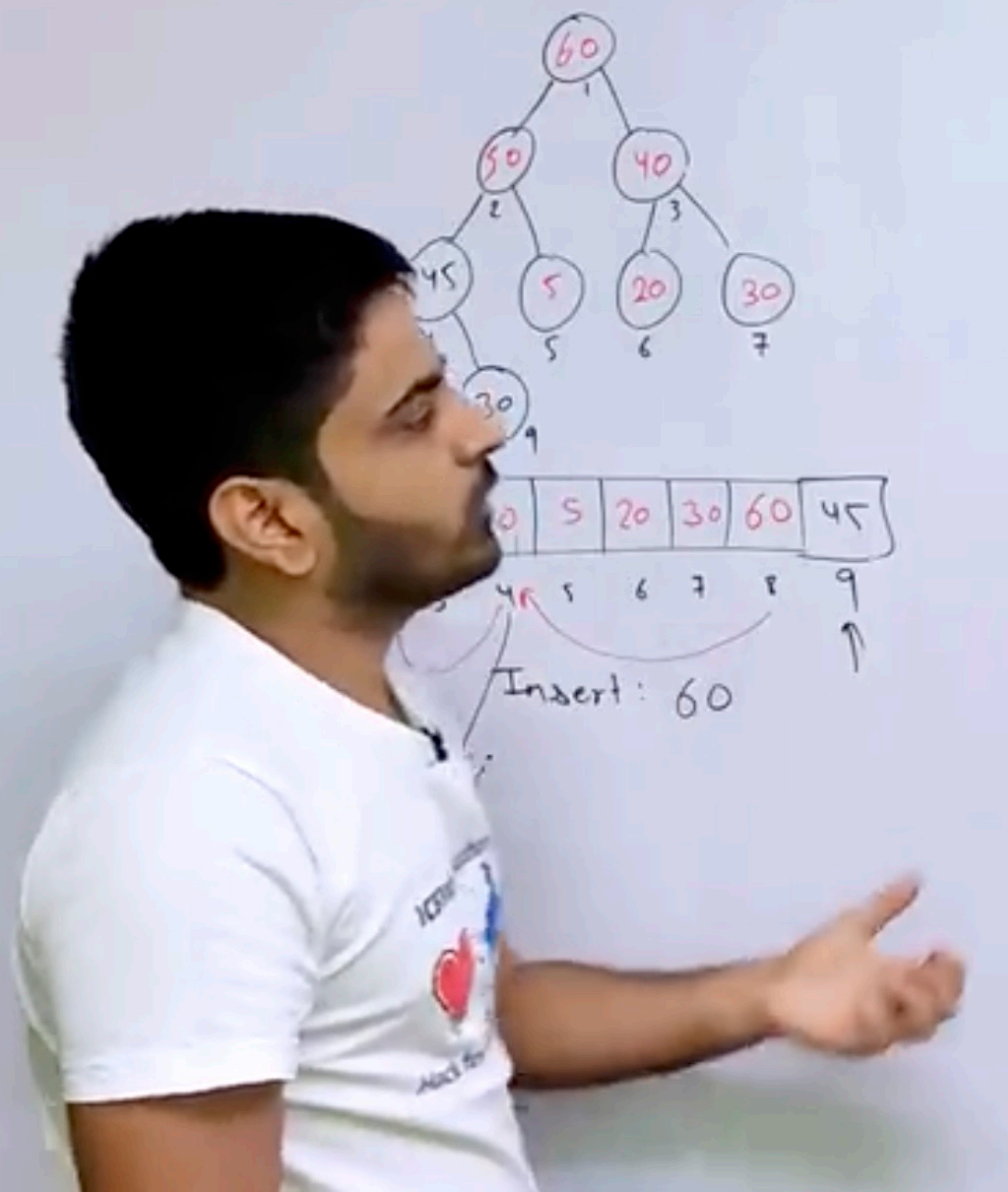
# Heap



```
void insert ( A[ ], n, value) {
    n = n+1;
    A[n] = value;
    int i = n;
    while ( i > 1) {
        int parent = i/2;
        if (a[parent] < a[i]) {
            swap (A, parent, i);
            i = parent;
        } else {
            return;
        }
    }
}
```

# Heap

| | | | | | |
|---|---|---|---|---|---|
| 20 | 10 | 30 | 5 | 50 | 40 |

MAXHEAP

O(n log n)

| | | | | | |
|---|---|---|---|---|---|
| 50 | 30 | 40 | 5 | 10 | 20 |
| 1 | 2 | 3 | 4 | 5 | 6 |

