Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1

Given an n x n matrix, where every row and column is sorted in non-decreasing order. Find the kth smallest element in the given 2D array.

For example, consider the following 2D array.

```
        10, 20, 30, 40
        15, 25, 35, 45
        24, 29, 37, 48
        32, 33, 39, 50
The 3rd smallest element is 20 and 7th smallest element is 30
```

We strongly recommend to minimize the browser and try this yourself first.

The idea is to use min heap. Following are detailed step.
1) Build a min heap of elements from first row. A heap entry also stores row number and column number.
2) Do following k times.
…a) Get minimum element (or root) from min heap.
…b) Find row number and column number of the minimum element.
…c) Replace root with the next element from same column and min-heapify the root.
3) Return the last extracted root.

Following is C++ implementation of above algorithm.

```
// kth largest element in a 2d array sorted row-wise and column-wise
#include<iostream>
```


```
#include<climits>
using namespace std;

// A structure to store an entry of heap.  The entry contains
// a value from 2D array, row and column numbers of the value
struct HeapNode {
    int val;  // value to be stored
    int r;    // Row number of value in 2D array
    int c;    // Column number of value in 2D array
};

// A utility function to swap two HeapNode items.
void swap(HeapNode *x, HeapNode *y) {
    HeapNode z = *x;
    *x = *y;
    *y = z;
}

// A utility function to minheapify the node harr[i] of a heap
// stored in harr[]
void minHeapify(HeapNode harr[], int i, int heap_size)
{
    int l = i*2 + 1;
    int r = i*2 + 2;
    int smallest = i;
    if (l < heap_size && harr[l].val < harr[i].val)
        smallest = l;
    if (r < heap_size && harr[r].val < harr[smallest].val)
        smallest = r;
    if (smallest != i)
    {
        swap(&harr[i], &harr[smallest]);
        minHeapify(harr, smallest, heap_size);
    }
}

// A utility function to convert harr[] to a max heap
void buildHeap(HeapNode harr[], int n)
{
    int i = (n - 1)/2;
    while (i >= 0)
    {
        minHeapify(harr, i, n);
        i--;
    }
}
```


Popular Posts

```
// This function returns kth smallest element in a 2D array mat[][]
int kthSmallest(int mat[4][4], int n, int k)
{
    // k must be greater than 0 and smaller than n*n
    if (k <= 0 || k > n*n)
       return INT_MAX;

    // Create a min heap of elements from first row of 2D array
    HeapNode harr[n];
    for (int i = 0; i < n; i++)
        harr[i] =  {mat[0][i], 0, i};
    buildHeap(harr, n);

    HeapNode hr;
    for (int i = 0; i < k; i++)
    {
       // Get current heap root
       hr = harr[0];

       // Get next value from column of root's value. If the
       // value stored at root was last value in its column,
       // then assign INFINITE as next value
       int nextval = (hr.r < (n-1))? mat[hr.r + 1][hr.c]: INT_MAX;

       // Update heap root with next value
       harr[0] =  {nextval, (hr.r) + 1, hr.c};

       // Heapify root
       minHeapify(harr, 0, n);
    }

    // Return the value at last extracted root
    return hr.val;
}
```

```
// driver program to test above function
int main()
{
  int mat[4][4] = { {10, 20, 30, 40},
                    {15, 25, 35, 45},
                    {25, 29, 37, 48},
                    {32, 33, 39, 50},
                  };
  cout << "7th smallest element is " << kthSmallest(mat, 4, 7);
  return 0;
}
```

Output:

```
7th smallest element is 30
```

Time Complexity: The above solution involves following steps.
1) Build a min heap which takes O(n) time
2) Heapify k times which takes O(kLogn) time.
Therefore, overall time complexity is O(n + kLogn) time.

The above code can be optimized to build a heap of size k when k is smaller than n. In that case, the kth smallest element must be in first k rows and k columns.

We will soon be publishing more efficient algorithms for finding the kth smallest element.

This article is compiled by Ravi Gupta. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)

Recent Comments

- Bucket Sort
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3
- Sort n numbers in range from 0 to n^2 – 1 in linear time

Writing code in comment? Please use **ideone.com** and share the link here.

