**Q-3 b**

Write a complete Greedy algorithm for the Fractional Knapsack problem. The problem is defined as follows:

Terrorist robbing a store finds n items each item worth Pi Rupees and weighs Wi kg. Terrorist has a knapsack of capacity W kg (Terrorist can carry a total weight of not more than W kg). Terrorist is allowed to carry all the quantity available of an item or a fraction thereof. What items should the Terrorist take and how much of each of those items to maximize the value of the theft?

Capacity = W,

Assume

Weight 1 = W1

Value 1 = P1

Weight 2 = W2

Value 2 = P2

Weight 3 = W3

Value 3 = P3

Goal : Fill the knapsack such that value is maximum and total weight is equal or less than Capacity W, item can be broken down to maximize the knapsack value,

Assuming,

W1 = 10kg

P1 = 60PKR

W2 = 20kg

P2 = 100 PKR

W3 = 30kg

P3 = 120PKR

W = 50

**Fractional Knapsack**

Take A, B and 2/3rd of C

Total weight: 10+20+ 30\*2/3

Total Value = 60 + 100 + 120\*2/3 = 240

**Greedy Approach:**

1. Calculate the ratio (value/weight) for each item
2. Sort the items based on this ratio
3. Take the item with highest ratio and add them until we can’t add the next item as whole
4. At the end add the next item as much(Fraction) as we can

**Time complexity:**

O(nlogn)

**import** java.util.Arrays;

**import** java.util.Comparator;

**public** **class** FractionalKnapSack {

**public** **static** **void** main(String[] args) {

**int** wt[] = {10,20,30};

**int** val[] = {60,100,120};

**int** capacity = 50;

**double** maxValue = *getMaxValue*(wt,val,capacity);

System.***out***.println(+maxValue);

}

**static** **double** getMaxValue(**int**[] wt,**int** [] val, **int** capacity)

{

ItemValue[] iVal = **new** ItemValue[wt.length];

**for**(**int** i = 0;i < wt.length;i++)

{

iVal[i] = **new** ItemValue(wt[i], val[i], i);

}

Arrays.*sort*(iVal, **new** Comparator<ItemValue>() {

**public** **int** compare(ItemValue o1, ItemValue o2)

{

**return** o2.cost.compareTo(o1.cost);

}

});

**double** totalValue = 0d;

**for**(ItemValue i : iVal)

{

**int** curWt= (**int**) i.wt;

**int** curVal = (**int**) i.val;

**if**(capacity - curWt >= 0)

{

capacity = capacity - curWt;

totalValue += curVal;

}

**else**

{

**double** fraction = ((**double**)capacity/(**double**)curWt);

totalValue += (curVal\*fraction);

capacity = (**int**)(capacity - (curWt\*fraction));

**break**;

}

}

**return** totalValue;

}

}

**class** ItemValue

{

Double cost;

**double** wt,val,ind;

**public** ItemValue(**int** wt, **int** val,**int** ind)

{

**this**.wt = wt;

**this**.val = val;

**this**.ind = ind;

cost = **new** Double(val/wt);

}

}