DAY 10:

SESSION 1:

Problem 1:

Largest Possible Odd Integer

ID:11123 Solved By 734 Users

The program must accept an integer **N** as the input. The program must print the largest possible odd integer using all the digits in N as the output. If it is not possible to form such an integer, the program must print no as the output.

Boundary Condition(s):
10 <= N <= 10^17

Input Format:
The first line contains N.

Output Format:
The first line contains the largest possible odd integer using all the digits in N or no.

Example Input/Output 1:
Input:
120087460153

Output:
876543210001

Explanation:
The largest possible odd integer using all the digits in 120087460153 is **876543210001**.

Example Input/Output 2:
Input:
246228

Output:
no

Max Execution Time Limit: 500 millisecs

Code:

```c
#include<stdio.h>
#include<stdlib.h>
#define ULL unsigned long long int
int main()
{
    ULL N;
    scanf("%llu", &N);
    int digits[10] = {0};
    while(N !=0 ){
        digits[N % 10]++; //increments the unit value in the digits array
        N /= 10;
    }
```

```
    int unitDigit = -1;
    for(int dig = 1; dig < 10; dig += 2){ //loop to check for odd number for the
unit digit
     // checking from the beginning of the digits array as to print largest
possible odd number so the unit digit can or should he the smallest value
        if(digits[dig] > 0){
            unitDigit = dig;
            digits[unitDigit]--;
            break;
        }
    }
    if(unitDigit == -1){
        printf("no");
        return;
    }
    int start = 1;
    for(int digit = 1; digit <= 9; digit++){ //this loop to avoid leading zeros
//starting from 1 to avoid zero as the first value
        if(digits[digit] > 0){
            start = 0;  //after printing the first value we can change the start
to 0
            break;
        }
    }
    for(int digit = 9; digit >= start; digit--){ //loop to print the output
        while(digits[digit]-- > 0){  //using post decrement
            printf("%d", digit);     //can also be coded as digits[digit]--
        }
    }
    printf("%d", unitDigit);
}
```

Problem 2:

Smallest Possible Odd Integer

ID:11124 Solved By 704 Users

The program must accept an integer **N** as the input. The program must print the smallest possible odd integer using all the digits in N as the output. If it is not possible to form such an integer, the program must print no as the output.

Boundary Condition(s):
10 <= N <= 10^17

Input Format:
The first line contains N.

Output Format:
The first line contains the smallest possible odd integer using all the digits in N or no.

Example Input/Output 1:
Input:
1670078423

Output:
1002346787

Explanation:
The smallest possible odd integer using all the digits in 1670078423 is **1002346787**.

Example Input/Output 2:
Input:
40068

Output:
no

Max Execution Time Limit: 500 millisecs

Code:

```
#include<stdio.h>
#include<stdlib.h>
#define ULL unsigned long long int
int main()
{
    ULL N=1670078423;
    // scanf("%llu",&N);
    int digits[10]={0};
    while(N!=0){
        digits[N%10]++; //increments the unit value in the digits array
        N/=10;
    }
    int unitDigit=-1;
    for(int dig=9;dig>=1;dig-=2){
        if(digits[dig]>0){ //loop to check for odd number for the unit digit
```

```
                        // checking from the last of the digits array as to
print smallest possible odd number so the unit digit can or should he the highest
value
            unitDigit=dig;
            digits[dig]--;
            break;
        }
    }
    if(unitDigit==-1){
        printf("no");
        return;
    }

    int start=1;
    for(int digit=1;digit<=9;digit++){ //this loop to avoid leading
zeros   //starting from 1 to avoid zero as the first value
        if(digits[digit]>0){
            printf("%d",digit);
            digits[digit]--;
            start=0;    //after printing the first value we can change the start
to 0
            break;
        }
    }
    for(int digit=start ;digit<=9;digit++){ //loop to print the output
        while(digits[digit]-->0){ //using post decrement //can also be coded as
digits[digit]--
            printf("%d",digit);
        }
    }
    printf("%d",unitDigit);
}
```

Session 2:

Problem1:

Matrix Zig-Zag from Top Left

ID:11125 Solved By 699 Users

The program must accept an integer matrix of size **R*C** as the input. The program must print the elements from the top left of the matrix in diagonally zig-zag order.

Boundary Condition(s):
2 <= R, C <= 100

Input Format:
The first line contains R and C separated by a space.
The next R lines, each containing C integers separated by a space.

Output Format:
The first line contains all R*C elements in diagonally zig-zag order, with the elements separated by a space.

Example Input/Output 1:
Input:
3 7
1 9 3 9 4 7 2
4 5 2 4 8 3 5
9 3 6 3 5 1 8

Output:
1 9 4 9 5 3 9 2 3 6 4 4 7 8 3 5 3 2 5 1 8

Example Input/Output 2:
Input:
4 4
29 76 80 57
18 69 47 36
36 26 68 61
48 34 30 82

Output:
29 76 18 36 69 80 57 47 26 48 34 68 36 61 30 82

Max Execution Time Limit: 500 millisecs

Stimulation:

https://pythontutor.com/visualize.html#code=import%20java.util.*%3B%0Apublic%20class%20matrixZigZagFromTopLeft%20%7B%0A%0A%20%20%20%20public%20static%20void%20main%28String%5B%5D%20args%29%20%7B%0A%20%20%20%20%20%20%20%20//Your%20Code%20Here%0A%20%20%20%20%20%20%20%20Scanner%20in%20%3D%20new%20Scanner%28System.in%29%3B%0A%20%20%20%20%20%20%20%20int%20R%3D3%3B%0A%20%20%20%20%20%20%20%20int%20C%3D3%3B%0A%20%20%20%20%20%20%20%20int%20matrix%5B%5D%5B%5D%20%3D%7B%7B1,2,3%7D,%7B4,5,6%7D,%7B7,8,9%7D%7D%3B%0A%20%20%20%20%20%20%20%20//%20for%28int%20row%3D0%3Brow%3CR%3Brow%2B%2B%29%7B%0A%20%20%20%20%20%20%20%20//%20%20%20%20%20%20for%28int%20col%3D0%3Bcol%3CC%3Bcol%2B%2B%29%7B%0A%20%20%20%20%20%20%20%20//%20%20%20%20%20%20%20%20matrix%5Brow%5D%5Bcol%5D%3Din.nextInt%28%29%3B%0A%20%20%20%20%20%20%20%20//%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20//%20%7D%0A%0A%0A%20%20%20%20%20%20%20%20int%20row%3D0,col%3D0,direction%3D1%3B%0A%20%20%20%20%20%20%20%20for%28int%20iter%3D1%3Biter%3C%3DR%2BC-

1%3Biter%2B%2B%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20if%28direction%3D%3
D1%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20while%28row%3E%3
D0%20%26%26%20col%3CC%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2
0%20%20%20%20%20System.out.print%28matrix%5Brow%5D%5Bcol%5D%2B%22%20%22%29%3B%0A
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20row--
%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20col%2B%2
B%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20
%20%20%20%20%20%20%20%20%20%20%20%20direction%3D-
1%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20//Top%20overflow%20%2
0%20%20%20%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20if%28row%3C0%2
0%26%26%20col%3CC%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20
%20%20%20%20row%3D0%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20
%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20//Right%20overflow%0A%2
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20if%28col%3E%3DC%29%7B%0A%20%2
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20col%3DC-
1%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20row%2B
%3D2%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20
%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20else%
7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20while%28row%3CR%20%26%
26%20col%3E%3D0%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2
0%20%20%20System.out.print%28matrix%5Brow%5D%5Bcol%5D%2B%22%20%22%29%3B%0A%20%20
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20row%2B%2B%3B%0A%20%
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20col--
%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%
20%20%20%20%20%20%20%20%20%20%20%20direction%3D1%3B%0A%20%20%20%20%20%20%20%
20%20%20%20%20%20%20%20%20%20//Left%20overflow%0A%20%20%20%20%20%20%20%20%20%
20%20%20%20%20%20if%28col%3C0%20%26%26%20row%3CR%29%7B%0A%20%20%20%20%20%20%
20%20%20%20%20%20%20%20%20%20%20%20%20%20col%3D0%3B%0A%20%20%20%20%20%20%2
0%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20%20
%20%20%20//Bottom%20overflow%0A%20%20%20%20%20%20%20%20%20%20%20%20%20
%20if%28row%3E%3DR%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2
0%20%20%20%20row%3DR-
1%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20col%2B%
3D2%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%
20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%7D%0A%20%20%20%2
0%7D%0A%7D&cumulative=false&curInstr=101&heapPrimitives=nevernest&mode=display&origin=opt-
frontend.js&py=java&rawInputLstJSON=%5B%5D&textReferences=false

Code:

```
import java.util.*;
public class matrixZigZagFromTopLeft {

    public static void main(String[] args) {
        //Your Code Here
        Scanner in = new Scanner(System.in);
        int R=in.nextInt();
        int C=in.nextInt();
        int matrix[][] = new int[R][C];

        for(int row=0;row<R;row++){
            for(int col=0;col<C;col++){
                matrix[row][col]=in.nextInt();
            }
        }

        int row=0,col=0,direction=1;
        for(int iter=1;iter<=R+C-1;iter++){
            if(direction==1){
                while(row>=0 && col<C){
                    System.out.print(matrix[row][col]+" ");
                    row--;
                    col++;
                }
                direction=-1;
                //Top overflow
                if(row<0 && col<C){
                    row=0;
                }
                //Right overflow
                if(col>=C){
                    col=C-1;
                    row+=2;
                }
            }
            else{
                while(row<R && col>=0){
                    System.out.print(matrix[row][col]+" ");
                    row++;
                    col--;
                }
```

```
                direction=1;
                //Left overflow
                if(col<0 && row<R){
                    col=0;
                }
                //Bottom overflow
                if(row>=R){
                    row=R-1;
                    col+=2;
                }
            }
        }
    }
}
```

Problem 2:

Matrix Zig-Zag from Bottom Right

ID:11126 Solved By 662 Users

The program must accept an integer matrix of size **R*C** as the input. The program must print the elements from the bottom right of the matrix in diagonally zig-zag order.

Boundary Condition(s):
2 <= R, C <= 100

Input Format:
The first line contains R and C separated by a space.
The next R lines, each containing C integers separated by a space.

Output Format:
The first line contains all R*C elements in diagonally zig-zag order, with the elements separated by a space.

Example Input/Output 1:
Input:
3 7
44 23 14 62 34 24 29
18 66 22 77 14 51 60
13 67 35 26 34 40 72

Output:
72 60 40 34 51 29 24 14 26 35 77 34 62 22 67 13 66 14 23 18 44

Example Input/Output 2:
Input:
4 4
6 7 5 8
8 3 2 1
9 1 2 6
5 4 5 1

Output:
1 6 5 4 2 1 8 2 1 5 9 3 5 7 8 6

Max Execution Time Limit: 500 millisecs

Stimulation:

https://pythontutor.com/visualize.html#code=import%20java.util.*%3B%0Apublic%20class%20matrixZigZagFromTopLeft%20%7B%0A%0A%20%20%20%20public%20static%20void%20main%28String%5B%5D%20args%29%20%7B%0A%20%20%20%20%20%20%20%20//Your%20Code%20Here%0A%20%20%20%20%20%20%20%20Scanner%20in%20%3D%20new%20Scanner%28System.in%29%3B%0A%20%20%20%20%20%20%20%20int%20R%3D3%3B%0A%20%20%20%20%20%20%20%20int%20C%3D3%3B%0A%20%20%20%20%20%20%20%20int%20matrix%5B%5D%5B%5D%20%3D%7B%7B1,2,3%7D,%7B4,5,6%7D,%7B7,8,9%7D%7D%3B%0A%20%20%20%20%20%20%20%20//%20for%28int%20row%3D0%3Brow%3CR%3Brow%2B%2B%29%7B%0A%20%20%20%20%20%20%20%20//%20%20%20%20%20for%28int%20col%3D0%3Bcol%3CC%3Bcol%2B%2B%29%7B%0A%20%20%20%20%20%20%20%20//%20%20%20%20%20%20%20%20%20matrix%5Brow%5D%5Bcol%5D%3Din.nextInt%28%29%3B%0A%20%20%20%20%20%20%20%20//%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20//%20%7D%0A%0A%0A%20%20%20%20%20%20%20%20int%20row%3DR-1,col%3DC-1,direction%3D1%3B%0A%20%20%20%20%20%20%20%20for%28int%20iter%3D1%3Biter%3C%3DR%2BC-1%3Biter%2B%2B%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20if%28direction%3D%3D1%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20while%28row%3E%3D0%20%26%26%20col%3CC%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20System.out.print%28matrix%5Brow%5D%5Bcol%5D%2B%22%20%22%29%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20row--%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20col%2B%2B%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20direction%3D-1%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20//Top%20overflow%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20if%28row%3C0%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20row%3D0%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20col%3Dcol-2%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20//Right%20overflow%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20if%28col%3E%3DC%20%26%26%20row%3E%3D0%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20col%3DC-1%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20else%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20while%28row%3CR%20%26%26%20col%3E%3D0%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20System.out.print%28matrix%5Brow%5D%5Bcol%5D%2B%22%20%22%29%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20row%2B%2B%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20col--%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20direction%3D1%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20//Left%20overflow%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20if%28col%3C0%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20

20%20%20%20%20%20%20%20col%3D0%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20row%3Drow-2%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20//Bottom%20overflow%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20if%28row%3E%3DR%29%7B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20row%3DR-1%3B%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20%20%20%7D%0A%20%20%20%20%7D%0A%7D&cumulative=false&curInstr=92&heapPrimitives=nevernest&mode=display&origin=opt-frontend.js&py=java&rawInputLstJSON=%5B%5D&textReferences=false

Code:

```
import java.util.*;
public class matrixZigZagFromBottomRight {

    public static void main(String[] args) {
        //Your Code Here
        Scanner in = new Scanner(System.in);
        int R=in.nextInt();
        int C=in.nextInt();
        int matrix[][] = new int[R][C];
        for(int row=0;row<R;row++){
            for(int col=0;col<C;col++){
                matrix[row][col]=in.nextInt();
            }
        }
        int row=R-1,col=C-1,direction=1;
        for(int iter=1;iter<=R+C-1;iter++){
            if(direction==1){
                while(row>=0 && col<C){
                    System.out.print(matrix[row][col]+" ");
                    row--;
                    col++;
                }
                direction=-1;
                //Top overflow
                if(row<0){
                    row=0;
                    col=col-2;
                }
                //Right overflow
                if(col>=C && row>=0){
                    col=C-1;
```

```
                }
            }
            else{
                while(row<R && col>=0){
                    System.out.print(matrix[row][col]+" ");
                    row++;
                    col--;
                }
                direction=1;
                //Left overflow
                if(col<0){
                    col=0;
                    row=row-2;
                }
                //Bottom overflow
                if(row>=R){
                    row=R-1;
                }
            }
        }

    }
}
```

Session 3:

Problem 1:

Street Travel Count

ID:11127 Solved By 465 Users

Mr.X has a bike and is travelling in a town which has **N** horizontal (West to East direction) and N vertical (North to South direction) streets. At the meeting junctions of these horizontal and vertical streets there may be a block. If there is a block Mr.X can take diversion to any other street and travel to his destination. A value of 1 indicates that a junction (meeting point of two roads) is NOT blocked and a value of 0 indicates that a junction is blocked. The streets are numbered from 0 to N-1 (similar to array indices). The source (starting junction of Mr.X and the destination junctions details are passed as the input. The program must print the number of streets through which Mr.X must travel to travel from the source to destination.

Boundary Condition(s):
1 <= N <= 100

Input Format:
The first line contains N.
The next N lines each containing N values 1 or 0 separated by a space.
The $(N+2)^{nd}$ line contains the source junction co-ordinates separated by a space.
The $(N+3)^{rd}$ line contains the destination junction co-ordinates separated by a space.

Output Format:
The number of streets Mr.X must travel to travel from source to destination.

Example Input/Output 1:
Input:
3
1 0 1
1 0 1
1 1 1
0 0
0 2

Output:
3

Explanation:
The source is (0,0) indicated as S and the destination (0,2) by D.
S 0 **D**
1 0 1
1 1 1

0 implies block. Hence Mr.X must travel along 1s. Hence the path to travel is denoted by letter P from S to D.
S 0 **D**
P 0 **P**
P P P

Hence we can notice that Mr.X must travel through 3 streets to reach the destination.

Example Input/Output 2:
Input:
4
1 1 1 0
0 0 1 1
1 1 0 1
0 1 1 1
0 1
2 0

Output:
7

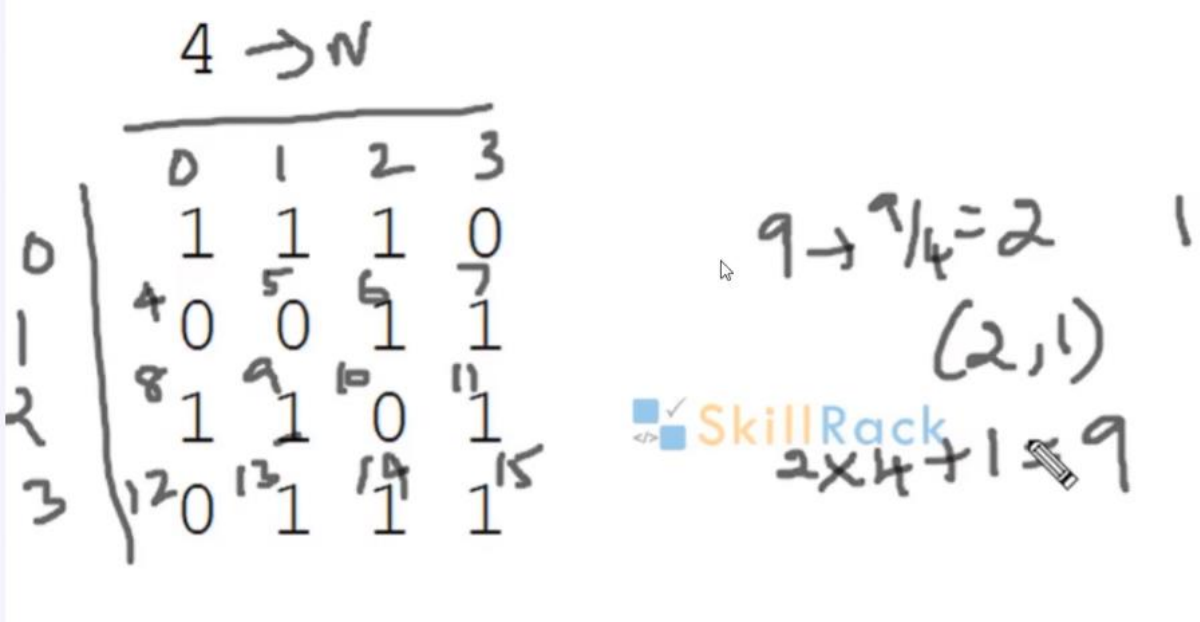
Explanation:
The path denoted by the letter P is
1 **S P** 0
0 0 **P P**
D P 0 **P**
0 **P P P**
Hence we can notice that Mr.X must travel through 7 streets to reach the destination.

Example Input/Output 3:
Input:
4
1 1 1 0
0 0 1 1
1 1 0 1
0 1 1 1
0 1
2 1

Output:
6

Max Execution Time Limit: 2000 millisecs

Logic for input for Source and Destination



Code:

```
import java.util.*;
public class streetTravelCount {

    public static void main(String[] args) {
        //Your Code Here
        Scanner in = new Scanner(System.in);
        int N=in.nextInt();
        int matrix[][] = new int[N][N];
        for(int row=0;row<N;row++){
            for(int col=0;col<N;col++){
                matrix[row][col]=in.nextInt();
            }
        }
        int source=in.nextInt()*N+in.nextInt();    //converting given input
values to numerical representation
        int destination= in.nextInt()*N+in.nextInt();
        boolean visited[] = new boolean[N*N];
        int streets[] = new int[N*N];  //street array to track the number of
streets travelled
        Queue<Integer> queue = new ArrayDeque<>();
        queue.add(source);
        visited[source]=true;
        streets[source]=0; //marking the source as zero as it is not considered
in street count since it is a street
        while(!queue.isEmpty()){
            int node = queue.poll();
            List<Integer> related = getRelated(matrix,node,N); //getRelated func
is to get all adjacent connected

            for(Integer relNode:related){ //traversing the related list
                if(!visited[relNode]){
                    queue.add(relNode);
                    visited[relNode]=true;
                    streets[relNode]=1+streets[node]; //incrementing the street
count
                    if(relNode==destination){ //if the node is equal to the
destination
                        System.out.println(streets[relNode]);  //printing the
last count of street reaching the index
                        return;
                    }
```

```
                }
            }
        }
        System.out.println(streets[destination]); //printing the count if the
value is zero then no path exists
    }
    private static List<Integer> getRelated(int [][] matrix,int node,int N){
        List<Integer> nodes = new ArrayList<>();
        int nodeRow=node/N,nodeCol=node%N; //calculating row and col index
//refer the docs


        for(int col=nodeCol-1;col>=0;col--){ // left traversal find all possible
street in left direction
            if(matrix[nodeRow][col]==1){
                nodes.add(nodeRow*N+col);
            }
            else{
                break;
            }
        }

        for(int col=nodeCol+1;col<N;col++){ //right
            if(matrix[nodeRow][col]==1){
                nodes.add(nodeRow*N+col);
            }
            else{
                break;
            }
        }


        for(int row=nodeRow-1;row>=0;row--){ //Top
            if(matrix[row][nodeCol]==1){
                nodes.add(row*N+nodeCol);
            }
            else{
                break;
            }
        }


        for(int row=nodeRow+1;row<N;row++){ //Bottom
            if(matrix[row][nodeCol]==1){
                nodes.add(row*N+nodeCol);
```

```
            }
            else{
                break;
            }
        }

        return nodes;
    }
}
```

