21AD302 – Analysis of Algorithms

Question Set

(Common for CSE / IT / AI&DS)

Recursion

1. A naughty kid climbing stairs with N steps. From any step, he can take 1 step or 2 steps at a time. Find the total distinct ways of reaching.

Input format

Input consists of the number of steps in the staircase.

Output format

Print the number of distinct ways.

```
import java.util.*;
class Staircase
{
    static int ways(int n)
    {
        if(n<=1)
        {
            return n;
        }
        else
        {
            return ways(n-1)+ways(n-2);
        }
    }
    public static void main(String args[])
    {
        Scanner m=new Scanner(System.in);
        int n=m.nextInt();
        System.out.print(ways(n+1));
    }
}
```

2. The problem is to count all the possible paths from the top left to the bottom right of an MxN matrix with the constraints that from each cell you can either move only to the right or down.

Input format

Input consists of the number of rows (M) & columns (N) of the matrix.

Output format

Print the number of distinct ways.

```java
import java.util.*;
class Path
{
    static int ways(int r,int c)
    {
        if(r==1 || c==1)
        {
            return 1;
        }
        else
        {
            return ways(r-1,c)+ways(r,c-1);
        }
    }
    public static void main(String args[])
    {
        Scanner m=new Scanner(System.in);
        int r=m.nextInt();
        int c=m.nextInt();
        System.out.print(ways(r,c));
    }

}
```

3. Given an integer array of coins[ ] of size N representing different types of currency and an integer sum(M). The task is to find the number of ways to make a sum by using different combinations from coins[].

Input format

Input consists of the number of coins (N).

N number of array elements denoting the coin denominations.

The integer sum (M) to be produced with those coins.

Output format

Print the number of distinct ways.

```java
class Main {

    static int count(int coins[], int n, int sum)
    {

        if (sum == 0)
            return 1 ;

        if (sum < 0)
            return 0 ;

        if (n <= 0)
            return 0 ;

        return count(coins, n - 1, sum)
            + count(coins, n, sum - coins[n - 1]) ;
    }

    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in) ;
        int n = sc.nextInt() ;
        int coins[] = new int[n] ;
```

4. write a program to find whether the string is a palindrome string or not using recursion.

Input format

Input consists of a string.

Output format

Output consists of true or false

```java
import java.util.*;
class Main
{
    public static void main(String args[])
    {
        Scanner m=new Scanner(System.in);
        String s=m.nextLine();
        String rev=reversed(s);
        System.out.println("Reversed String:"+rev);
        if(s.equals(rev))
          System.out.print("True");
        else
          System.out.print("False");
    }
    static public String reversed(String str)
    {
        if(str.isEmpty())
           return str;
        return reversed(str.substring(1))+str.charAt(0);
    }
}
```

```java
import java.util.*;
class Main
{
public static void main(String args[])
{
        String str="";
        Scanner s=new Scanner(System.in);
        str=s.nextLine();
        boolean flag= palindromeCheck(str);
        System.out.println(flag);
}
public static boolean palindromeCheck(String s)
{
    if(s.length() == 0 || s.length() == 1)
        return true;
    if(s.charAt(0) == s.charAt(s.length()-1))
    {
        return palindromeCheck(s.substring(1, s.length()-1));
    }
    return false;
}
}
```

Math Algorithm

5. Sieve of Eratosthenes Concept

Given a number n, print all primes smaller than or equal to n.

Input format

Input integer is the n value.

Output format

List of prime numbers <= n.

```java
static void sieveOfEratosthenes(int n)
{
    boolean prime[]=new boolean[n+1];
    for(int i=0;i<=n;i++)
    {
        prime[i]=true;
    }
    for(int p=2;p<=n/2;p++)
    {
        if(prime[p]==true)
        {
            for(int i=p*p;i<=n;i+=p)
            {
                prime[i]=false;
            }
        }
    }
    for(int i=2;i<n;i++)
    {
        if(prime[i]==true)
          System.out.print(i+" ");
    }
}
```

```java
public static void main(String args[])
{
    Scanner m=new Scanner(System.in);
    int n=m.nextInt();
    sieveOfEratosthenes(n);

}
```

6. Sieve of Sundaram Concept

Given a number n, print all primes smaller than or equal to n.

