# CSE12 - Lecture 21

Wednesday, May 22, 2024 10:00 AM

PA8 released - due next Wednesday @ 8am
PA7 - hard deadline - Thursday @ 8am
PA5 Late/Resubmit - hard deadline - Thursday @ 8am
No class next Monday (5/27) - holiday

Lecture 21

## 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.peak() + pq1.peak()) / 2; }
    if(pq2.size() > pq1.size()) { return pq2.peak(); }
    else { return pq1.peak(); }
  }

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

BC: 1 (sorted data) WC: $\log_2(N)$

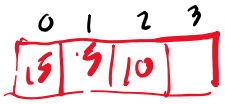
BC: 1 WC: $\log_2(N) + \log_2(N) \rightarrow \log_2(N)$

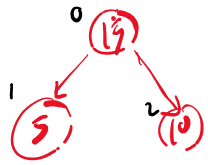
O(1)

Draw the picture and the arrays for the following:

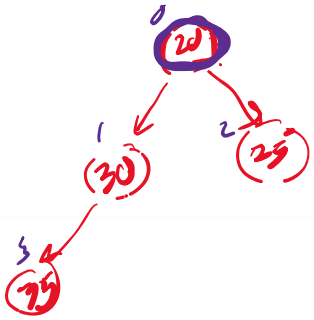
Add the following elements to the Tracker (in this order):
5, 10, 15, 20, 25, 30, 35

pq1 max heap




pq2 min heap


median → 20

Name: _____ PID: _____ Code: 7250

What is the result of the call to get() after adding all the elements? 20

What is the run-time for the tracker?

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

Best Case: $\Theta(1)$

Write a method to use the tracker: N

```
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:

$\Theta(N * \log_2$

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

See pre-lecture for Heap Sort implementation