Data Structures and Algo in Java - Day 28

import java.util.Arrays;

public class day28

{

public static void main (String [] args)

{

// int arr [][] = {

// {0,0,1,1,1},

// {0,0,0,0,0},

// {0,1,1,1,1},

// {0,0,1,1,1},

// {0,1,1,1,1},

// };

// findMaxOnesin2DArray(arr);

// int arr [] [] = {

// {3,4,7,9},

// {12,13,16,18},

// {20,21,23,29},

// };

// int target = 0;

// findTargetin2DArrayOptimized(arr,target);

int arr [] [] = {

{1,4,7,11,15},

{2,5,8,12,19},

{3,6,9,16,22},

{10,13,14,17,24},

{18,21,23,26,30},

};

int target = 23;

int ans [] = searchInA2DMatrix(arr, target);

System.out.println(Arrays.toString(ans));

}

public static void findMaxOnesin2DArray(int arr [] [])

{

int n = arr.length;

int m = arr[0].length;

int max\_count = 0;

int index = -1;

for(int i=0;i<n;i++)

{

int count\_ones = m - lowerBound(arr[i],m,1);

System.out.println(count\_ones);

if(count\_ones > max\_count)

{

max\_count = count\_ones;

index = i;

}

}

System.out.println("index is "+index);

}

public static int lowerBound(int arr[], int n , int x) //{0,0,1,1,1},

{

int low = 0;

int high = n-1;

int ans = n;

while(low<=high)

{

int mid = (low+high)/2;

if(arr[mid]>=x)

{

ans = mid;

high = mid -1;

}

else

{

low = mid +1;

}

}

return ans;

}

public static void findTargetin2DArray(int arr [] [], int target)

{

int n = arr.length;

int m = arr[0].length;

boolean found = false;

for(int i=0;i<n;i++)

{

if(arr[i][0]<= target && arr[i][m-1]>= target)

{

if(binarySearch(arr[i],target))

{

found = true;

break;

}

}

}

if(found)

{

System.out.println("Target Found");

}

else

{

System.out.println("Not Found");

}

}

public static boolean binarySearch(int arr [] , int tar)

{

int low = 0;

int high = arr.length-1;

while(low<=high)

{

int mid = (low+high)/2;

if(arr[mid]==tar)

{

return true;

}

else if(arr[mid]>tar)

{

high = mid-1;

}

else

{

low = mid +1;

}

}

return false;

}

public static void findTargetin2DArrayOptimized(int arr [] [], int target)

{

int n = arr.length;

int m = arr[0].length;

int low = 0;

int high = (n\*m-1);

boolean found = false;

while(low<=high)

{

int mid = (low+high)/2;

int row = mid/m;

int col = mid%m;

if(arr[row][col]==target)

{

found = true;

break;

}

else if(arr[row][col]<target)

{

low = mid+1;

}

else

{

high = mid-1;

}

}

if(found)

{

System.out.println("Found");

}

else{

System.out.println("Not Found");

}

}

public static int[] searchInA2DMatrix(int arr [] [], int target)

{

/\* better solution - go loop from i to n-1

keep index as a constant where index is equal to binary search of matrix i and target

we will get a index if we find the target so if index is not equal to -1 then return i,index

loop outside just return -1,-1.

\*/

int n = arr.length;

int m = arr[0].length;

int row = 0;

int col = m-1;

while(row<n && col>=0)

{

if(arr[row][col]==target)

{

return new int[]{row,col};

}

else if(arr[row][col]<target)

{

row++;

}

else{

col--;

}

}

return new int[]{-1,-1};

}

}