Application of Priority Queue and Heap

By Ethan Jelo P. Martinez

Question

Given a N x N matrix, where every row and column is sorted in non-decreasing order. Find the kth smallest element in the matrix.

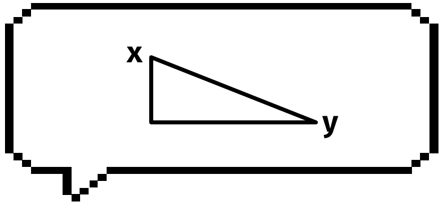
```
Input:
N=4
Mat[][] = {{16,28,60,64},
           {22,41,63,91},
           {27,50,87,83},
           {36,78,87,94}}
K = 3
Output: 27
```

Explanation: 27 is the 3rd smallest element

https://practice.geeksforgeeks.org/problems/kth-element-in-matrix/1

Solution:

There are two ways of solving this problem which is binary search or binary tree specifically a **MIN-HEAP**



01

Components of a Min Heap

01 Push

Adds an item to a heap while maintaining its heap property

02 Pop

Removes an item from a heap while maintaining its heap

03 PercolateDown

Restores the heap property from a child node to a root node.

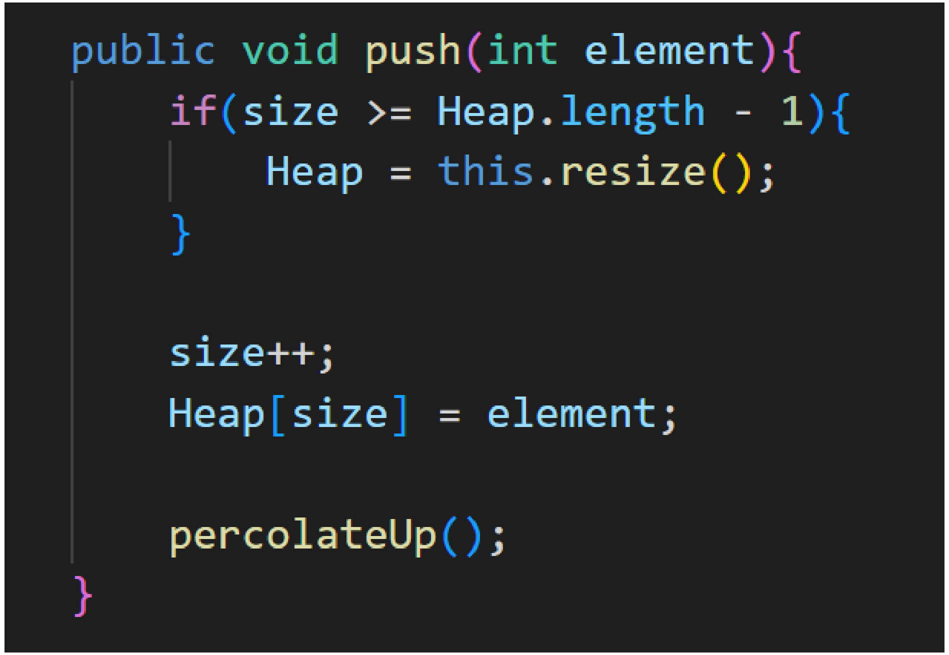
04 PercolateUp

Restores the heap property from a root node to a child node

05 Helper Functions

Functions that allow the system to work

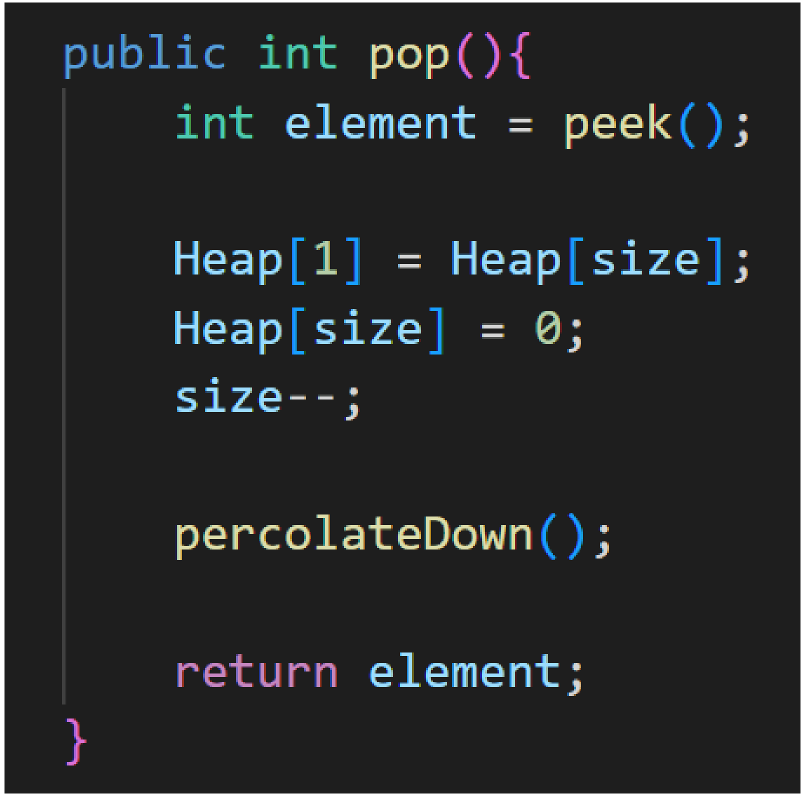
Push



```
public void push(int element){
    if(size >= Heap.length - 1){
        Heap = this.resize();
    }

    size++;
    Heap[size] = element;

    percolateUp();
}
```

Adds a value to the min-heap.

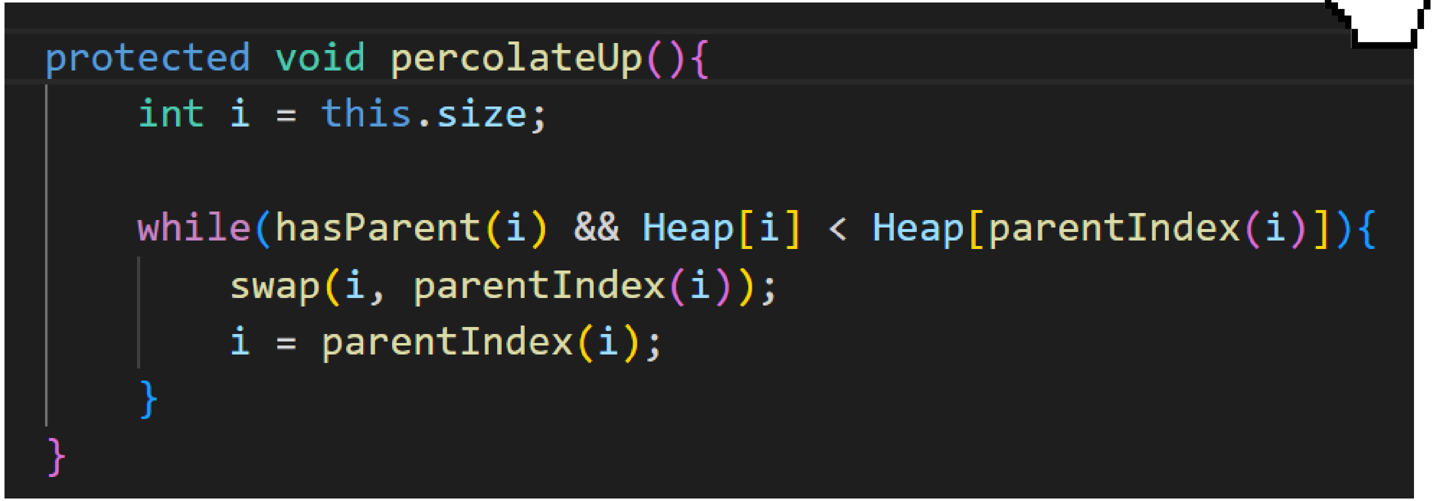
Pop



```
public int pop(){
    int element = peek();

    Heap[1] = Heap[size];
    Heap[size] = 0;
    size--;

    percolateDown();

    return element;
}
```

Removes and returns the minimum element in the heap.

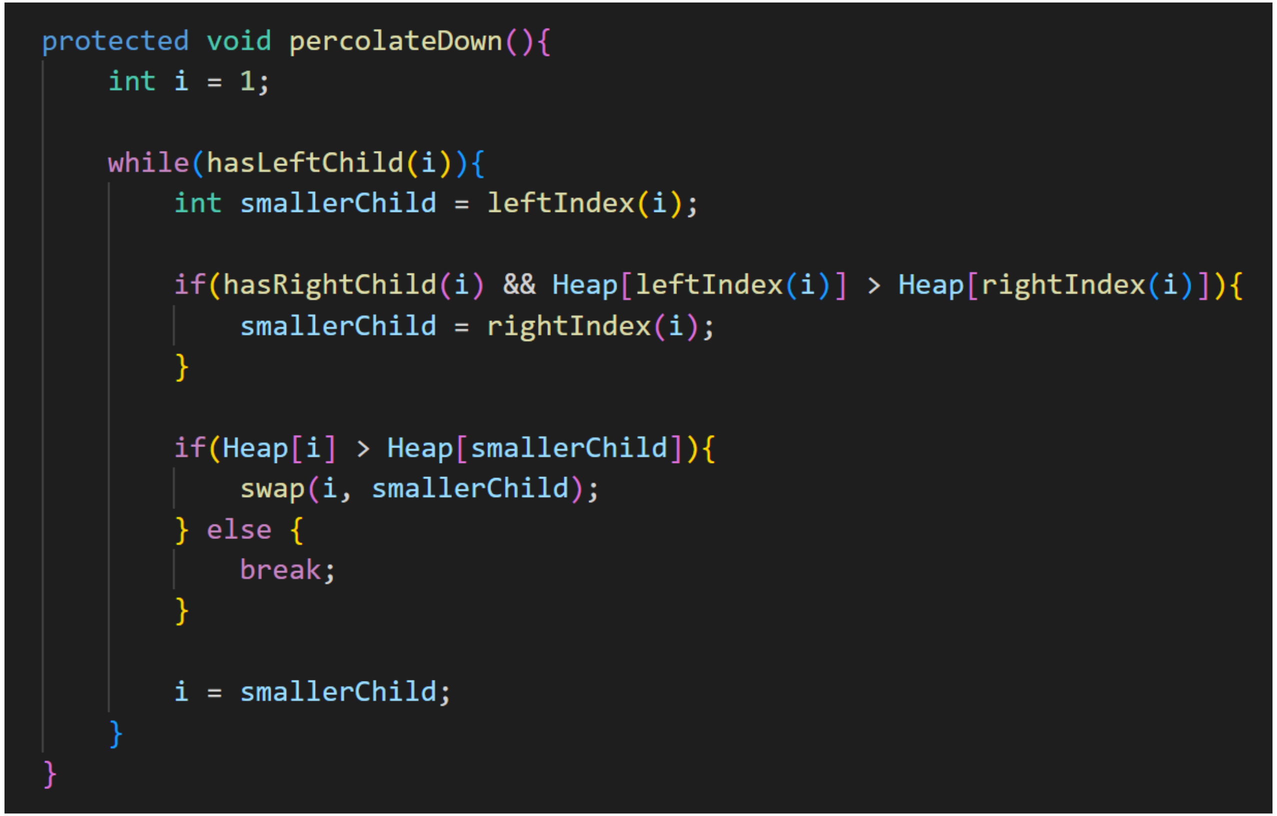
PercolateUp



```java
protected void percolateUp(){
    int i = this.size;

    while(hasParent(i) && Heap[i] < Heap[parentIndex(i)]){
        swap(i, parentIndex(i));
        i = parentIndex(i);
    }
}
```

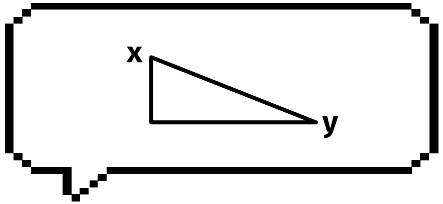
Place a newly inserted element in its correct place so that the heap maintains the min-heap order property.