# Heap

```
Void heapify(int a[], int n, int i){
    int largest = i;
    int l = 2*i;
    int r = 2*i+1;
    if(l<=n && a[l] > a[largest]){
        largest = l;
    }
    if(r<=n && a[r] > a[largest]){
        largest = r;
    }
    if(largest != i){
        swap(a, i, largest);
        heapify(a, n, largest);
    }
}
```

| 10 | 30 | 50 | 20 | 35 | 15 |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |

i
2
35
20
30
l
4
5
n
r
largest
10
1
30
2
50
3
20
4
35
5
15
6

# Heap

```
Void heapify(int a[], int n, int i){
    int largest = i;
    int l = 2*i;
    int r = 2*i+1;
    if(l<=n && a[l] > a[largest]){
        largest = l;
    }
    if(r<=n && a[r] > a[largest]){
        largest = r;
    }
    if (largest != i){
        swap(a, i, largest);
        heapify(a, n, largest);
    }
}
```

```
buildheap(int a[], int n){

    for(int i = n/2; i>0; i--){

        heapify(a,n,i);
    }
}
```

$\boxed{O(n)}$

10
30
50
20
35
15
1
2
3
4
5
6
n/2
50
35
15
20
30
10
50 1
35 2
15 3
20 4
30 5
10 6

# Heap Sort

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 40 | 10 | 30 | 50 | 60 | 15 |

Convert into a Heap → O(n)

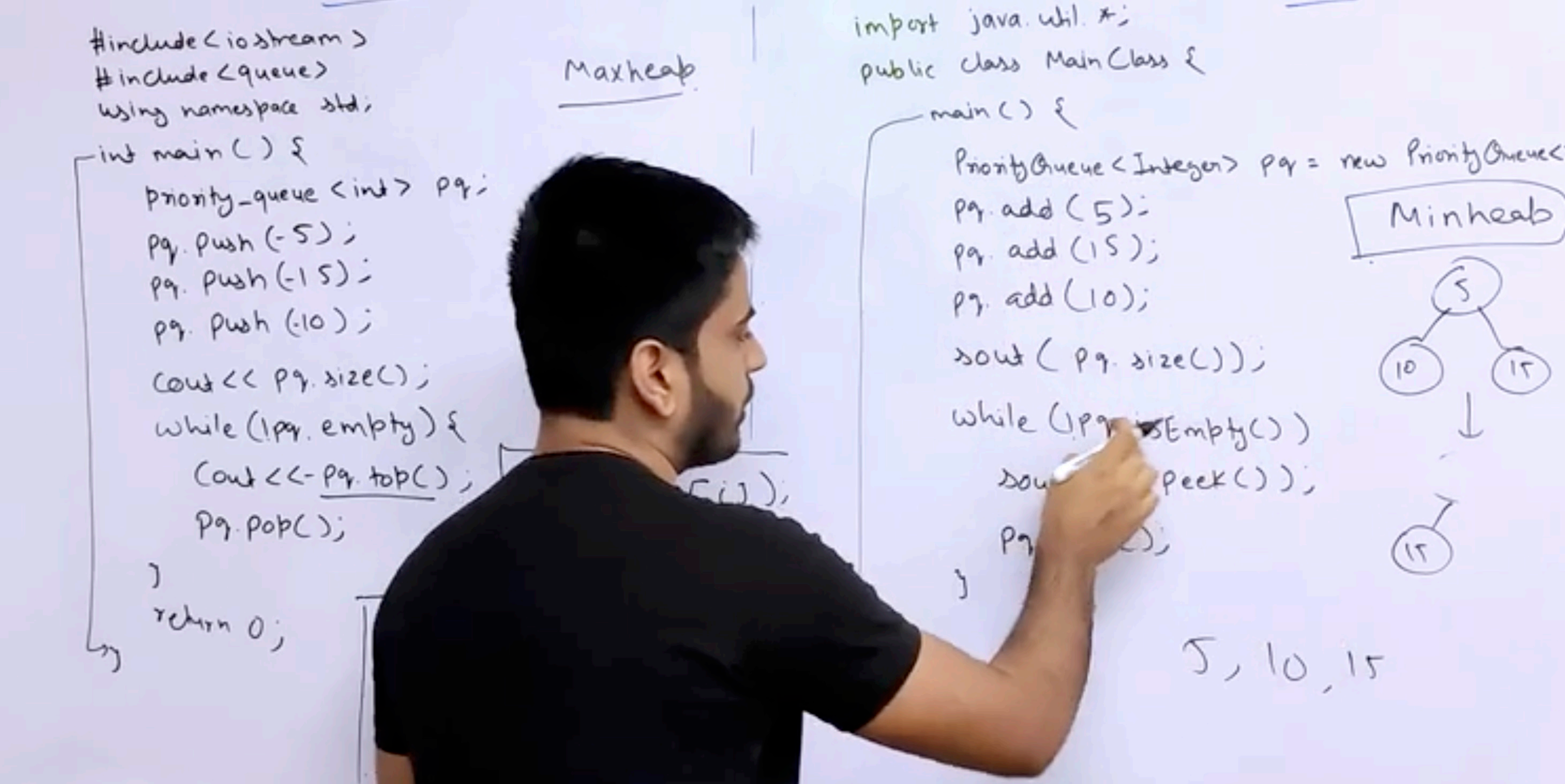
| | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| | 50 | 30 | 40 | 10 | 15 |