21 Comments GeeksforGeeks

Sort by Newest

Munira • 6 days ago

write function called negatives sum that takes an array of integers and returns array

Debabrata • 21 days ago

why we cant apply simple merge technique?

```
#define ROW 4
#define COL 4
void merge(int arr[ROW][COL],int *sa){
int tmpArr[ROW*COL];
int i_end = COL;
int j_end = COL;
int i,ith_row ;
int j ,k;
```

for(ith_row=0;ith_row<col;ith_row++){ sa[ith_row]= arr[0][ith_row]; }= for(ith_
ith_row<row;ith_row++){="" i="0;j=0;" for(k="0;k<i_end+j_end;k++){" if((i<="" "
arr[ith_row][j])){
tmpArr[k] = sa[i++];
}else{
tmpArr[k] = arr[ith_row][j++];
}

}
for(k=0;k<i_end+j_end;k++){ sa[k]="tmpArr[k];" }="" i_end="" +="COL;" }="" }='
*arr,size_t="" size){="" for(size_t="" i="0;i<size-1;i++){" std::cout<<="" arr[i]<
int="" arr[row][col]="{" {1,6,9,15},="" {12,14,20,30},="" {3,6,9,10},="" {40,44,46,
col];="" merge(arr,sorted_array);="" print_1d_array(sorted_array,="" row="" *=

Adarsh Singh • 23 days ago

Using PriorityQueue Library in Java

```
import java.util.*;

class pair<x,y,z>
{
X _data ;
Y _row;
Z _column;

pair(X x , Y y , Z z){_data=x; _row=y; _column=z;}

X data(){return _data;}
Y row(){return _row;}
Z column(){return _column;}

void setData(X x){_data=x;}
```

```
void setColumn(Z z){_column=z;}
```

see more

Reply • Share ›

novice • 23 days ago

Do we really need a heap ? Why not

```
row = 1
col = 1
count = 0

while (count <= k ) {
// checks to ensure row , col < n omitted for simplicity
if (a[row][col+1] < a[row+1][col]) {
++col
if (col > n)
++row
}
else {
++row
if ( row > n)
++col
}

++count
}

return a[row][col]
```

o(k) complexity

Reply • Share ›

Maulish Soni ➚ novice • 19 days ago

How about using RANDOM-SELECTION algo?

^ | v • Reply • Share ›

Mearaj ➚ novice • 20 days ago

I guess you have a bug here, you are checking only right side and dow
corners. Here after 24 , you shd move to 25 not 29.

10, 20, 30, 40
|
15, 25, 35, 45
|
24,--> 29 --> 37, 48

32, 33, 39, 50

^ | v • Reply • Share ›

novice ➚ Mearaj • 20 days ago

You are right. But my main point is that a heap might not be nec

^ | v • Reply • Share ›

sunil • 25 days ago

But this is as good as sorting n sorted arrays i.e you are not making use of the
as well.

^ | v • Reply • Share ›

sunil ➚ sunil • 25 days ago

http://www.geeksforgeeks.org/m...

^ | v • Reply • Share ›

sunil ➚ sunil • 25 days ago

ignore it .. you are taking care of that as well .. good solution

• Reply • Share ›

gg • a month ago

can we do it by using k-way merge?

• Reply • Share ›

Aditya J gg • 8 days ago

k-way merge is done using a heap.

• Reply • Share ›

pulkit mehra gg • 25 days ago

We can, but the time complexity would be more I guess, as we would have to merge the row elements) instead of processing n+k elements method.

• Reply • Share ›

Chenzhan Liu • a month ago

Instead of adding items into heap row-wisely, I suppose the following strategy
(1)pop out the top value and get its row and col
(2)add matrix[row+1][col] and matrix[row][col+1] into heap, if they exist and ha

2 • Reply • Share ›

Aditya Gaurav Chenzhan Liu • 16 days ago

in this method the maximum size of heap at any point of time can O(n- complexity is O(log (size of heap)) and hence the worst case complex

• Reply • Share ›

Ankul Chenzhan Liu • 23 days ago

+1 on above soln. We can start the heap with empty heap each time p the min is popped. This removes the extra overhead of O(n) or O(k).

• Reply • Share ›

pulkit mehra ➦ Chenzhan Liu • a month ago

Can you explain as to how will it be more effective ?

• Reply • Share ›

Ahmed Saleh • a month ago

I would just converted the 2D Array to 1D, sort it, then get kth element.

• Reply • Share ›

Shivam Goel ➦ Ahmed Saleh • 25 days ago

n^2 (logn)^2

• Reply • Share ›

Ahmed Saleh ➦ Shivam Goel • 25 days ago

Converting 2D array to 1D is O(N^2), sort it is O(NLogN).

• Reply • Share ›

omar salem ➦ Ahmed Saleh • a month ago

that's N^2 right there

2 • Reply • Share ›