Max Heap → largest value comes out first

Assume the key and value are identical for this example

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

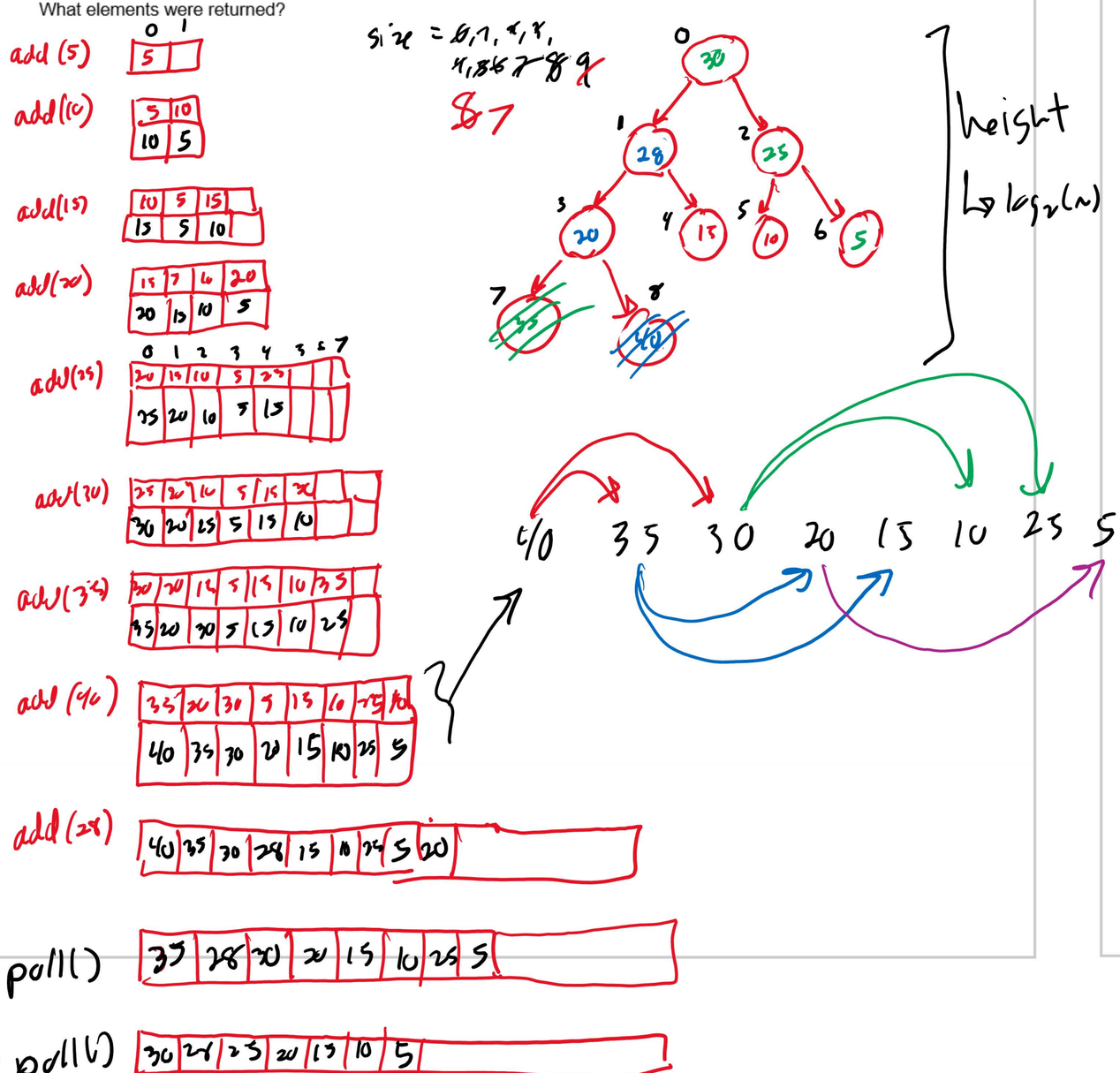
ArrayList<Integer> heap = new ArrayList<>(2); //initial capacity of 2

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

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

Call poll() twice → draw picture & array

What elements were returned?




```
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 $\mathcal{O}(\log_2(n))$

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

sorted

Best Case: $\mathcal{O}(1)$

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

added key already in heap order

poll()

Worst Case $\mathcal{O}(\log_2(n))$

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

Best Case: X

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