PercolateDown



```
protected void percolateDown(){
    int i = 1;

    while(hasLeftChild(i)){
        int smallerChild = leftIndex(i);

        if(hasRightChild(i) && Heap[leftIndex(i)] > Heap[rightIndex(i)]){
            smallerChild = rightIndex(i);
        }

        if(Heap[i] > Heap[smallerChild]){
            swap(i, smallerChild);
        } else {
            break;
        }

        i = smallerChild;
    }
}
```

place the element that is at the root of the heap in its correct place so that the heap maintains the min-heap order property.



Helper Functions

01 Peek

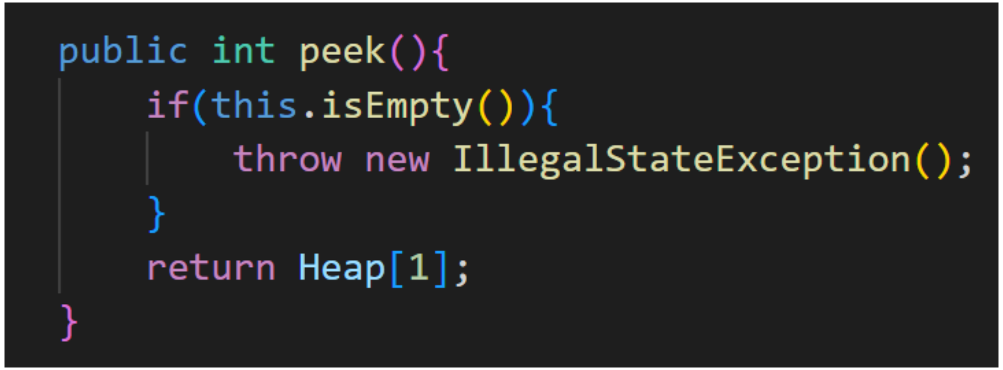
02 Swap

03 Resize

04 Constructor

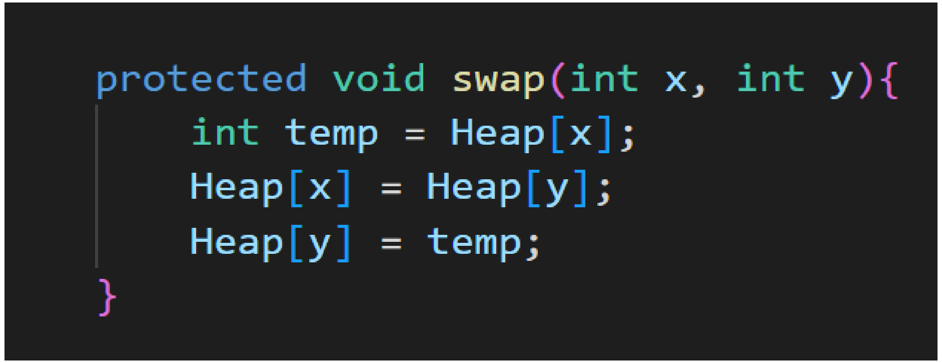
05 Index Functions

01 Peek



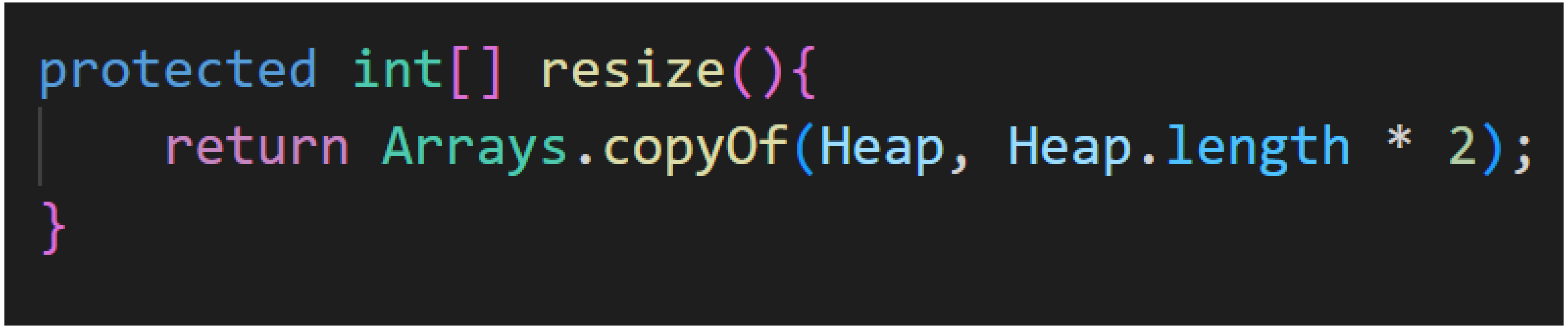
```
public int peek(){
    if(this.isEmpty()){
        throw new IllegalStateException();
    }
    return Heap[1];
}
```