Delete elements One by one

Sorted Array

```
void heapSort(int a[], int n) {
    for(int i = n; i > 1; i--) {
        swap(a, 1, i);
        heapify(a, i-1, 1);
    }
}
```

| 1 | | | | | |
|---|---|---|---|---|---|
| 10 | 15 | 30 | 40 | 50 | 60 |



Maxheap

```
#include<iostream>
#include<queue>
using namespace std;
int main() {
    priority_queue <int> pq;
    pq.push(-5);
    pq.push(-15);
    pq.push(-10);
    cout<< pq.size();
    while(!pq.empty) {
        cout<<-pq.top();
        pq.pop();
    }
    return 0;
}
```

```
import java.util.*;
public class MainClass {
  main() {
    PriorityQueue<Integer> pq = new PriorityQueue<
    pq.add(5);
    pq.add(15);
    pq.add(10);
    sout(pq.size());
    while(!pq.isEmpty())
       sou      peek());
       pq       ();
    }
```

Minheap

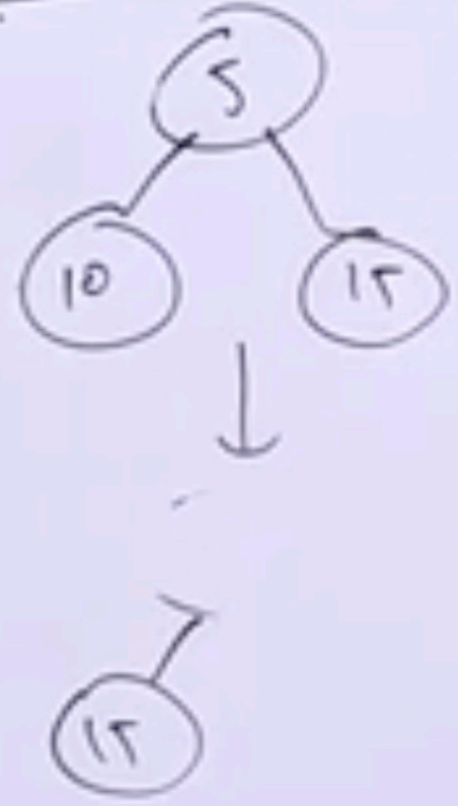
5, 10, 15

JAVA

```
PriorityQueue<Integer> pq = new
        PriorityQueue<>(Collections.reverseOrder());
                      ↑
                  pq is MAXHeap
```

```
import java.util.*;
public class MainClass {
    main() {
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        pq.add(5);
        pq.add(15);
        pq.add(10);
        sout(pq.size());
        while(!pq.isEmpty())
            sout(pq.peek());
            pq.poll();
        }
}
```

Minheab



(5, 10, 15)

# Question:

Print $k^{th}$ largest element in an Array.

