

DAA-Assignment

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

Test Case 1:

Input:

$R = 3, C = 3$

$M = \begin{bmatrix} 1, & 3, & 5 \\ 2, & 6, & 9 \\ 3, & 6, & 9 \end{bmatrix}$

Output: 5

Explanation: Sorting matrix elements gives us $\{1, 2, 3, 3, 5, 6, 6, 9, 9\}$.
Hence, 5 is median.

➤ **Constraints:**

$1 \leq R, C \leq 400$

$1 \leq \text{matrix}[i][j] \leq 2000$

Program:

//Implementation using C++

```
main.cpp  [Icons] [Run]

1  //Program to find median of a matrix where no. of rows and columns are odd
    numbered and it should be sorted row wise.
2  #include<iostream>
3  #include<bits/stdc++.h>
4  using namespace std;
5
6  const int MAX=2000;
7  //Defining a function named Median
8  int Median(int m[][MAX],int r,int c)
9  {
10     if(r>=1&&c<=400){
11         int mn=INT_MAX, mx=INT_MIN;
12         for (int i=0;i<r;i++)
13         {
14             if(m[i][0]<mn)
15                 mn=m[i][0];
16
17             if(m[i][c-1]>mx)
18                 mx=m[i][c-1];
19         }
20         int wanted=(r * c+1)/2;
21         while(mn<mx)
22         {
23             int middle=mn +(mx-mn)/2;
24             int p=0;
25
26             for (int i=0; i<r; i++)
```

1°C
cloudy

[Windows Taskbar Icons]

```
main.cpp
18     mx=m[l][c-1],
19 }
20 int wanted=(r * c+1)/2;
21 while(mn<mx)
22 {
23     int middle=mn +(mx-mn)/2;
24     int p=0;
25
26     for (int i=0; i<r; ++i)
27         p+= upper_bound(m[i],m[i]+c,middle)- m[i];
28     if(p<wanted)
29         mn=middle+1;
30     else
31         mx=middle;
32 }
33 return mn;
34 }
35 }
36
37 //main
38 int main()
39 {
40     int r=3,c=3;
41     int m[][MAX]={ {1,3,5},{2,6,9},{3,6,9} };
42     cout<<"Median is "<<Median(m,r,c)<<endl;
43     return 0;
44 }
```

0°C
cloudy

Search

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]

1  //Program to find median of a matrix where no. of rows and columns are odd
    numbered and it should be sorted row wise.
2  #include<iostream>
3  #include<bits/stdc++.h>
4  using namespace std;
5
6  const int MAX=2000;
7  //Defining a function named Median
8  int Median(int m[][MAX],int r,int c)
9  {
10     if(r>=1&&c<=400){
11         int mn=INT_MAX, mx=INT_MIN;
12         for (int i=0;i<r;i++)
13         {
14             if(m[i][0]<mn)
15                 mn=m[i][0];
16
17             if(m[i][c-1]>mx)
18                 mx=m[i][c-1];
19         }
20         int wanted=(r * c+1)/2;
21         while(mn<mx)
22         {
23             int middle=mn +(mx-mn)/2;
24             int p=0;
25
26             for (int i=0; i<r; ++i)
```

1°C
cloudy

🪟 🔍 Search 🖨️ 💬 📁