Input format

Input integer is the n value.

Output format

List of prime numbers <= n.

```java
import java.util.*;
class SieveOfSundaram
{
    public static int Prime(int n)
    {
        int New=(n-1)/2;
        boolean arr[]=new boolean[New+1];

        for(int i=1;i<=New;i++)
        {
            for(int j=i;(i+j+2*i*j)<=New;j++)
            {
                arr[i+j+2*i*j]=true;
            }
        }

        if(n>2)
        {
            System.out.print(2+" ");
        }

        for(int i=1;i<=New;i++)
        {
            if(arr[i]==false)
            {
                System.out.print(2*i+1+" ");
            }
        }
        return -1;
    }
    public static void main(String args[])
    {
        Scanner m=new Scanner(System.in);
        int n=m.nextInt();
        Prime(n);
    }
```

7. Toggle the Bulbs

There are n bulbs that are initially off. You first turn on all the bulbs, then you turn off every second bulb. On the third round, you toggle every third bulb (turning on if it's off or turning off if it's on). For the ith round, you toggle every i bulb. For the nth round, you only toggle the last bulb. Return the number of bulbs that are on after n rounds.

Input format

Input integer is the n value.

Output format

Number of bulbs that are on.

```java
import java.util.*;
class ToggleTheBulbs
{
    public static void main(String args[])
    {
        Scanner m=new Scanner(System.in);
        int n=m.nextInt();
        int ans=(int)Math.sqrt(n);
        System.out.print(ans);
    }
}
```

8. Euclidean Algorithm

Write a program to compute the GCD of 2 numbers using recursion (Euclidean Algorithm).

Input format

Input consists of a non - negative integer.

Output format

Print the sum of digits of a given number.

```java
import java.util.*;
class GCD
{
    static int gcd(int n1,int n2)
    {
        if(n2!=0)
        {
            return gcd(n2,n1%n2);
        }
        else
        {
            return n1;
        }
    }
    public static void main(String args[])
    {
        Scanner m=new Scanner(System.in);
        int n1=m.nextInt();
        int n2=m.nextInt();
        System.out.print("G.C.D of "+n1+" and "+n2+" is "+gcd(n1,n2));
    }
}
```

9. Emirp number

Check whether the given number(N) is Emirp or not.

Note: Primes that become a different prime when their decimal digits are reversed. The name "emirp" is obtained by reversing the word "prime".

Input format

Input integer is the N value.

Output format

Emirp or Not Emirp message.

```java
import java.util.*;
class Main
{
    public static void main(String args[])
    {
        Scanner m=new Scanner(System.in);
        int n=m.nextInt();
        int rev=0;
        while(n!=0)
        {
            int a=n%10;
            rev=(rev*10)+a;
            n=n/10;
        }
       // System.out.print(rev);
       int s=0;
       for(int i=2;i<rev;i++)
       {
           if(rev%i!=0)
           {
               s=1;
           }
       else
       {
           s=0;
           break;
       }
   }
   if(s==1)
   {
       System.out.print("Emirp");
   }
   else
   {
       System.out.print("Not Emirp");
   }
}
```

10. Euler's Totient

Find the Euler's Totient function of the given input n.

```java
static int phi(int n)
{
    int result = n;
    for (int p = 2; p * p <= n; ++p)
    {
        if (n % p == 0)
        {
            while (n % p == 0)
                n /= p;
            result -= result / p;
        }
    }
    if (n > 1)
        result -= result / n;
    return result;
}
public static void main (String[] args)
{
    Scanner sc = new Scanner(System.in) ;
    int n = sc.nextInt() ;
    System.out.print( phi(n) ) ;
}
```

11. Prime numbers Sum

Given three numbers sum S, prime P, and N, find all N prime numbers after prime P such that their sum is equal to S.

