**Task:-Fractional Knapsack(greedy approach)**

**Objective:-**

The objective of our experiment is to devise a greedy approach to solve the knapsack problem such that the output is an array having fraction of each article included in the bag.

**Introduction:-**

The knapsack problem or more formally, fractional knapsack problem is a problem which can be solved by greedy algorithm. The problem statement of the knapsack is that we are given a knapsack of given size and some objects n of given weight. Each object has some profit which we will get if the object is added to the bag. We need to fill the knapsack by adding these objects to the knapsack, completely or fractionally, such that the net profit we get is maximized. From the problem statement it is clear that it is an optimization problem in which we have to maximize the profit, thus it can be solved using the greedy technique of algorithm design. The idea behind maximizing the profit is to obtain the profit by weight ratio of each object and then sort the list in decreasing order. After that we will pick the item and fill the bag until either we are left with no items or the bag gets completely filled. The greedy technique of algorithm design generally consists of an objective function and a feasibility criterion. Here in case of the knapsack problem, the objective function is to maximize the net profit obtain whereas the feasibility criterion is that the net weight of the bag should never exceed its capacity. There are various techniques of solving the problems which can pass the feasibility criterion like choosing the item with minimum weight or selecting an item with maximum profit but the profit obtained through these approaches can never be optimal. Such solutions provide sub optimal solutions because although they pass the feasibility criterion but they are not fulfilling the objective function. The best technique is to sort the list on the basis of their P/W and then fill in the bag with until no item is left or the bag fills up to maximum capacity.

**Algorithm:-**

while (m > 0)

{

if (w[i] <= m){

m = m - w[i]

b[i] = 1;

q = q + a[i].first \* w[i];

i--;

}

else

{

b[i] = m / w[i];

q = q + a[i].first \* m;

i--;

}

}

**Procedure:-**

* First we take input of weight and the respective profit array.
* Then we store the profit per unit weight ration and respective weight in a vector pair.
* The we sort the vector pair such that we have the profits in decreasing order.
* Now we keep on adding the weight as per the capacity in the knapsack.
* If the given weight will be more than the current capacity of the knapsack then we will add fractional weight.
* Our main objective is to maximize the profit.

**Code:-**

#include <bits/stdc++.h>

#include <vector>

using namespace std;

int main()

{

    int n;

    cin >> n;

    vector<pair<int, int>> a;

    int p[n], w[n];

    for (int i = 0; i < n; i++)

    {

        cin >> w[i] >> p[i];

        a.push\_back(make\_pair(float(float(p[i]) / float(w[i])), w[i]));

        sort(a.begin(), a.end());

    }

    int m, i = n-1;

    float q = 0.0;

    cin >> m;

    vector<float> b(n, 0);

    while (m > 0)

    {

        if (w[i] <= m){

            m = m - w[i];

        b[i] = 1; /\* code \*/

        q = q + a[i].first \* w[i];

        i--;

    }

    else

    {

        b[i] = m / w[i];

        q = q + a[i].first \* m;

    i--;

    }

    }

for (int i = 0; i < n; i++)

{cout<<b[i]<<" ";

    /\* code \*/

}

}

**Learnings –**

From this experiment we learnt the greedy technique of algorithm design. This problem of knapsack gave us an insight on how the greedy algorithm are designed, framed and how they work. We learnt how to think about the greedy logic and after that turn the logic into code by implementing it. We learnt how this technique works on optimization problems like finding the maximum or minimum value.

**Conclusion –**

From this experiment, we learnt how to solve a greedy problem of fractional knapsack and other similar problems. Here we can conclude that for optimization based solutions greedy technique of algorithm design is a good technique.