NIRMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

NAAC ACCREDITED 'A' GRADE

NAME: - Darji Akshatkumar Hiteshbhai

RollNo: - 23MCD001

Branch: - M.tech-CSE**(Data Science)**

Subject: - Complexity Theory & Algorithms

Practical-6

Aim: Implement Fraction Knapsack problem using Greedy approach.

Code for Fractional Knapsack –

```
#include<bits/stdc++.h>
using namespace std;

void knapsackProblem(map<float, pair<int, int>, greater<float>> &final, int
object){
    int M;
    cout << endl;
    cout << "Enter the Knapsack capacity : ";
    cin >> M;
    float p = 0;
    for(auto it : final){
        if(M > 0 && (it.second.second <= M)){
            M -= it.second.second;
            p += it.second.first;
        }
        else{
            if(M > 0){
                p += it.second.first * (static_cast<float>(M) /
it.second.second);
            }
            break;
        }
    }
    cout << "----------------------------------------";
    cout << endl << "Maximum Profit : " << p;
}
int main(){
    int object;
    cin >> object;
    vector<int> profit(object);
    vector<int> weight(object);
    vector<pair<int, int>> pw;
    vector<float> pw_ratio;
    map<float, pair<int, int>, greater<float>> final;
    for(int i=0;i<object;i++){
        cin >> profit[i];
        cin >> weight[i];
    }
```

```
    for(int i=0;i<object;i++){
        for(int j=i;j<i+1;j++){
            pw.push_back({profit[i], weight[i]});
        }
    }
    cout << endl;
//    for(auto it : pw){
//        cout << "[profit: " << it.first << ", weight: " << it.second << "]";
//    }
    for(int i=0;i<object;i++){
        float result = static_cast<float>(pw[i].first) /
static_cast<float>(pw[i].second);
        pw_ratio.push_back(result);
    }
//    for(auto it : pw_ratio){
//        cout << it << ", ";
//    }
    for(int i=0;i<object;i++){
        final[pw_ratio[i]] = {pw[i].first, pw[i].second};
    }
    cout << "Knapsack Table is as follows : ";
    cout << endl;
    for(auto it : final){
        cout << "[p/w: "<< it.first << ", (Profit: " << it.second.first << ",
Weight: " << it.second.second << ")]";
    }
    knapsackProblem(final, object);
    return 0;
}
```

Output –

Test Case – 1

```
Run    knapsack.cpp
H:\Nirma\CTA\Practical-6\knapsack.exe
3
25 18
24 15
15 10

Knapsack Table is as follows :
[p/w: 1.6, (Profit: 24, Weight: 15)][p/w: 1.5, (Profit: 15, Weight: 10)][p/w: 1.38889, (Profit: 25, Weight: 18)]
Enter the Knapsack capacity :20
 ----------------------------------------
Maximum Profit : 31.5
Process finished with exit code 0
```

Test Case – 2

```
Run    knapsack.cpp
H:\Nirma\CTA\Practical-6\knapsack.exe
3
60 10
100 20
120 30

Knapsack Table is as follows :
[p/w: 6, (Profit: 60, Weight: 10)][p/w: 5, (Profit: 100, Weight: 20)][p/w: 4, (Profit: 120, Weight: 30)]
Enter the Knapsack capacity :50
 ----------------------------------------
Maximum Profit : 240
Process finished with exit code 0
```

Test Case – 3

```
Run    knapsack.cpp
H:\Nirma\CTA\Practical-6\knapsack.exe
2
60 10
100 20

Knapsack Table is as follows :
[p/w: 6, (Profit: 60, Weight: 10)][p/w: 5, (Profit: 100, Weight: 20)]
Enter the Knapsack capacity :50
 ----------------------------------------
Maximum Profit : 160
Process finished with exit code 0
```

Test Case – 4

```
Run    knapsack.cpp
H:\Nirma\CTA\Practical-6\knapsack.exe
4
3 6
6 1
1 5
4 3

Knapsack Table is as follows :
[p/w: 6, (Profit: 6, Weight: 1)][p/w: 1.33333, (Profit: 4, Weight: 3)][p/w: 0.5, (Profit: 3, Weight: 6)][p/w: 0.2, (Prof
it: 1, Weight: 5)]
Enter the Knapsack capacity :10
 ----------------------------------------
Maximum Profit : 13
Process finished with exit code 0
```