CSE12 - Lecture 21 - A00

Monday, November 14, 2022 8:00 AM

PA7 due tomorrow
PA4 Late / Resubmit
Friday → Exam 2
↳ From Exam 1 up to hash tables

Heap Applications

Median

heap — high to low → max heap — low to high → min heap

```
class Tracker {
  PriorityQueue<Integer> pq1 = new PriorityQueue<>(Collections.reverseOrder(Integer::compare));
  PriorityQueue<Integer> pq2 = new PriorityQueue<>(Integer::compare);
  void add(int n) {
    if(pq2.size() == 0 && pq1.size() == 0) {
      pq2.add(n);
      return;
    }
    int current = get();
    if(n >= current) {
      pq2.add(n);
    }
    else {
      pq1.add(n);
    }
    int sizeDifference = pq2.size() - pq1.size();
    if(sizeDifference > 1) { pq1.add(pq2.poll()); }
    else if(sizeDifference < -1) { pq2.add(pq1.poll()); }
  }

  int get() {
    if(pq2.size() == pq1.size()) { return (pq2.peek() + pq1.peek()) / 2; }
    if(pq2.size() > pq1.size()) { return pq2.peek(); }
    else { return pq1.peek(); }
  }

  public String toString() {
    return "" + pq1 + " " + this.get() + " " + pq2;
  }
}
```

N → # elements

O(1)

sorted input → BC: 1

height ↳ $\log_2(N)$ — add()

WC: $\log_2(N)$

½ [ ]

BC: $\log_2(N)$ WC: $2\log_2(N)$

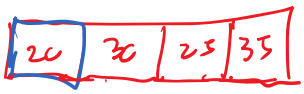
poll() @ ($\log_2(N)$)

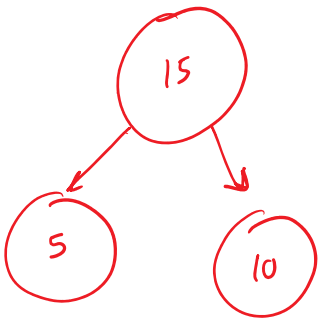
Draw the picture and the arrays for the following:

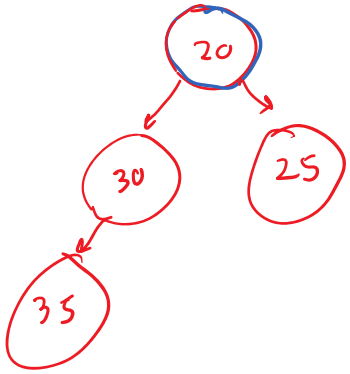
Add the following elements to the Tracker (in this order):
5, 10, 15, 20, 25, 30, 35

max heap pq1

min heap pq2







value of final get()? 20

What is the result of the call to get() after adding all the elements?

20

What is the run-time for the tracker?

	add()	set()
Worst Case	$\Theta(\log_2(N))$	$\Theta(1)$
Best Case:		

Write a method to use the tracker:

```
int findNumber ( Integer[] arr ) {
    MedianTracker tracker = new ____________ 1
    for ( int i=0; i< arr.length; i++ ) {
        tracker.add( arr[i] ); ] log2(N)  } N * log2(N)
    }
    return tracker.get();          1
}
```

What is the total run-time using the tracker:

$\Theta(N * \log_2(N))$

Using a PriorityQueue, write a Heap Sort method to perform an in-place sort of an array: