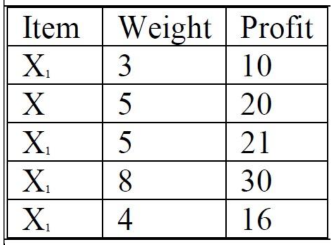
Bansilal Ramnath Agarwal Charitable Trust's
Vishwakarma Institute of Information Technology

Department of
Artificial Intelligence and Data Science

Name: Siddhesh Dilip Khairnar		
Class: TY	Division: B	Roll No: 372028
Semester: V	Academic Year: 2023-2024	
Subject Name & Code: Design and Analysis of Algorithm: ADUA31202		

Title of Assignment: Solve the following instance of the knapsack problem given the knapsack capacity in w=20 using greedy methods. The total number of items is 5.



Item	Weight	Profit
X_1	3	10
X	5	20
X_1	5	21
X_1	8	30
X_1	4	16

Date of Performance: 10-09-2023	Date of Submission: 16-09-2023

ASSIGNMENT NO. 3

DAA Assignment no: 3

Aim: Solve the following instance of the knapcack problem given the knapsack capacity in $w = 20$ using greedy method. The total no. of item is 5.

Item	weight	Profit
X_1	3	10
X_2	5	20
X_3	5	21
X_4	8	30
X_5	4	16

Background Information :→

* knapsack problem using greedy method :

The selection of some thing each with profit and weight values to be packed into one or more knapsack with capacity is the fundamental idea behind all families of knapsack problem. The knapsack problem had two version that are as follow.

1. Fractional knapsack problem
2. 0/1 knapsack problem.

The fractional knapsack problem using the greedy method is an efficient method to solve its, where you need to sort the item according of their ratio of value weight. In a fractional knapsack. we can break item to maximise the knapsack total value this problem is which we can break an item is also called the fractional knapsack problem.

Page No.
Date

* What is knapsack problem using greedy method?

In this method, the knapsack filling done so that the maximum capacity of the knapsack is utilized so that maximum profit can be earned from it. The knapsack problem using the greedy method is referred to as.

Given a list of n object {I1, I2, ... IN} & a knapsack (or bag)
The Capacity of knapsack is M.
Each object I_j has weight w_j & a profit P_j
if a fractional x_j (where $x \in \{0, \cdots 1\}$) of an object I_j is placed into a knapsack then a profit of $P_i x_j$ is earned.
The problem (or objective) is to fill the knapsack (up to maximum capacity m). maximizing the total profit earned.

Mathematically :→
maximise (the profit) $= \sum_{j=1}^{n} P_i x_j$

$$= \sum_{j=1}^{n} w_j x_j \le M \text{ and } x_j \in \{0, \cdots 1\}$$
$$1 \le j \le n$$

* knapsack problem using Greedy method :→

Note that the value of x_j will be only value b/w 0 and 1 (inclusive). If any object J is completed placed into a knapsack it's value is 1 ($x_j = 1$). If we do not pick (or select) that object to fill into a knapsack it's value 0 ($x_j = 0$): otherwise if we take a ~~fraction~~ fraction of any object then its value will be any value between 0 and 1.

Page No.
Date

Software Requirement :→ Vscode

Conclusion :→

Solved the problem using knapsack greedy algorithm getting a final answer of 79.5 (maximum profit)

✓ Ⓒ [illegible] 16/09/23

Aim: Solve the following instance of the knapsack problem given the knapsack capacity in w=20 using greedy methods. The total number of items is 5.

Item	Weight	Profit
X_1	3	10
X	5	20
X_1	5	21
X_1	8	30
X_1	4	16

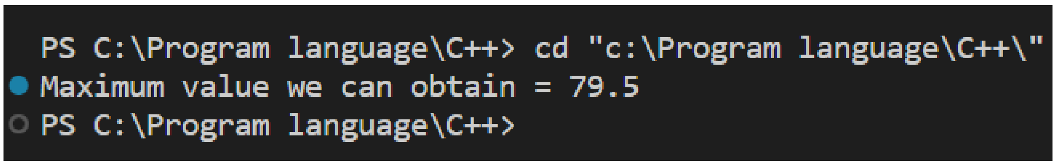
Problem Statement: Use the knapsack greedy method to find maximum profit.

Program Code:

```
#include <iostream>
#include <algorithm>

using namespace std;

struct Item
{
    int value, weight;

    Item(int value, int weight)
    {
        this->value = value;
        this->weight = weight;
    }
};

bool cmp(struct Item a, struct Item b)
{
    double r1 = (double)a.value / (double)a.weight;
    double r2 = (double)b.value / (double)b.weight;
    return r1 > r2;
}
double fractionalKnapsack(int W, struct Item arr[], int N)
{
    sort(arr, arr + N, cmp);

    double finalvalue = 0.0;
    for (int i = 0; i < N; i++)
    {
        if (arr[i].weight <= W)
        {
            W -= arr[i].weight;
            finalvalue += arr[i].value;
        }
```

```cpp
        else
        {
            finalvalue += arr[i].value * ((double)W / (double)arr[i].weight);
            break;
        }
    }
    return finalvalue;
}
int main()
{
    int W = 20;
    Item arr[] = {{10, 3}, {20, 5}, {21, 5}, {30, 8}, {16, 4}};

    int N = sizeof(arr) / sizeof(arr[0]);

    cout << "Maximum value we can obtain = " << fractionalKnapsack(W, arr, N);
    return 0;
}
```

Result:



```
  PS C:\Program language\C++> cd "c:\Program language\C++\"
● Maximum value we can obtain = 79.5
○ PS C:\Program language\C++>
```

Conclusion: Solved the problem using knapsack greedy algorithm, getting a final answer of 79.5 (maximum profit).