Input format

Input integers are the values of N, P and S respectively.

Output format

List of such prime numbers in the new lines.

12. Segmented Sieve of Eratosthenes

Given a number n, print all primes smaller than or equal to n.

Input format

Input integer is the n value.

Output format

List of prime numbers <= n.

Searching And Sorting

13. Linear Search

Find the value in the given array. If found print "Yes" else "No".

Input format

First integer is the array size(N).

Next N integers are the array elements.

Next integer is the search element.

Output format

Print "Yes" or "No".

```
int n=m.nextInt();
int a[]=new int[n];
for(int i=0;i<n;i++)
{
    a[i]=m.nextInt();
}
int s=m.nextInt();
int c=0;
for(int i=0;i<n;i++)
{
    if(s==a[i])
    {
        c=1;
        break;
    }
    else
    {
        c=0;
    }
}
if(c==1)
   System.out.print("Yes");
else
   System.out.print("No");
```

14. Bubble Sort

Nandini asked her students to arrange a set of numbers in ascending order. She taught bubble sorting in the classroom which compares the adjacent elements and sorts them. She asked the students to arrange them using the BUBBLE SORT algorithm.

Example

Input

5

1 2 4 5 3

Output

1 2 3 4 5

Explanation

1 2 4 5 3

1 2 4 3 5

1 2 3 4 5

```
int n=m.nextInt();
int a[]=new int[n];
int temp=0;
for(int i=0;i<n;i++)
{
    a[i]=m.nextInt();
}
for(int i=0;i<n;i++)
{
    for(int j=0;j<n-1;j++)
    {
        if(a[j]>a[j+1])
        {
            temp=a[j];
            a[j]=a[j+1];
            a[j+1]=temp;
        }
    }
}
for(int i=0;i<n;i++)
{
    System.out.print(a[i]+" ");
}
```

15. Binary Search

Given a sorted array, the problem is to find the occurrence of an element X in the given array. Start the array index from 0. If element X is not present in the array, print "NO OCCURRENCES".

Input format

The input consists of a number of array elements, followed by array elements and the number X.

Output format

The output consists of an index of occurrence of an element x or not.

```
public static void binary(int a[],int x,int n)
{
    int f=0;
    int l=n-1;
    int mid=(f+l)/2;
    int s=0;
    while(f<=l)
    {
        if(a[mid]<x)
        {
            f=mid+1;
        }
        else if(a[mid]>x)
        {
            l=mid-1;
        }
        else if(a[mid]==x)
        {
            s=1;
            System.out.print(mid);
            break;
        }
        mid=(f+l)/2;
    }
```

```
    if(s!=1)
    {
        System.out.print("NO OCCURRENCES");
    }


}
public static void main(String args[])
{
    Scanner m=new Scanner(System.in);
    int n=m.nextInt();
    int a[]=new int[n];
    for(int i=0;i<n;i++)
    {
        a[i]=m.nextInt();
    }

    int x=m.nextInt();
    binary(a,x,n);
}
```

16. Merge Sort

Write a program to implement merge sort algorithm.

Example

Input

5

2 4 1 10 23

Output

1 2 4 10 23

Explanation

Sort array in an ascending order using divide and conquer method.

Input format

First integer represents size of input values.

