key difference between greedy algorithms and other types of algorithms is that greedy algorithms **do not use backtracking**. This means that once a decision is made, it is not revisited or changed
- Another difference is that greedy algorithms are only suitable for problems that have **specific structure** that allows locally optimal choices to lead to a globally optimal solution. Eg. Knapsack Problem.

Aravind S - 21MID0229