02 Swap



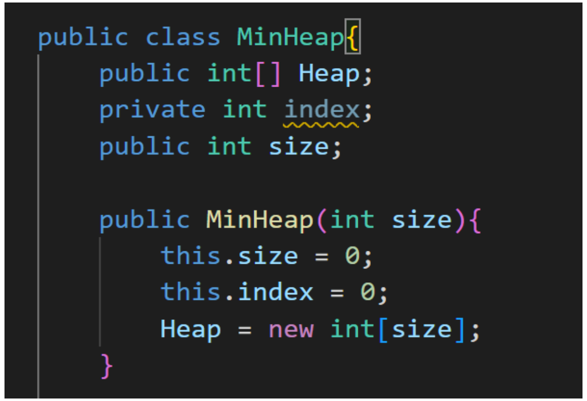
```
protected void swap(int x, int y){
    int temp = Heap[x];
    Heap[x] = Heap[y];
    Heap[y] = temp;
}
```

03 Resize



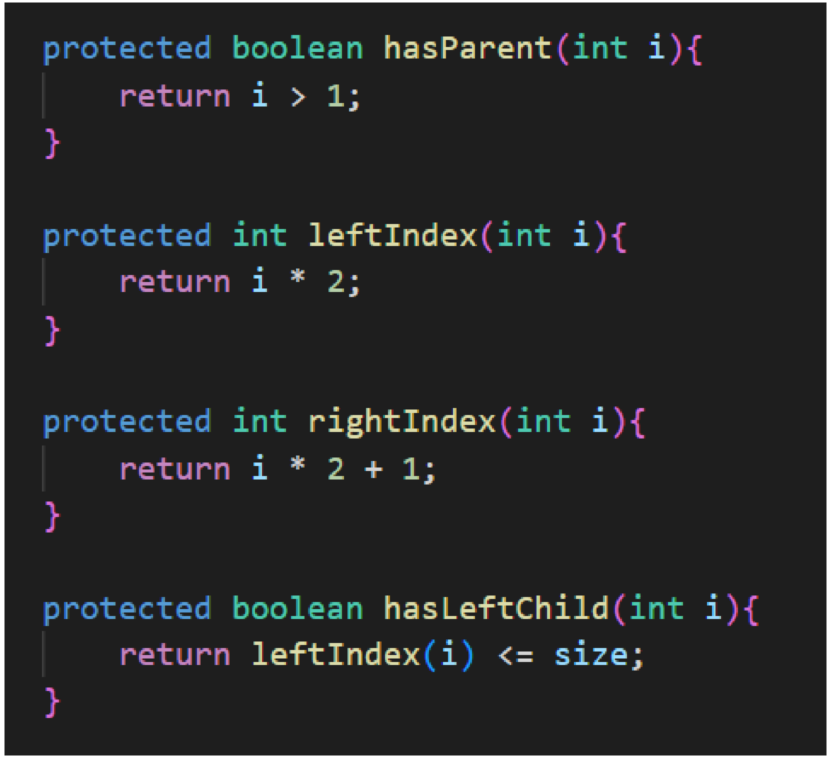
```
protected int[] resize(){
    return Arrays.copyOf(Heap, Heap.length * 2);
}
```