```
main.cpp
18     mx=m[i][c-1],
19 }
20 int wanted=(r * c+1)/2;
21 while(mn<mx)
22 {
23     int middle=mn +(mx-mn)/2;
24     int p=0;
25
26     for (int i=0; i<r; ++i)
27         p+= upper_bound(m[i],m[i]+c,middle)- m[i];
28     if(p<wanted)
29         mn=middle+1;
30     else
31         mx=middle;
32 }
33 return mn;
34 }
35 }
36
37 //main
38 int main()
39 {
40     int r=3,c=1;
41     int m[][MAX]={ {1},{2},{3} };
42     cout<<"Median of this is "<<Median(m,r,c)<<endl;
43     return 0;
44 }
```

1°C
loudy

Search

Output:

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

Test Case 2 Passed.

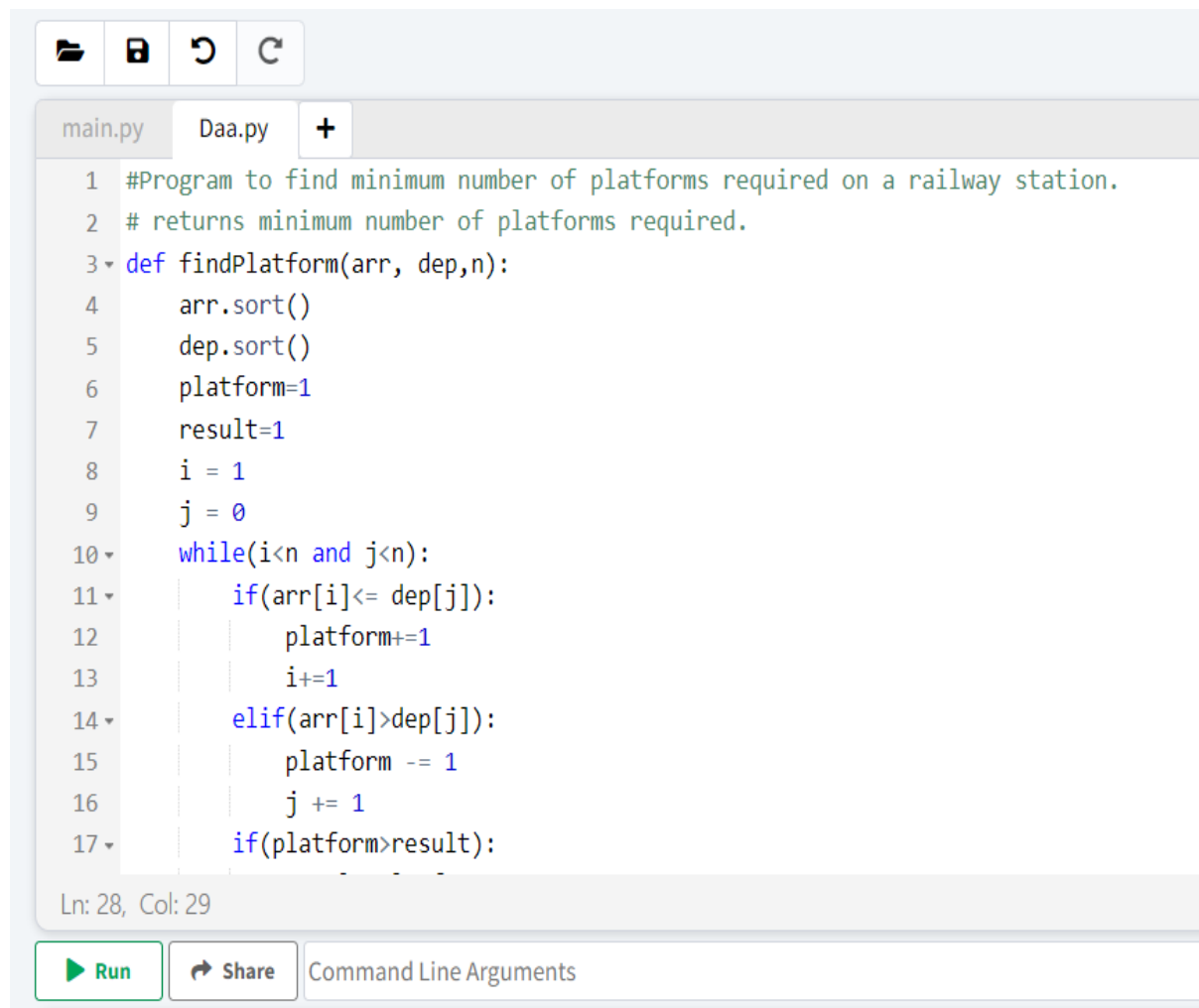
2. Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays that represent the arrival and departure times of trains that stop.

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:

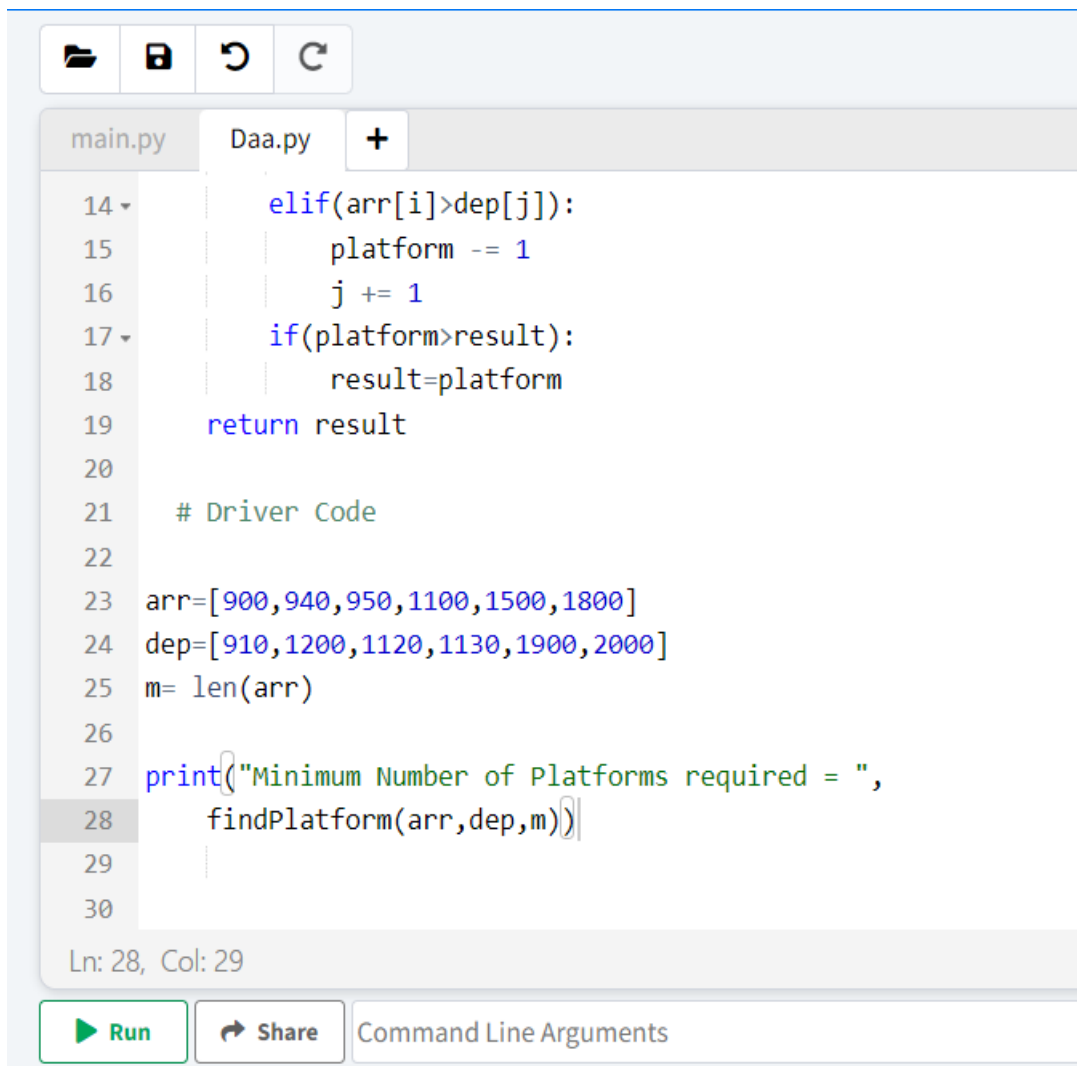
//Implementation using Python