Output format

Numbers in sorted order

```
public static void mergeSort(int a[],int n)
{
    if(n<2)
    {
       return;
    }
    int mid=n/2;
    int l[]=new int[mid];
    int r[]=new int[n-mid];

    for(int i=0;i<mid;i++)
    {
        l[i]=a[i];
    }
    for(int i=mid;i<n;i++)
    {
        r[i-mid]=a[i];
    }
    mergeSort(l,mid);
    mergeSort(r,n-mid);

    merge(a,l,r,mid,n-mid);

}
```

```java
public static void merge(int a[],int l[],int r[],int left,int right)
{
    int i=0,j=0,k=0;
    while(i<left && j<right)
    {
        if(l[i]<=r[j])
        {
            a[k++]=l[i++];
        }
        else
        {
            a[k++]=r[j++];
        }
    }
    while(i<left)
    {
            a[k++]=l[i++];
    }
    while(j<right)
    {
            a[k++]=r[j++];
    }
```

```java
public static void main(String args[])
{
    Scanner m=new Scanner(System.in);
    int num=m.nextInt();
    int arr[]=new int[num];
    for(int i=0;i<num;i++)
    {
        arr[i]=m.nextInt();
    }
    mergeSort(arr,num);
  //  System.out.println();
    for(int i=0;i<num;i++)
    {
        System.out.print(arr[i]+" ");
    }
}
```

17. Selection Sort

Its the first day for the students at school and the students enter the class and get seated at random places without any height order. So the students who are short and sitting back are not able to see the board since they sit behind taller students.

Understanding this difficulty, the teacher decides to make the students sit in height order.

Suppose there are n students in the class. She makes all the students to stand in a line and compares the first student's height with the remaining (n-1) students. If the first student's height is greater than the ith student, then the taller person goes to the ith place and ith student comes to the first place. Again the new first student's height is compared with remaining students and if his height is greater than ith student the first student goes to ith place and ith place student comes to first place and this goes on till the end.

This process continues for all the students. Finally the students are in height order.

Write a program to perform selection sort on an array of n elements.

Input format

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array.

The next n integers correspond to the elements in the array.

```
public static void main(String args[])
{
    Scanner m=new Scanner(System.in);
    int n=m.nextInt();
    int a[]=new int[n];
    int temp=0;
    for(int i=0;i<n;i++){
        a[i]=m.nextInt();
    }
    System.out.println("Student's height order before sorting:");
    for(int i=0;i<n;i++){
        System.out.print(a[i]+" ");
    }
    System.out.println();
    for(int i=0;i<n;i++)
    {
        //int min=i;
        for(int j=i+1;j<n;j++)
        {
            if(a[i]>a[j])
            {
               temp=a[i];
               a[i]=a[j];
               a[j]=temp;
            }
```

```
    }
    if((i+1)==n)
    {
      System.out.println("After final comparison of all students , the height order becomes:");
      for(int k=0;k<n;k++)
      {
        System.out.print(a[k]+" ");
      }
    }
    else
    {
      System.out.println("Height order of students after iteration "+(i+1)+" ");
      for(int k=0;k<n;k++)
      {
        System.out.print(a[k]+" ");
      }
    }
    System.out.println();
}
```

SELECTION SORT

```
for(int i=0;i<n;i++)
{
    int min=i;
    for(int j=i+1;j<n;j++)
    {
        if(a[min]<a[j])
        {
            min=j;
        }
    }
        int temp=a[min];
        a[min]=a[i];
        a[i]=temp;
```

18. String Sorting

Jack is working on sorting strings in ascending order for his school project. Help him with the process by writing a bubble sort code to achieve the result.

Example

Input

4

apple

orange

banana

pine

Output

apple banana orange pine

```
public static void sortStrings(String[] arr, int n)
{
        String temp;
        for (int j = 0; j < n - 1; j++)
       {
          for (int i = j + 1; i < n; i++)
          {
              if (arr[j].compareTo(arr[i]) > 0)
              {
                 temp = arr[j];
                 arr[j] = arr[i];
                 arr[i] = temp;
              }
          }
        }
}
```

```java
public static void main(String[] args)
{
     Scanner sc=new Scanner(System.in);
     int n=sc.nextInt();
     String[] arr =new String[n];
     for(int i=0;i<n;i++)
     {
        arr[i]=sc.next();
     }
    sortStrings(arr, n);
    for (int i = 0; i < n; i++)
       System.out.print(arr[i]+" ");
}
}
```

19. Mr.Davit has N*N square boxes with some numbers. Write the program to arrange the numbers in the square box in ascending order.

