EXPERIMENT NO.3:- Knapsack Problem using Greedy Method

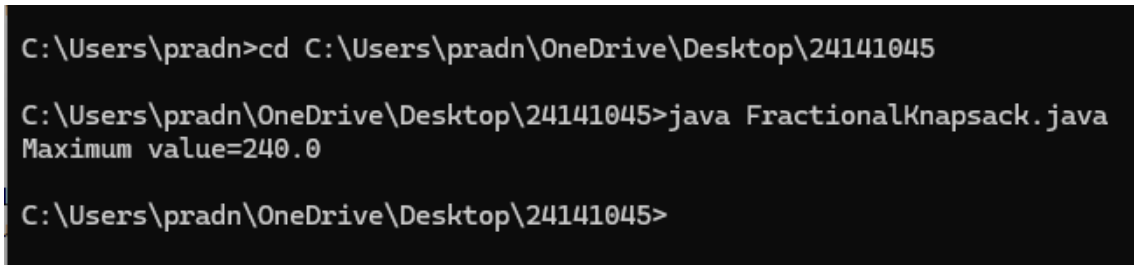
Reg No.:-24141045

Program1:-

```
import java.util.*;
class Item{
  int weight,value;
   Item (int weight, int value){
     this.weight=weight;
     this.value=value;
   }
}
 public class FractionalKnapsack{
  public static double getMaxValue(int[] weight, int[] value, int capacity){
    int n=weight.length;
    Item[] items=new Item[n];
    for(int i=0; i<n; i++){
    items[i]=new Item(weight[i],value[i]);
     }
 Arrays.sort(items, new Comparator<Item>(){
   @Override
   public int compare(Item a, Item b){
      double ratio1=(double)a.value/a.weight;
      double ratio2=(double)b.value/b.weight;
      return Double.compare(ratio2, ratio1);
  }
 });
 double totalValue = 0.0;
```

```
 int currentWeight = 0;
  for (Item item : items){
    if(currentWeight+item.weight<=capacity){
       currentWeight+=item.weight;
       totalValue+=item.value;
  }else{
    int remaining = capacity-currentWeight;
    totalValue += (double)item.value*((double)remaining/item.weight);
    break;
    }
   }
  return totalValue;
 }
  public static void main(String[] args){
    int[] weight={10,40,20,30};
    int[] value={60,40,100,120};
    int capacity=50;
    double MaxValue = getMaxValue(weight, value, capacity);
    System.out.println("Maximum value="+MaxValue);
   }
}
```

Output:-



Application:-

```
import java.util.*;
class Cargo {
   int weight, value;
   Cargo(int weight, int value) {
      this.weight = weight;
      this.value = value;
   }
}
public class CargoLoad {
   public static double getMaxValue(int[] weight, int[] value, int capacity) {
      int n = weight.length;
      Cargo[] cargos = new Cargo[n];
      for (int i = 0; i < n; i++) {
         cargos[i] = new Cargo(weight[i], value[i]);
      }
}
      Arrays.sort(cargos, new Comparator<Cargo>() {
         @Override
         public int compare(Cargo a, Cargo b) {
            double ratio1 = (double) a.value / a.weight;
            double ratio2 = (double) b.value / b.weight;
            return Double.compare(ratio2, ratio1);
         }
      });
      System.out.println("Cargo Loading Process:");
      System.out.println("-----------------------------------------------");
      double totalValue = 0.0;
```

```
        int currentWeight = 0;


        for (Cargo c : cargos) {
            if (currentWeight + c.weight <= capacity) {
                currentWeight += c.weight;
                totalValue += c.value;
                System.out.println("Loaded full cargo (weight=" + c.weight + ", value=" + c.value
+ ")");
            } else {
                int remaining = capacity - currentWeight;
                totalValue += (double) c.value * ((double) remaining / c.weight);
                System.out.println("Loaded partial cargo (weight=" + remaining + ", value="
                        + (double) c.value * remaining / c.weight + ")");
                break;
            }
        }
        System.out.println("--------------------------------------------");
        System.out.println("\nTotal loaded weight: " + currentWeight + "/" + capacity);
        return totalValue;
    }
    public static void main(String[] args) {
        int[] weight = {10, 20, 30, 40, 50};
        int[] value = {60, 100, 120, 240, 300};
        int capacity = 100;
        double maxValue = getMaxValue(weight, value, capacity);
        System.out.println("Maximum total cargo value = " + maxValue);
    }
}
```

Output:-

```
C:\Users\pradn>cd C:\Users\pradn\OneDrive\Desktop\24141045

C:\Users\pradn\OneDrive\Desktop\24141045>java CargoLoad.java
Cargo Loading Process:
-----------------------------------------------
Loaded full cargo (weight=10, value=60)
Loaded full cargo (weight=40, value=240)
Loaded full cargo (weight=50, value=300)
Loaded partial cargo (weight=0, value=0.0)
-----------------------------------------------

Total loaded weight: 100/100
Maximum total cargo value = 600.0
```

List of applications:-

1. Resource allocation and budget optimization
2. Cargo loading and logistics
3. Data compression and transmission
4. Project selection and portfolio management
5. Time management and task scheduling
6. Cloud computing and resource sharing
7. Investment and stock portfolio optimization
8. Bandwidth allocation in networks
9. Cutting stock or material utilization problems
10. Advertisement scheduling and placement optimization
11. Energy distribution in smart grids
12. Dynamic pricing and profit maximization in e-commerce