**Practical – 03**

**Aim :**

Write a program to solve a fractional Knapsack problem using a greedy method.

**Program:**

import java.util.Arrays;

import java.util.Comparator;

import java.util.Scanner;

class Item {

int weight;

int value;

public Item(int weight, int value) {

this.weight = weight;

this.value = value;

}

}

public class FractionalKnapsack {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the no: ");

int n = sc.nextInt(); // Number of items

Item[] items = new Item[n];

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

System.out.println("Enter the Weight: ");

int weight = sc.nextInt();

System.out.println("Enter the Value: ");

int value = sc.nextInt();

items[i] = new Item(weight, value);

}

int capacity = sc.nextInt(); // Knapsack capacity

sc.close();

Arrays.sort(items, new Comparator<Item>() {

public int compare(Item a, Item b) {

double v1 = (double) a.value / a.weight;

double v2 = (double) b.value / b.weight;

return Double.compare(v2, v1);

}

});

double maxTotalValue = 0.0;

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

if (capacity >= items[i].weight) {

maxTotalValue += items[i].value;

capacity -= items[i].weight;

} else {

double vw = (double) items[i].value / items[i].weight;

maxTotalValue += vw \* capacity;

break;

}

}

System.out.println("maxTotalValue "+maxTotalValue);

}

}

/\*

Fractional Knapsack :

Time Complexity: O(N \* log N)

Auxiliary Space: O(N)

\*/

**Output:**

****