Example

Input

3 9 8 7 6 5 4 3 2 1

Output

1 2 3

4 5 6

7 8 9

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

class Matrix {

    //static int SIZE = 10;
    static void sortMat(int mat[][], int n)
    {
        int temp[] = new int[n * n];
        int k = 0;

        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                temp[k++] = mat[i][j];

        Arrays.sort(temp);

        k = 0;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                mat[i][j] = temp[k++];
    }
```

```
    static void printMat(int mat[][], int n)
    {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++)
                System.out.print( mat[i][j] + " ");
            System.out.println();
        }
    }
    public static void main(String args[])
    {
        Scanner my=new Scanner(System.in);
        int n=my.nextInt();
        int mat[][] = new int[n][n];
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                mat[i][j]=my.nextInt();
            }
        }
        sortMat(mat, n);
        printMat(mat, n);
    }
}
```

20. Sorting the Students' data

Write a program to arrange the student record according to their average marks.

Example

Input

2

1 Ram 57 68 90

2 Sita 90 80 100

Output

2 Sita 270 90.00

1 Ram 215 71.67

```java
// You are using Java
import java.util.*;
class data{
    int roll;
    String name;
    int m1,m2,m3;
    float avg;
    int sum;
    data(int num,String name,int m1,int m2,int m3,float avg,int sum){
        this.roll=num;
        this.name=name;
        this.m1=m1;
        this.m2=m2;
        this.m3=m3;
        this.avg=avg;
        this.sum=sum;
    }
}
```

```java
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        data[] d=new data[n+1];
        for(int i=0;i<n;i++){
            int num=sc.nextInt();
            String name=sc.next();
            int m1=sc.nextInt();
            int m2=sc.nextInt();
            int m3=sc.nextInt();
            float avg=(float)(m1+m2+m3)/3;
            int sum=m1+m2+m3;
            d[i]=new data(num,name,m1,m2,m3,avg,sum);
            }
        for(int i=0;i<n;i++){
            for(int j=i+1;j<n;j++){
                if(d[i].avg<d[j].avg){
                    data t=d[i];
                    d[i]=d[j];
                    d[j]=t;
                }
            }
        }
        for(int i=0;i<n;i++){
            System.out.println(d[i].roll+" "+d[i].name+" "+d[i].sum+" "+d[i].avg);
```

21. Insertion Sort

Sort the given numbers using Insertion Sort algorithm.

Example

Input

5

4

2

5

1

3

Output

1 2 3 4 5

```
class InsertionSort {
    static void sort(int arr[])
    {
        int n = arr.length;
        for (int i = 1; i < n;i++) {
            int key = arr[i];
            int j = i - 1;
            while (j >= 0 && arr[j] > key) {
                arr[j + 1] = arr[j];
                j = j - 1;
            }
            arr[j + 1] = key;
        }
    }
    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + " ");

        System.out.println();
    }
```

```
public static void main(String args[])
{
    Scanner scan=new Scanner(System.in);
    int n=scan.nextInt();
    int arr[]=new int[n];
    for(int i=0;i<n;i++){
        arr[i]=scan.nextInt();
    }

    sort(arr);

    printArray(arr);
}
```

22. Heap Sort

Sort the given numbers using the Heap Sort algorithm.

Example

Input

5

4

2

5

1

3

Output

1 2 3 4 5

STRING ALGORITHM

23. Naive Algorithm

Find the occurrence of Substring in the main String.

Example

Input

captain

cap

Output

Found at index 0

