Lecture 39 Heap

1. <https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/>

class Solution {

public:

int kthSmallest(vector<vector<int>>& matrix, int k) {

int rows = matrix.size();

int cols = matrix[0].size();

priority\_queue<int> pq; // max heap

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

{

for(int j=0; j<cols; j++)

{

pq.push(matrix[i][j]);

if(pq.size() > k)

{

pq.pop();

}

}

}

return pq.top();

}

};

1. <https://leetcode.com/problems/find-median-from-data-stream/>

class MedianFinder {

public:

/\*\* initialize your data structure here. \*/

priority\_queue<int, vector<int>, greater<int>> rightHeap; //minHeap

priority\_queue<int> leftHeap; // maxHeap

MedianFinder() {

}

void addNum(int num) {

if(leftHeap.size()==0)

{

leftHeap.push(num);

}

else

{

double med = findMedian();

if(leftHeap.size() > rightHeap.size())

{

if(num < med)

{

rightHeap.push(leftHeap.top());

leftHeap.pop();

leftHeap.push(num);

}

else

{

rightHeap.push(num);

}

}

else if(leftHeap.size()==rightHeap.size())

{

if(num < med)

{

leftHeap.push(num);

}

else

{

rightHeap.push(num);

}

}

else

{

if(num > med)

{

leftHeap.push(rightHeap.top());

rightHeap.pop();

rightHeap.push(num);

}

else

{

leftHeap.push(num);

}

}

}

}

double findMedian() {

int l = leftHeap.size();

int r = rightHeap.size();

if(l>r)

{

return double(leftHeap.top());

}

else if(l<r)

{

return double(rightHeap.top());

}

else

{

return (double(rightHeap.top())+double(leftHeap.top()))/2;

}

}

};

1. <https://practice.geeksforgeeks.org/problems/merge-k-sorted-arrays/1>

typedef pair<int, pair<int, int>> node;

class Solution

{

public:

//Function to merge k sorted arrays.

vector<int> mergeKArrays(vector<vector<int>> arr, int K)

{

//code here

vector<int> res;

priority\_queue<node, vector<node>, greater<node>> pq;

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

{

pq.push({arr[i][0], {i, 0}});

}

while(!pq.empty())

{

node current=pq.top();

pq.pop();

res.push\_back(current.first);

int x=current.second.first;

int y=current.second.second;

if(y+1<K)

{

pq.push({arr[x][y+1], {x, y+1}});

}

}

return res;

}

};