Practical – 12

AIM: Implementation of a knapsack problem using greedy algorithm

```
// C++ program to solve fractional Knapsack Problem

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

struct Item
{
    int profit, weight;

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

static bool cmp(struct Item a, struct Item b)
{
    double r1 = (double)a.profit / (double)a.weight;
    double r2 = (double)b.profit / (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].profit;
        }

        else
        {
            finalvalue += arr[i].profit * ((double)W / (double)arr[i].weight);
            break;
        }
    }

    return finalvalue;
}
```

```
int main()
{
    int W = 50;
    Item arr[] = {{60, 10}, {100, 20}, {120, 30}};
    int N = sizeof(arr) / sizeof(arr[0]);

    cout << fractionalKnapsack(W, arr, N);
    return 0;
}
```

OUTPUT

Time analysis

Operation	Time Complexity
Sorting items by value-to-weight ratio	O(n log n)
Iterating through sorted items	O(n)
Checking knapsack capacity and adding items	O(1) (amortized per item)
Total Time Complexity	O(n log n)