	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology **Department of** **Artificial Intelligence and Data Science**

Name: **Sahil Dilip Sonawane**

Class: **TY**	Division: **C**	Roll No: **373057**

Semester: **V**	Academic Year: **2023-24**

Subject Name & Code: **Design and Analysis of Algorithm**

Title of Assignment**: Solve the following instance of the knapsack problem given the knapsack capacity in w=20 using greedy methods. The total no 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

Assignment No. 3

Aim:
Solve the following instance of the knapsack problem given the knapsack capacity in w=20 using greedy methods. The total no 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.

Background Information:

Knapsack Problem Using Greedy Method:

The selection of some things, each with profit and weight values, to be packed into one or more knapsacks with capacity is the fundamental idea behind all families of knapsack problems. The knapsack problem had two versions that are as follows:

1. Fractional Knapsack Problem
2. 0 /1 Knapsack Problem

The fractional Knapsack problem using the Greedy Method is an efficient method to solve it, where you need to sort the items according to their ratio of value/weight. In a fractional knapsack, we can break items to maximize the knapsack's total value. This problem in which we can break an item is also called the Fractional knapsack problem.

What is Knapsack Problem Using Greedy Method?

In this method, the Knapsack's filling is 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 objects, say {I1, I2,……, In) and a knapsack (or bag).

The capacity of the knapsack is M.

Each object Ij has a weight wj and a profit of pj

If a fraction xj (where x ∈ {0...., 1)) of an object Ij is placed into a knapsack, then a profit of pjxj is earned.

The problem (or Objective) is to fill the knapsack (up to its maximum capacity M), maximizing the total profit earned.

Mathematically:

$$\text{Maximize (the profit)} = \sum_{j=1}^{n} p_j x_j$$

$$= \sum_{j=1}^{n} w_j x_j \leq M \text{ and } x_j \in \{0, \ldots., 1\}, 1 \leq j \leq n$$

Knapsack Problem Using Greedy Method

Note that the value of xj will be any value between 0 and 1 (inclusive). If any object Ij is completely placed into a knapsack, its value is 1 (xj = 1). If we do not pick (or select) that object to fill into a knapsack, its value is 0 (xj = 0). Otherwise, if we take a fraction of any object, then its value will be any value between 0 and 1.

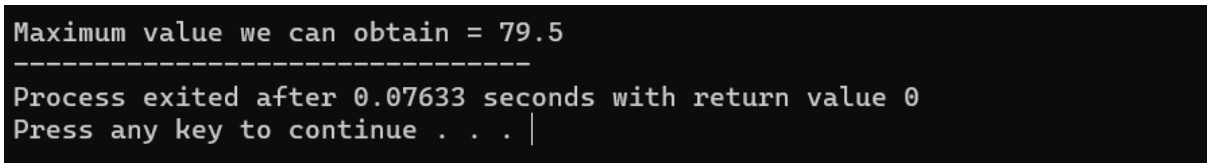
Software Requirements:

VS code, GDB Compiler

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;
        }

        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;
}
```

Output:



```
Maximum value we can obtain = 79.5
--------------------------------
Process exited after 0.07633 seconds with return value 0
Press any key to continue . . .
```

Conclusion:

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