```
        Scanner m=new Scanner(System.in);
        String str=m.next();
        String p=m.next();
        naiveSearch(str,p);
    }
    public static void naiveSearch(String str,String p)
    {
        int n=str.length();
        int m=p.length();
        for(int i=0;i<=n-m;i++)
        {
            int j;
           for( j=0;j<m;j++)
           {
               if(str.charAt(i+j)!=p.charAt(j))
                  break;
           }
           if(j==m)
           {
               System.out.println("Found at "+i);
               System.exit(0);
           }
        }
           System.out.println("Not Found");
```

24. KMP Algorithm

Find the occurrence of Substring in the main String using KMP Algorithm.

Input format

First string is the main String

Second string is the sub-string.

25. Rail fence Algorithm

Write a program for the String encryption and Decryption using Rail Fence Algorithm.

Input format

First input is the string

Second integer input is the key

Output format

Encrypted message (String)

Decrypted message (String)

```
class RailFence
{
    public static void main(String args[])
    {
        Scanner m=new Scanner(System.in);
        String s=m.nextLine();
        int d=m.nextInt();
        System.out.println("Encrypted text is: "+Encryption(s,d));
        System.out.println("Decrypted text is: "+s);
    }
    static String Encryption(String s,int d)
    {
        if(d<2)
          return s;
        int r=d;
        int c=s.length();
        char mat[][]=new char[r][c];
        for(int i=0;i<r;i++)
        {
          for(int j=0;j<c;j++)
          {
              mat[i][j]='*';
          }
        }
```

```
    boolean d_down=false;
    int row=0,col=0;
    for(int i=0;i<c;i++)
    {
        if(row==0 || row==r-1)
        d_down=!d_down;
        mat[row][col++]=s.charAt(i);
        if(d_down)
          row++;
        else
          row--;
    }
    char result[]=new char[c];
    int k=0;
    for(int i=0;i<r;i++)
    {
      for(int j=0;j<c;j++)
      {
          if(mat[i][j]!='*')
            result[k++]=mat[i][j];
      }
    }
    String eString=new String (result);
    return eString;
```

GREEDY APPROACH

26. Coin Calculator

The King of Holumba Island has devised a coin system for exchange of goods and trade among the people. There are 4 coins provided by the king. The coins are of the

value Rs. 1, Rs. 2, Rs. 3, and Rs. 5. Devise a coin calculator which finds out the number of coins required to compute an amount of money, such that a least amount of coins are needed. When there is more than one solution, the output should be such that the coins with more value are given preference.

Example

Input

12

Output

2

0

1

0

```java
import java.util.*;
class Main
{
    public static void main(String args[])
    {
        Scanner m=new Scanner(System.in);
        int n=m.nextInt();
        int a[]={5,3,2,1};
        for(int i=0;i<4;i++)
        {
           System.out.println(n/a[i]);
           n%=a[i];

        }


    }
}
```

BACKTRACKING

27. Rat in a Maze:

A Maze is given as N*N binary matrix of blocks where source block is the upper left most block i.e., maze[0][0], and destination block is lower rightmost block i.e., maze[N-1][N-1]. A rat starts from the source and has to reach its destination. The rat can move only in two directions: forward and down.

In the maze matrix, 0 means the block is a dead end, and 1 means the block can be used in the path from source to destination.

Write a program for the same.

```
public static boolean isSafe(int a[][],int i,int j)
{
    return(i>=0 && i<a.length && j>=0 && j<a[0].length && a[i][j]==1);
}
public static boolean solveMaze(int a[][],int i,int j,int sol[][])
{
   if(i==a.length-1 && j==a[0].length-1 && a[i][j]==1)
   {
       sol[i][j]=1;
       return true;
   }

   if(isSafe(a,i,j))
   {
       sol[i][j]=1;
       if(solveMaze(a,i+1,j,sol))
         return true;

       if(solveMaze(a,i,j+1,sol))
         return true;
       sol[i][j]=0;
       return false;
   }
   return false;
```

```
public static void main(String args[])
{
    Scanner m=new Scanner(System.in);
    int n=m.nextInt();
    int a[][]=new int[n][n];
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            a[i][j]=m.nextInt();
        }
    }
    int sol[][]=new int[n][n];
    if(solveMaze(a,0,0,sol))
    {
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                System.out.print(sol[i][j]+" ");
            }
            System.out.println();
        }
    }
```

```
    }
    else
    {
        System.out.print("Solution doesn't exist");
    }
}
```

DYNAMIC PROGRAMING

28. Longest Common Substring(LCS)

Write a program to find the length of LCS between the given two strings using Dynamic Programming (DP).

