22075A6713

DAA-Assignment

1. Given a row wise sorted matrix of size **R*C** where R and C are always **odd**, find the median of the matrix.

Test Case 1:

```
Input:
R = 3, C = 3
M = [[1, 3, 5],
     [2, 6, 9],
     [3, 6, 9]]
Output: 5
Explanation: Sorting matrix elements gives
us {1,2,3,3,5,6,6,9,9}.
Hence, 5 is median.
```

- **Constraints:**
  1 <= R, C <= 400
  1 <= matrix[i][j] <= 2000

Program:

//Implementation using C++

main.cpp

Run

```cpp
//Program to find median of a matrix where no. of rows and columns are odd numbered and it should be sorted row wise.
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

const int MAX=2000;
//Defining a function named Median
int Median(int m[][MAX],int r,int c)
{
    if(r>=1&&c<=400){
        int mn=INT_MAX, mx=INT_MIN;
        for (int i=0;i<r;i++)
        {
            if(m[i][0]<mn)
              mn=m[i][0];

            if(m[i][c-1]>mx)
            mx=m[i][c-1];
        }
        int wanted=(r * c+1)/2;
        while(mn<mx)
        {
            int middle=mn +(mx-mn)/2;
            int p=0;

            for (int i=0; i<r; ++i)
```

1°C
Cloudy

Search

main.cpp

```cpp
        mx=m[i][c-1];
        }
        int wanted=(r * c+1)/2;
        while(mn<mx)
        {
            int middle=mn +(mx-mn)/2;
            int p=0;

            for (int i=0; i<r; ++i)
                p+= upper_bound(m[i],m[i]+c,middle)- m[i];
            if(p<wanted)
               mn=middle+1;
            else
            mx=middle;
        }
        return mn;
    }
}

//main
int main()
{
    int r=3,c=3;
    int m[][MAX]={ {1,3,5},{2,6,9},{3,6,9} };
    cout<<"Median is "<<Median(m,r,c)<<endl;
    return 0;
}
```

Output:

Output

```
/tmp/vVJpdNQJEv.o
Median is 5
```

Test Case 1 Passed.

Test Case 2:

```
Input:
R = 3, C = 1
M = [[1], [2], [3]]
Output: 2
Explanation: Sorting matrix elements gives
us {1,2,3}. Hence, 2 is median.
```

Program:

main.cpp

Run

```cpp
//Program to find median of a matrix where no. of rows and columns are odd
    numbered and it should be sorted row wise.
#include<iostream>
#include<bits/stdc++.h>
using namespace std;

const int MAX=2000;
//Defining a function named Median
int Median(int m[][MAX],int r,int c)
{
    if(r>=1&&c<=400){
        int mn=INT_MAX, mx=INT_MIN;
        for (int i=0;i<r;i++)
        {
            if(m[i][0]<mn)
              mn=m[i][0];

            if(m[i][c-1]>mx)
            mx=m[i][c-1];
        }
        int wanted=(r * c+1)/2;
        while(mn<mx)
        {
            int middle=mn +(mx-mn)/2;
            int p=0;

            for (int i=0; i<r; ++i)
```

1°C
Cloudy

Search

main.cpp

Run

```cpp
        mx=m[i][c-1];
        }
        int wanted=(r * c+1)/2;
        while(mn<mx)
        {
            int middle=mn +(mx-mn)/2;
            int p=0;

            for (int i=0; i<r; ++i)
                p+= upper_bound(m[i],m[i]+c,middle)- m[i];
            if(p<wanted)
               mn=middle+1;
            else
            mx=middle;
        }
        return mn;
    }
}

//main
int main()
{
    int r=3,c=1;
    int m[][MAX]={ {1},{2},{3} };
    cout<<"Median  of this is "<<Median(m,r,c)<<endl;
    return 0;
}
```

Output:

Output

```
/tmp/vVJpdNQJEv.o
Median  of this is 2
```

Test Case 2 Passed.

2. Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays that represent the arrival and departure times of trains that stop.

Test Case 1:

Input: arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}, dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}

Program:

//Implementation using Python

main.py Daa.py

```python
#Program to find minimum number of platforms required on a railway station.
# returns minimum number of platforms required.
def findPlatform(arr, dep,n):
    arr.sort()
    dep.sort()
    platform=1
    result=1
    i = 1
    j = 0
    while(i<n and j<n):
        if(arr[i]<= dep[j]):
            platform+=1
            i+=1
        elif(arr[i]>dep[j]):
            platform -= 1
            j += 1
        if(platform>result):
```

Ln: 28, Col: 29

Run Share Command Line Arguments

main.py | Daa.py

```python
        elif(arr[i]>dep[j]):
            platform -= 1
            j += 1
        if(platform>result):
            result=platform
    return result

  # Driver Code

arr=[900,940,950,1100,1500,1800]
dep=[910,1200,1120,1130,1900,2000]
m= len(arr)

print("Minimum Number of Platforms required = ",
    findPlatform(arr,dep,m))
```

Ln: 28, Col: 29

Run | Share | Command Line Arguments

Expected Output: 3

Output:

Run | Share | Command Line Arguments

```
Minimum Number of Platforms required =  3


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Explanation: There are at-most three trains at a time (time between 9:40 to 12:00)

Test Case 1 Passed.

Test Case 2:

Input: arr[] = {9:00, 9:40}, dep[] = {9:10, 12:00}

Program:

main.py Daa.py Daa2.py

```python
def findPlatform(arr, dep,n):
    arr.sort()
    dep.sort()
    platform=1
    result=1
    i = 1
    j = 0
    while(i<n and j<n):
        if(arr[i]<= dep[j]):
            platform+=1
            i+=1
        elif(arr[i]>dep[j]):
            platform -= 1
            j += 1
        if(platform>result):
            result=platform
    return result
```

Ln: 25, Col: 80

main.py Daa.py Daa2.py

```python
            i+=1
        elif(arr[i]>dep[j]):
            platform -= 1
            j += 1
        if(platform>result):
            result=platform
    return result

  # Driver Code

arr=[900,940]
dep=[910,1200]
n= len(arr)

print("Minimum Number of Platforms required for this = ",findPlatform(arr,dep,n))
```

Ln: 6, Col: 11

Expected Output: 1

Output:

Run Share Command Line Arguments

```
Minimum Number of Platforms required for this =  1


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Explanation: Only one platform is needed.
Test Case 2 Passed.

22075A6713	**https://github.com/chaitanya-3009/DAA-Assignment.git**