04 Constructor

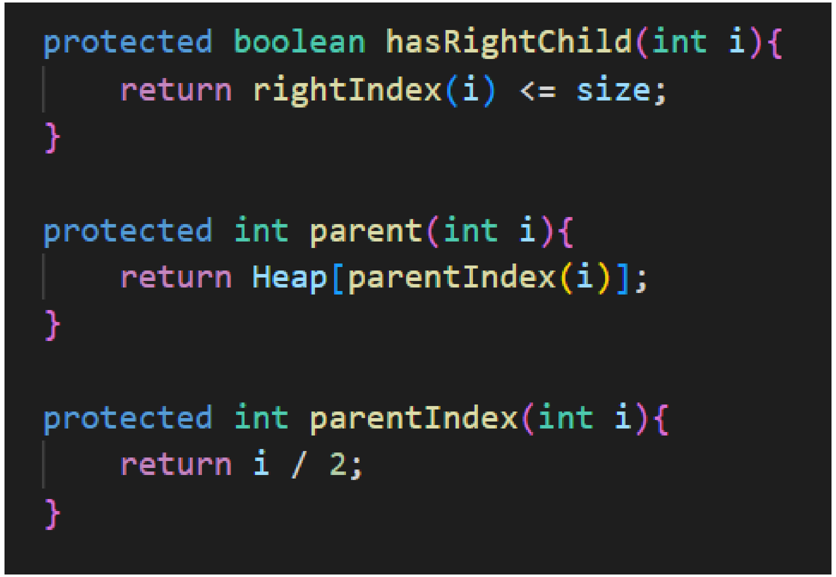


```
public class MinHeap{
    public int[] Heap;
    private int index;
    public int size;

    public MinHeap(int size){
        this.size = 0;
        this.index = 0;
        Heap = new int[size];
    }
```

05 Index Functions

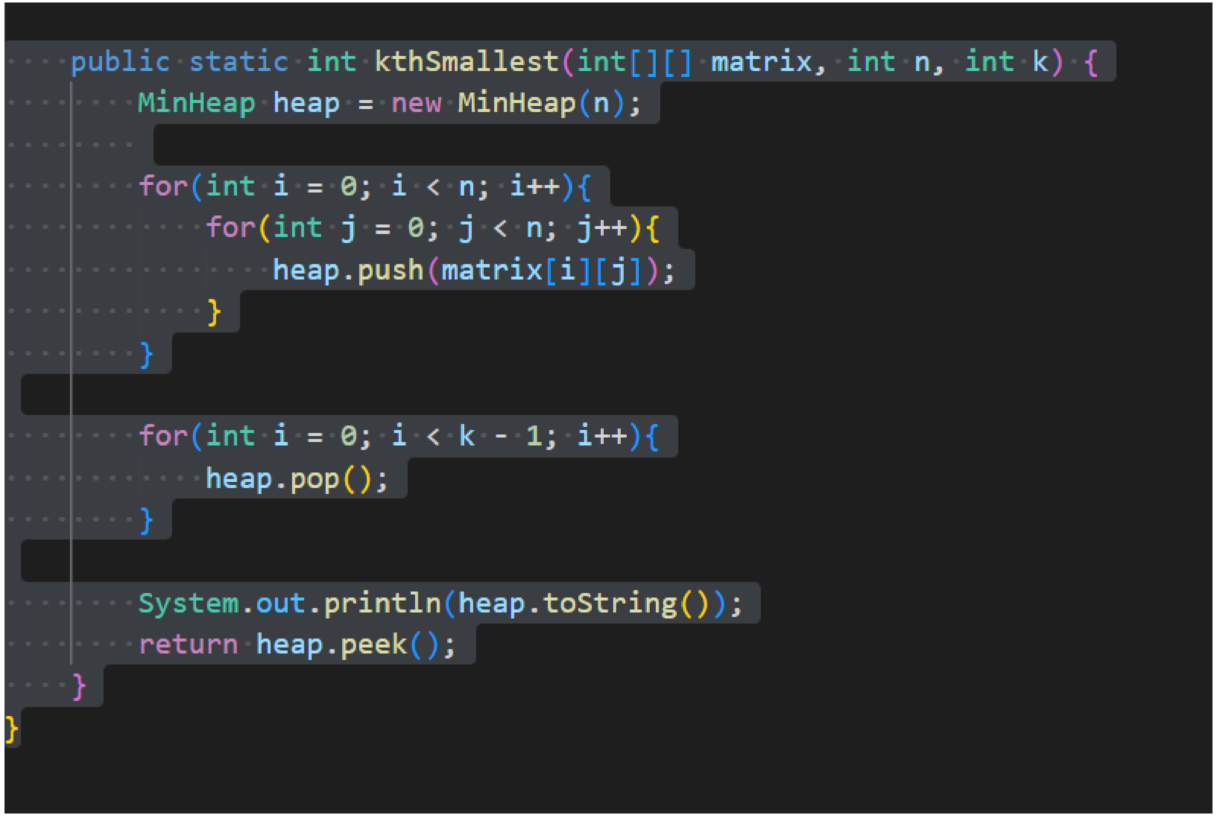


```java
protected boolean hasParent(int i){
    return i > 1;
}

protected int leftIndex(int i){
    return i * 2;
}

protected int rightIndex(int i){
    return i * 2 + 1;
}

protected boolean hasLeftChild(int i){
    return leftIndex(i) <= size;
}
```