```
import java.util.*;
class main{
    public static void main(String[] args){
        Scanner ob=new Scanner(System.in);

        String str1=ob.nextLine();
        String str2=ob.nextLine();
        char[] arr1=str1.toCharArray();
        char[] arr2=str2.toCharArray();
        int m=arr1.length;
        int n=arr2.length;
        int[][] arr=new int[m+1][n+1];
        for(int i=1;i<=m;i++){
            for(int j=1;j<=n;j++){
                if(arr1[i-1]==arr2[j-1]){
                    arr[i][j]=arr[i-1][j-1]+1;
                }
                else{
                    arr[i][j]=0;
                }
            }
        }
```

```
    int max=0;
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++){
            if(arr[i][j]>max){
                max=arr[i][j];
            }
        }
    }
    System.out.print(max);
}
```

TREE

29. Implementation of Segment Tree Write a Program to implement the Segment Tree and find the sum for the given range before and after updation.

Input format

First line of the input is the number of array elements Second line of the input is array of elements Third line of the input is query starting range(start) Fourth line of the input is query ending range(end) Fifth line of the input is index of the array element to be updated Six line of the input is the element to be updated.

Output format

First line of the output is sum of elements in the given range Second line of the output is sum of elements in the given range

GRAPH

30. Floyd Warshall Algorithm

The problem is to find the shortest distances between every pair of vertices in a given edge-weighted un-directed Graph using the Floyd Warshall Algorithm.

Input format

Number of vertices Number of edges Edge values with source & destination

Output format

Print the matrix with the shortest path between every node.

```
import java.util.*;
class Main {
    final static int INF = 999 ; static int V ;
    static void floydWarshall(int graph[][]){
        int dist[][] = new int[V][V];
        int i, j, k;
        for (i = 0; i < V; i++)
        for (j = 0; j < V; j++)
        dist[i][j] = graph[i][j];
            for (k = 0; k < V; k++) {
                for (i = 0; i < V; i++) {
                    for (j = 0; j < V; j++) {
                    if (dist[i][k] + dist[k][j]
                    < dist[i][j])
                    dist[i][j]
                    = dist[i][k] + dist[k][j];
                    }
                }
            }
            printSolution(dist);
        }
```

```
    static void printSolution(int dist[][]){
            for (int i = 0; i < V; ++i) {
                for (int j = 0; j < V; ++j) {
                    if (dist[i][j] == INF)
                    System.out.print("INF ") ;
                    else

                     System.out.print(dist[i][j] + " ") ;
                }
           System.out.println();
           }
    }
```

```
public static void main(String[] args){
    Scanner sc = new Scanner(System.in) ;
    V = sc.nextInt() ;
    int edges = sc.nextInt() ;
    int graph[][] = new int [V][V] ;
    for(int i = 0 ; i < V ; i++){
        for(int j = 0 ; j < V ; j++){
            if(i==j)
            graph[i][j] = 0 ;
            else
            graph[i][j] = INF ;
        }
    }
    int start, end, value ;
    for(int i = 0 ; i < edges ; i++){
        start = sc.nextInt() ;
        end = sc.nextInt() ;
        value = sc.nextInt() ;
        graph[start][end] = value ;
        graph[end][start] = value ;
    }
    System.out.println("Original matrix") ;
```

```
  for(int i = 0 ; i < V ; i++){
      for(int j = 0 ; j < V ; j++){
          if(graph[i][j] == INF)
          System.out.print("INF ") ;
          else
          System.out.print(graph[i][j]+" ") ;
      }
      System.out.println() ;
  }
  System.out.println() ;
  System.out.println("Shortest path matrix") ;
  floydWarshall(graph);
}
```