19BCE7346 VIT-AP UNIVERSITY **Taran**

CSE- 3004 LAB Assignment

Academic year: 2020-2021 **Semester:** WIN

Faculty Name: Dr. D Sumathi Mam **Date:** 2 /9/2021

Student name: M.Taran **Reg. no.:** 19BCE7346

GREEDY APPROACH

Greedy approach 1:

Chef has gone shopping with his 5-year old son. They have bought N items so far. The items are numbered from 1 to N, and the item i weighs Wi grams.
Chef's son insists on helping his father in carrying the items. He wants his dad to give him a few items. Chef does not want to burden his son. But he won't stop bothering him unless he is given a few items to carry. So the Chef decides to give him some items. Obviously, Chef wants to give the kid less weight to carry.

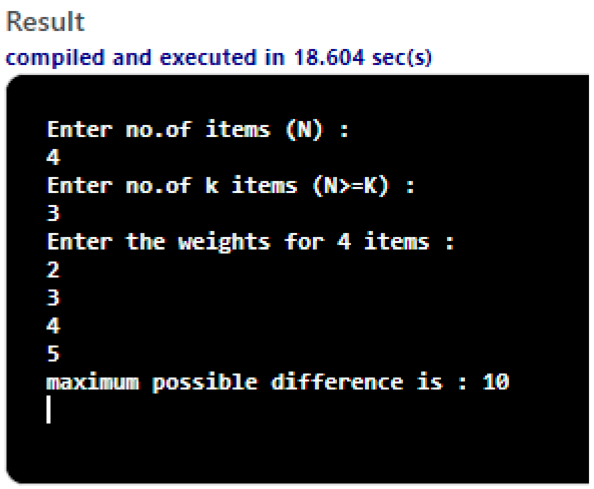
However, his son is a smart kid. To avoid being given the bare minimum weight to carry, he suggests that the items are split into two groups, and one group contains exactly K items. Then Chef will carry the heavier group, and his son will carry the other group.
Help the Chef in deciding which items should the son take. Your task will be simple. Tell the Chef the maximum possible difference between the weight carried by him and the weight carried by the kid.

CODE :

```
import java.util.*;
import java.lang.*;
import java.io.*;

public class Main
{
    public static void main (String[] args) throws java.lang.Exception
    {
        try {
            Scanner sc= new Scanner(System.in);
            System.out.println("Enter no.of items (N) :");
                int N=sc.nextInt();
            System.out.println("Enter no.of k items (N>=K) :");
                int K =sc.nextInt();
                int[] W = new int[N];
```

```java
            System.out.println("Enter the weights for "+W.length+" items :");
                for(int i =0 ; i < W.length ; i++){
                    W[i]=sc.nextInt();
                }
                if(K==N){
                    System.out.println(0);
                }
                Arrays.sort(W);

                int n=W.length-1;
                if(K<=(n/2)){
                int sum1=0;
                for(int i =0 ; i < K ;i++){
                    sum1+=W[i];
                }
                int sum2=0;
                for(int i = K ; i<W.length ; i++){
                    sum2+=W[i];
                }
                int res=sum2-sum1;
          System.out.println("maximum possible difference is :"+Math.abs(res));

                }
                else{
                    int sum1=0;
                    for(int i =W.length-1; i >=(N-K) ;i--){
                        sum1+=W[i];
                    }

                    int sum2=0;
                    for(int i =(N-K)-1;i>=0 ;i--){
                        sum2+=W[i];
                    }
                    int res=sum1-sum2;
                    System.out.println("maximum possible difference is : "+res);
                }

            sc.close();
        } catch (Exception e) {

        }
    }
}
```

OUTPUT :


Result
compiled and executed in 18.604 sec(s)

```
Enter no.of items (N) :
4
Enter no.of k items (N>=K) :
3
Enter the weights for 4 items :
2
3
4
5
maximum possible difference is : 10
```

Greedy approach 2 :

You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time.

Example:

Consider the following 6 activities.

start[] = {1, 3, 0, 5, 8, 5};
finish[] = {2, 4, 6, 7, 9, 9};

The maximum set of activities that can be executed by a single person is {0, 1, 3, 4}

The greedy choice is to always pick the next activity whose finish time is least among the remaining activities and the start time is more than or equal to the finish time of previously selected activity. We can sort the activities according to their finishing time so that we always consider the next activity as minimum finishing time activity.

Sort the activities according to their finishing time

Select the first activity from the sorted array and print it.

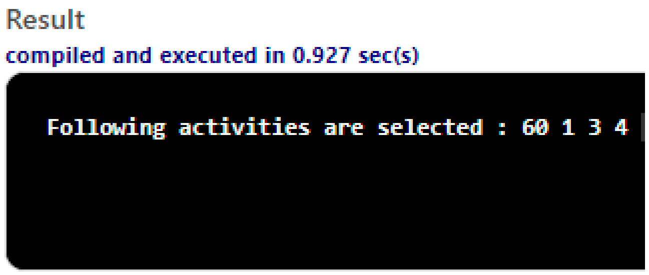
Do the following for remaining activities in the sorted array.

a) If the start time of this activity is greater than the finish time of previously selected activity then select this activity and print it.

CODE :

```
import java.util.*;
import java.lang.*;
import java.io.*;

public class ActivitySelection
{

      public static void printMaxActivities(int s[], int f[], int n)
      {
      int i, j;

      System.out.print("Following activities are selected : "+ n);

      i = 0;
      System.out.print(i+" ");

      for (j = 1; j < n; j++)
      {

            if (s[j] >= f[i])
            {
                  System.out.print(j+" ");
                  i = j;
            }
      }
      }

      public static void main(String[] args)
      {
      int s[] = {1, 3, 0, 5, 8, 5};
      int f[] = {2, 4, 6, 7, 9, 9};
      int n = s.length;

      printMaxActivities(s, f, n);
      }
}
```

OUTPUT :


Result

compiled and executed in 0.927 sec(s)

```
Following activities are selected : 60 1 3 4
```