CSE12 - Lecture 20 - Notes

Monday, November 20, 2023 8:00 AM

complete tree → filled in left to right
max heap → parent keys >= children keys

Lecture 20

Max Heap → largest value comes out first → is the root

Priority Queue

Assume the key and value are identical for this example

Draw the picture of the tree and the array for the following:

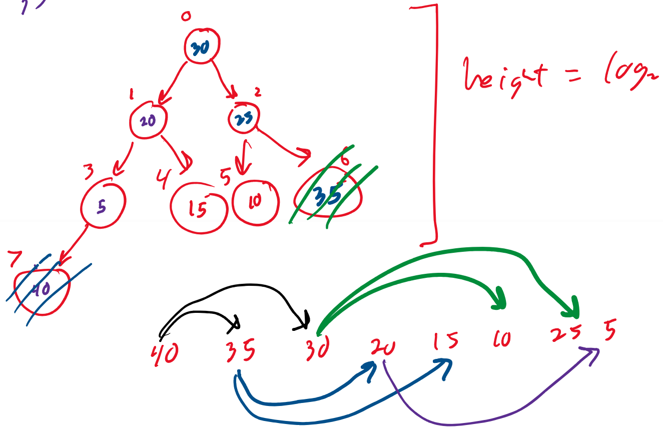
ArrayList<Integer> heap = new ArrayList<>(2); //initial capacity of 2

Add the following elements to the max heap (in this order):

5, 10, 15, 20, 25, 30, 35, 40

Call poll() twice

What elements were returned? 40, 35



height = $\log_2(n)$

add(5)

0	1
5	
5	

add(10)

5	10
10	5

add(15)

0	1	2	3
10	5	15	
15	5	10	

add(20)

15	5	10	20
20	15	10	5

add(25)

0	1	2	3	4	5	6	7
20	15	10	5	25			
25	20	10	5	15			

add(30)

25	20	10	5	15	30		
30	20	25	5	15	10		

add(35)

30	20	25	5	15	10	35	
35	20	30	5	15	10	25	

add(40)

35	20	30	5	15	10	25	40
40	35	30	20	15	10	25	5

← poll()
40

35	20	30	5	15	10	25	

← poll()
35

30	20	25	5	15	10		

Name: ______________ PID: ______________ Code: 5812

```
void bubbleDown(int index) {
   if(index >= this.entries.size()) { return; }
   int leftIndex = left(index);
   if(leftIndex >= this.entries.size()) { return; }
   int largerChildIndex = leftIndex;
   int rightIndex = right(index);
   if(existsAndGreater(rightIndex, leftIndex)) {
      largerChildIndex = rightIndex;
   }
   if(existsAndGreater(largerChildIndex, index)) {
      swap(index, largerChildIndex);
      bubbleDown(largerChildIndex);
   }
}

void bubbleUp(int index) {
   if(index <= 0) { return; }
   Entry<K,V> e = this.entries.get(index);
   Entry<K,V> parent = this.entries.get(parent(index));
   int comp = this.comparator.compare(e.key, parent.key);
   if(comp > 0) {
      swap(index, parent(index));
      bubbleUp(parent(index));
   }
   else {
      return;
   }
}
```

What is the run-time for a Max Heap

add()

Worst Case $\Theta(\log_2(N))$

What conditions make up the worst case for add()?

sorted order for max heap

Best Case: $\Theta(1)$

What conditions make up the best case for add()?

heap → $\overset{\text{some have}}{\text{no}}$ duplicate keys

→ do have duplicate keys (most)

max heap → reverse sorted list

min heap → sorted list

poll()

Worst Case $\Theta(\log_2(N))$

What conditions make up the worst case for poll()?

small #s at the bottom

Best Case: $\Theta(1)$

What conditions make up the best case for poll()?

all duplicate keys