**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:**

from bisect import bisect\_right as upper\_bound

M=100;

def binaryMedian(n, x, y):

mi=n[0][0]

mx=0

for i in range(x):

if n[i][0]<mi:

mi=n[i][0]

if n[i][y-1] > mx :

mx=n[i][y-1]

desired=(x\*y+1)//2

while (mi<mx):

mid=mi+(mx-mi)//2

place=[0];

for i in range(x):

j=upper\_bound(n[i], mid)

place[0]=place[0]+j

if place[0]<desired:

mi=mid+1

else:

mx=mid

print ("Median is", mi)

return

x, y = 3, 3

n=[ [1, 3, 5], [2, 6, 9], [3, 6, 9]]

binaryMedian(n, x, y)

Output:



**Test Case 2:**

from bisect import bisect\_right as upper\_bound

M=100;

def binaryMedian(n, x, y):

mi=n[0][0]

mx=0

for i in range(x):

if n[i][0]<mi:

mi=n[i][0]

if n[i][y-1] > mx :

mx=n[i][y-1]

desired=(x\*y+1)//2

while (mi<mx):

mid=mi+(mx-mi)//2

place=[0];

for i in range(x):

j=upper\_bound(n[i], mid)

place[0]=place[0]+j

if place[0]<desired:

mi=mid+1

else:

mx=mid

print ("Median is", mi)

return

x, y = 3,1

n = [ [1], [2], [3]]

binaryMedian(n, x, y)

Output:



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:**

import heapq

def fp(a, d, n):

a2=[]

for i in range(n):

a2.append([a[i], d[i]])

a2.sort()

p=[]

count=1

heapq.heappush(p, a2[0][1])

for i in range(1, n):

if p[0]>=a2[i][0]:

count+=1

else:

heapq.heappop(p)

heapq.heappush(p, a2[i][1])

return count

if \_\_name\_\_ == "\_\_main\_\_":

a=[900, 940, 950, 1100, 1500, 1800]

d=[910, 1200, 1120, 1130, 1900, 2000]

n=len(a)

print(fp(a, d, n))

Output:



**Test Case 2:**

import heapq

def fp(a, d, n):

a2=[]

for i in range(n):

a2.append([a[i], d[i]])

a2.sort()

p=[]

count=1

heapq.heappush(p, a2[0][1])

for i in range(1, n):

if p[0]>=a2[i][0]:

count+=1

else:

heapq.heappop(p)

heapq.heappush(p, a2[i][1])

return count

if \_\_name\_\_ == "\_\_main\_\_":

a=[900, 940]

d=[910, 1200]

n=len(a)

print(fp(a, d, n))

Output:



**RollNo: 21071A6792**

**GitHub Link: https://github.com/rithika0110/DAA-ASSIGNMENT.git**