SESSION 4:

Problem 1:

Corona Virus

ID:11128 Solved By 534 Users

An integer matrix of size **RxC** containing only the values **0**, **1** and **2** is given as the input to the program. The value **0** indicates an **empty** space, the value **1** indicates a person is **healthy** and the value **2** indicates a person is infected by the **corona virus**. Every day the virus is spread from infected person to other persons (all four adjacent persons). The program must print the minimum number of days required to spread the coronavirus to all individuals. If all the persons can not be affected by the corona virus, the program must print **-1** as the output.

Boundary Condition(s):
1 <= R, C <= 1000

Input Format:
The first line contains R and C separated by a space.
The next R lines, each containing C integers separated by a space.

Output Format:
The first line contains -1 or the minimum number of days required to spread the coronavirus to all individuals.

Example Input/Output 1:
Input:
3 5
2 1 0 2 1
1 0 1 2 1
1 0 0 2 1

Output:
2

Explanation:
After Day **1:**
2 2 0 2 2
2 0 2 2 2
1 0 0 2 2

After Day **2:**
2 2 0 2 2
2 0 2 2 2
2 0 0 2 2

Example Input/Output 2:
Input:
3 5
2 1 0 2 1
0 0 1 2 1
1 0 0 2 1

Output:
-1

Max Execution Time Limit: 1000 millisecs

Code:

```
import java.util.*;
public class coronaVirus {

    public static void main(String[] args) {
        //Your Code Here
        Scanner in = new Scanner(System.in);
        int R=in.nextInt();
        int C=in.nextInt();
        int healthy=0,days=0;
        int matrix[][] = new int[R][C];
        Queue<Integer> queue = new ArrayDeque<>();
        for(int row=0;row<R;row++){
            for(int col=0;col<C;col++){
                matrix[row][col]=in.nextInt();
                if(matrix[row][col]==1){
                    healthy++;
                }
                if(matrix[row][col]==2){
                    queue.add(row*C+col);
                }
            }
        }
        queue.add(-1);

        while(!queue.isEmpty()){
            int node=queue.poll();

            if(node==-1){   //-1 is the demarker
                if(!queue.isEmpty()){
                    days++;
                    queue.add(-1);
                }
                continue;
            }

            int row=node/C,col=node%C;
            if(col!=0 && matrix[row][col-1]==1){ //left //left has a healthy
person and he will be infected
                matrix[row][col-1]=2;
                queue.add(row*C+col-1);
                healthy--;
```

```
            }

            if(col!=C-1 && matrix[row][col+1] ==1){ //right
                matrix[row][col+1]=2;
                queue.add(row*C+col+1);
                healthy--;
            }

            if(row!=0 && matrix[row-1][col] ==1){ //top
                matrix[row-1][col]=2;
                queue.add((row-1)*C+col);
                healthy--;
            }

            if(row!=R-1 && matrix[row+1][col] ==1){ //bottom
                matrix[row+1][col]=2;
                queue.add((row+1)*C+col);
                healthy--;
            }
        }

        System.out.println(healthy==0 ? days :-1);
    }
}
```