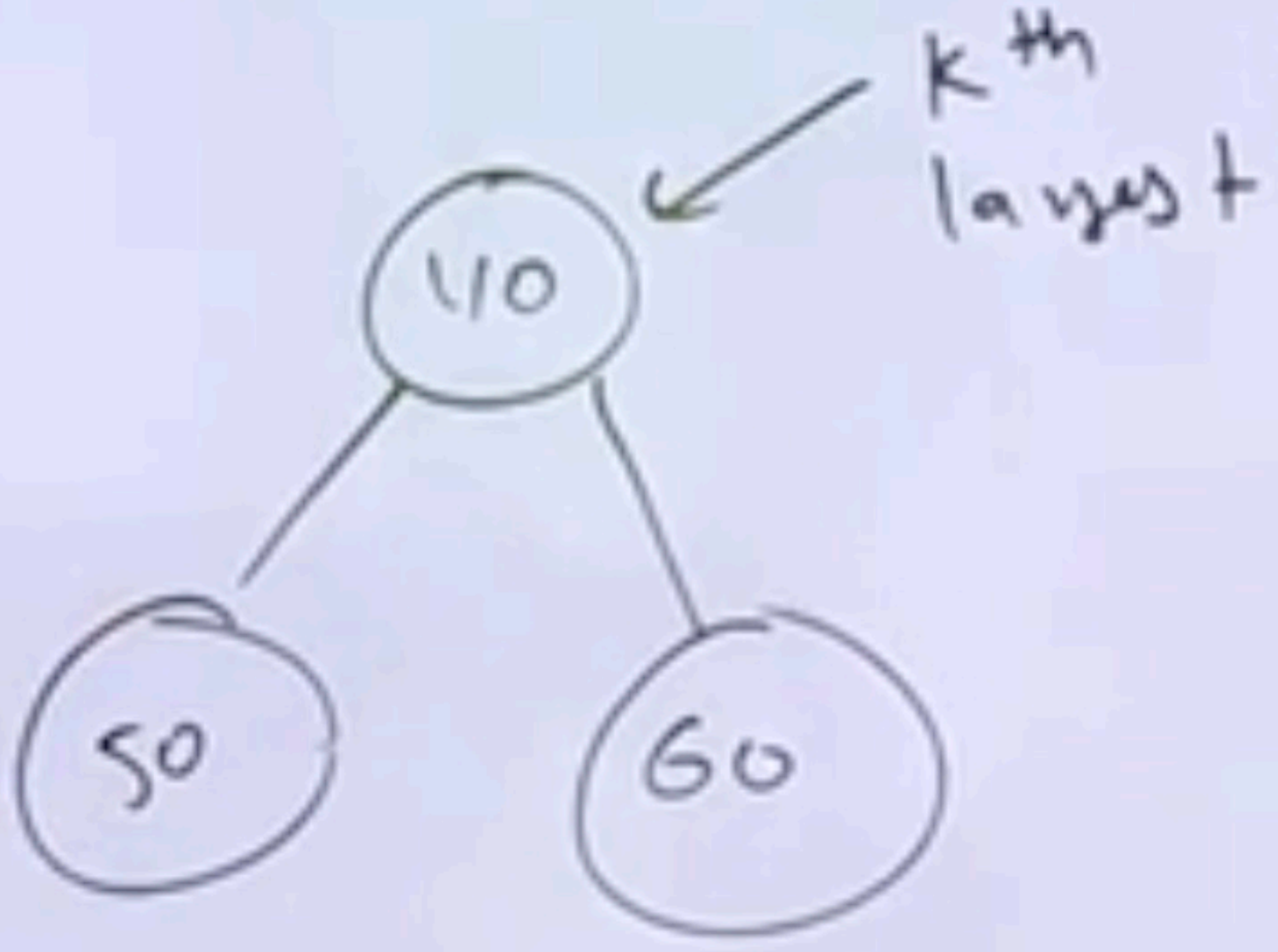
| | | | | | |
|---|---|---|---|---|---|
| 20 | 10 | 60 | 30 | 50 | 40 |

K = 3

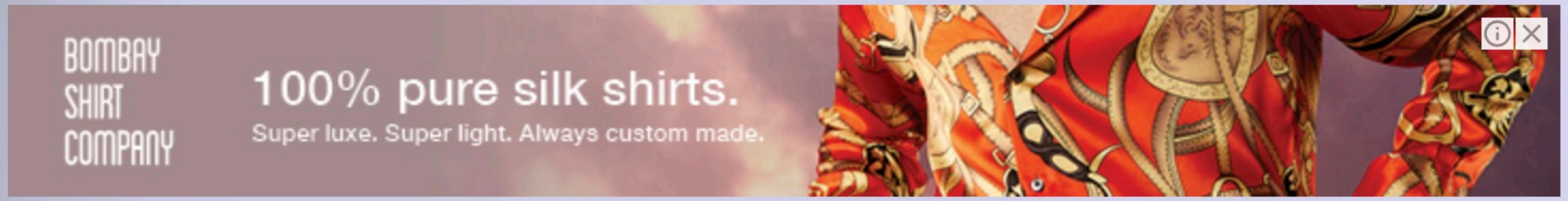
10 20 3 0 50 60

d

## Priority Queue



Min heap.



Find Kth Largest/Smallest Element in an Array | PriorityQueue in Java & C++ | DSA-One Course #33

# Priority Queue

Question :

Print kth largest element in an Array.

| | | | | | |
|---|---|---|---|---|---|
| 20 | 10 | 60 | 30 | 50 | 40 |

k = 3

10 20 30 40 50 60

3rd

```
int kth largest ( int a[], int k) {
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    for(int i=0; i<k; i++) {
        pq.add(a[i]);
    }
    for(int i=k; i<a.length; i++) {
        if(pq.peek() < a[i]) {
            pq.pop();
            pq.add(a[i]);
        }
    }
    return pq.peek();
}
```

are and hit that Subscribe button... It motivates me to m

11:40 / 13:14