```
1 #Program to find minimum number of platforms required on a railway station.
2 # returns minimum number of platforms required.
3 def findPlatform(arr, dep,n):
4     arr.sort()
5     dep.sort()
6     platform=1
7     result=1
8     i = 1
9     j = 0
10    while(i<n and j<n):
11        if(arr[i]<= dep[j]):
12            platform+=1
13            i+=1
14        elif(arr[i]>dep[j]):
15            platform -= 1
16            j += 1
17        if(platform>result):
```

Ln: 28, Col: 29

[Run](#) [Share](#) Command Line Arguments



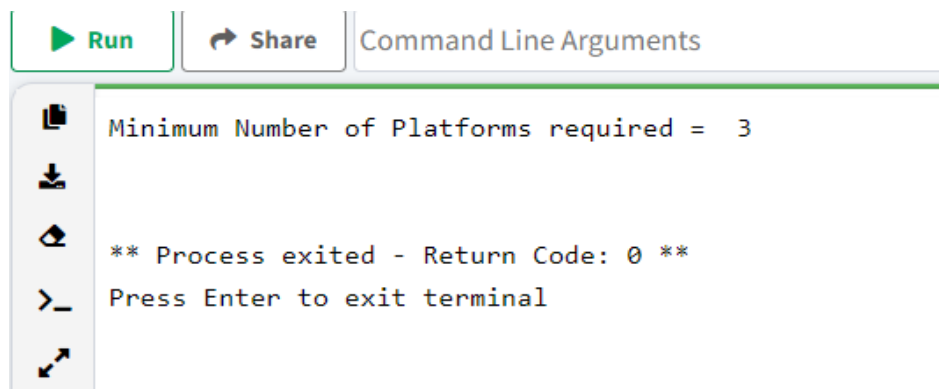
The screenshot shows a code editor with two tabs: 'main.py' and 'Daa.py'. The 'Daa.py' tab is active, displaying the following Python code:

```
14     elif(arr[i]>dep[j]):
15         platform -= 1
16         j += 1
17     if(platform>result):
18         result=platform
19     return result
20
21     # Driver Code
22
23 arr=[900,940,950,1100,1500,1800]
24 dep=[910,1200,1120,1130,1900,2000]
25 m= len(arr)
26
27 print("Minimum Number of Platforms required = ",
28       findPlatform(arr,dep,m))
29
30
```

Below the code editor, there are buttons for 'Run' (a green play icon), 'Share' (a share icon), and a text input field for 'Command Line Arguments'.

Expected Output: 3

Output:



The screenshot shows the output of the program in a terminal window. The output is as follows:

```
Minimum Number of Platforms required = 3
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

On the left side of the terminal window, there are icons for copy, download, share, and a terminal icon.

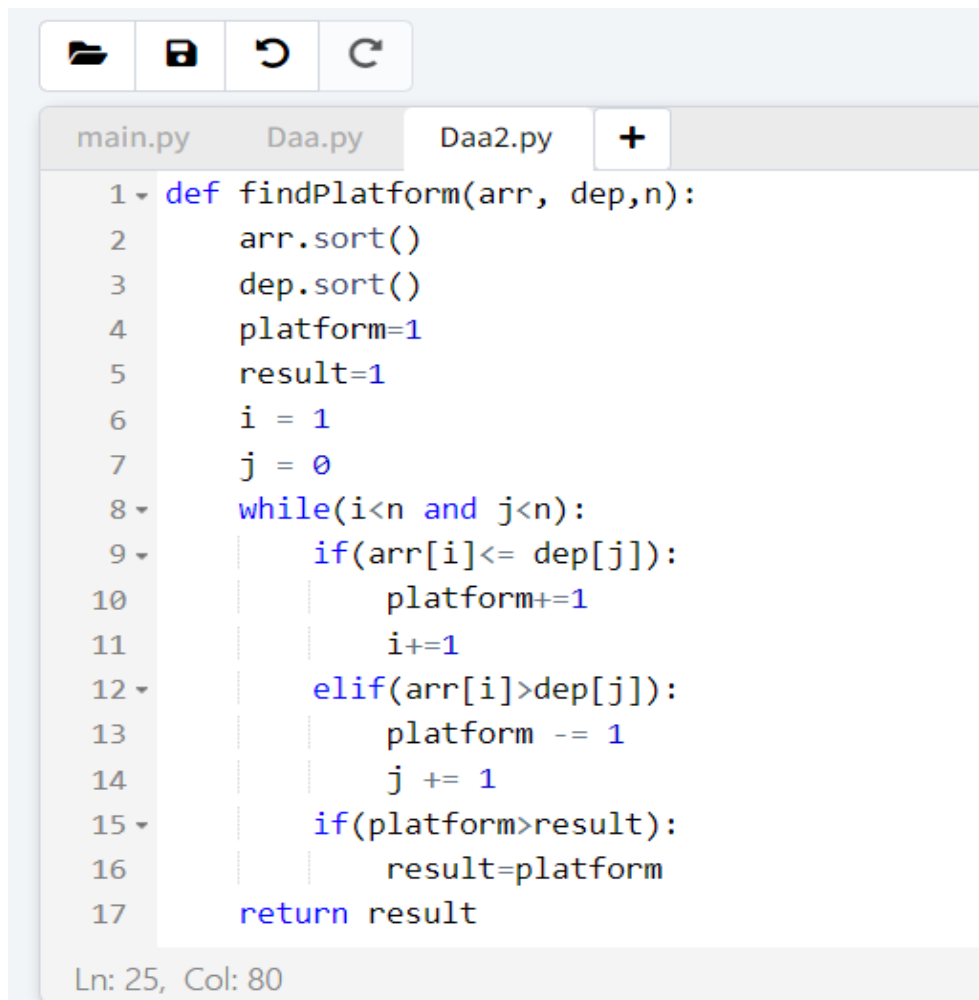
Explanation: There are at-most three trains at a time (time between 9:40 to 12:00)

Test Case 1 Passed.

Test Case 2:

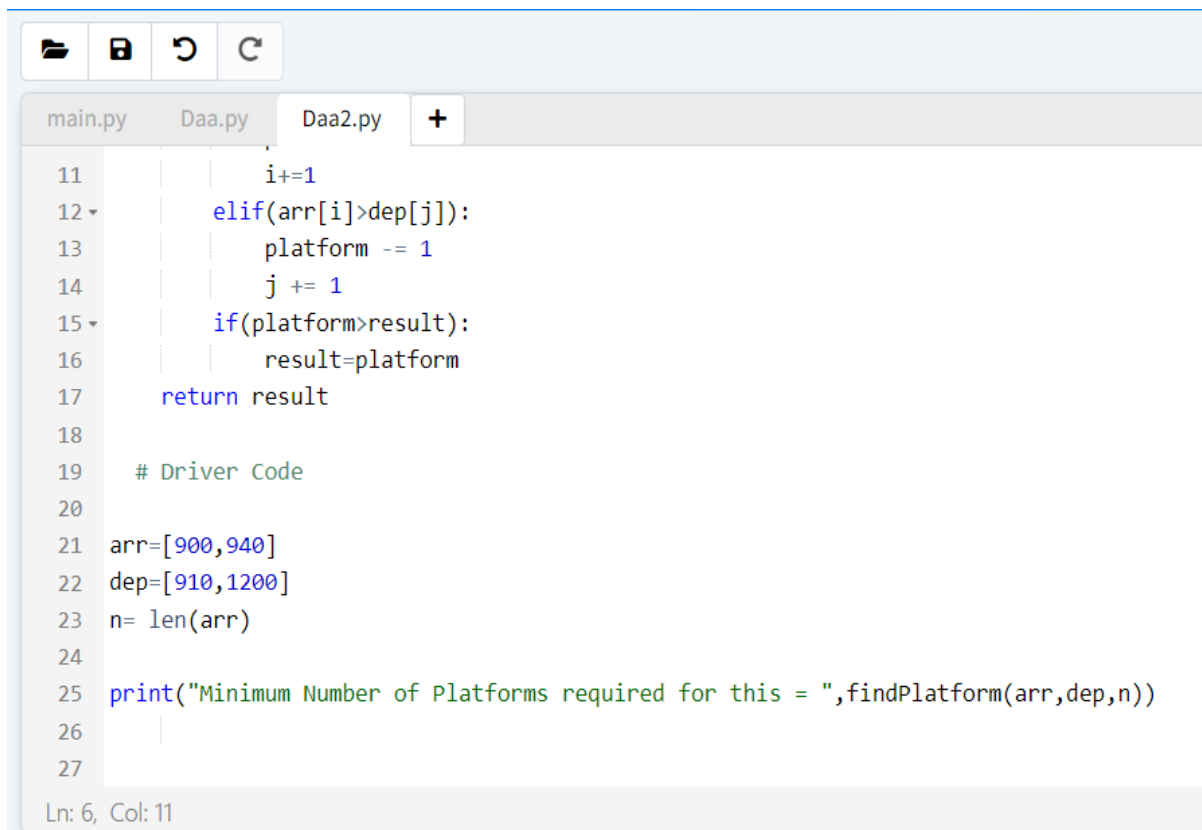
Input: arr[] = {9:00, 9:40}, dep[] = {9:10, 12:00}

Program:



```
1 def findPlatform(arr, dep, n):
2     arr.sort()
3     dep.sort()
4     platform=1
5     result=1
6     i = 1
7     j = 0
8     while(i<n and j<n):
9         if(arr[i]<= dep[j]):
10             platform+=1
11             i+=1
12         elif(arr[i]>dep[j]):
13             platform -= 1
14             j += 1
15         if(platform>result):
16             result=platform
17     return result
```

Ln: 25, Col: 80

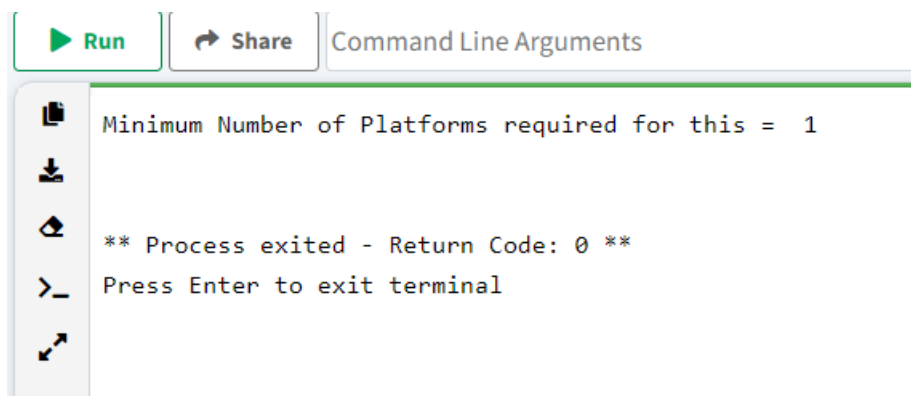


```
11         i+=1
12     elif(arr[i]>dep[j]):
13         platform -= 1
14         j += 1
15     if(platform>result):
16         result=platform
17     return result
18
19     # Driver Code
20
21 arr=[900,940]
22 dep=[910,1200]
23 n= len(arr)
24
25 print("Minimum Number of Platforms required for this = ",findPlatform(arr,dep,n))
26
27
```

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>