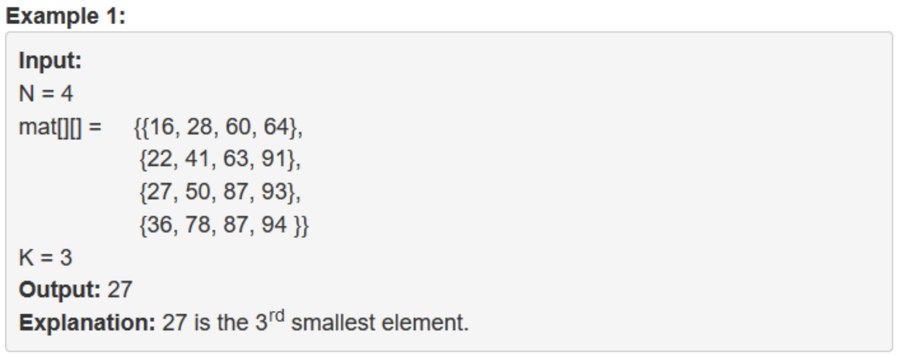
```java
protected boolean hasRightChild(int i){
    return rightIndex(i) <= size;
}

protected int parent(int i){
    return Heap[parentIndex(i)];
}

protected int parentIndex(int i){
    return i / 2;
}
```

Answer to the Question



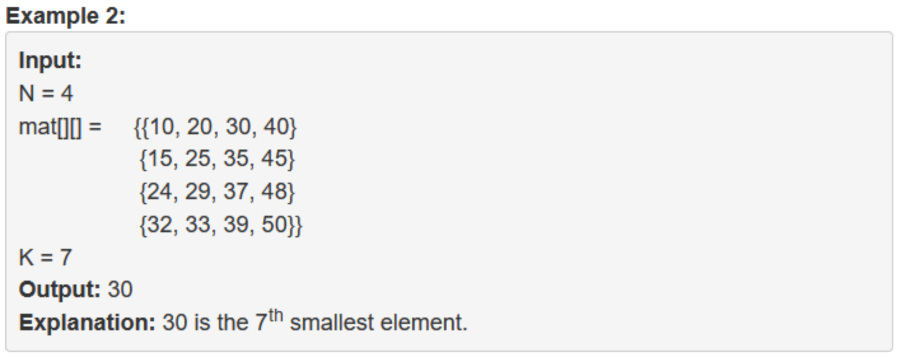
```
    public static int kthSmallest(int[][] matrix, int n, int k) {
        MinHeap heap = new MinHeap(n);

        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                heap.push(matrix[i][j]);
            }
        }

        for(int i = 0; i < k - 1; i++){
            heap.pop();
        }

        System.out.println(heap.toString());
        return heap.peek();
    }
}
```

Answers



Example 1:

Input:
N = 4
mat[][] = {{16, 28, 60, 64},
{22, 41, 63, 91},
{27, 50, 87, 93},
{36, 78, 87, 94 }}
K = 3
Output: 27
Explanation: 27 is the 3rd smallest element.



Example 2:

Input:
N = 4
mat[][] = {{10, 20, 30, 40}
{15, 25, 35, 45}
{24, 29, 37, 48}
{32, 33, 39, 50}}
K = 7
Output: 30
Explanation: 30 is the 7th smallest element.