CSE12 - Lecture 21 - C00

Monday, November 14, 2022 11:00 AM

PA7 due tomorrow

PA4 Late/Resubmit due tomorrow

Friday → Exam 2

↳ from Exam 1 up to hash tables

Heap Applications

Median

m → # of elements

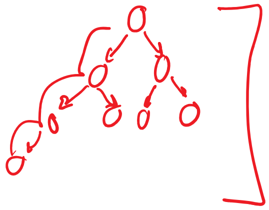
```
class Tracker {                                              Heap
  PriorityQueue<Integer> pq1 = new PriorityQueue<>(Collections.reverseOrder(Integer::compare));   high to low → max heap
  PriorityQueue<Integer> pq2 = new PriorityQueue<>(Integer::compare);
  void add(int n) {                                          low to high → min heap
    if(pq2.size() == 0 && pq1.size() == 0) { 1
      pq2.add(n);  ] O
      return;
    }
    int current = get(); O(1)
    if(n >= current) { 1
      pq2.add(n);
    }
    else {
      pq1.add(n);
    }
    int sizeDifference = pq2.size() - pq1.size(); 1
1/2 [ 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; } 3
    if(pq2.size() > pq1.size()) { return pq2.peek(); } 2
    else { return pq1.peek(); } 1
  }

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

WC $\log_2(m)$

BC 1 → sorted input

BC $\log_2(m)$ WC $2\log_2(m)$

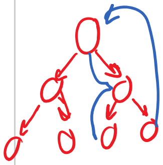
$O(1)$



height ↳ $\log_2(m)$

add() $\log_2(m)$

poll() $\log_2(m)$



Draw the picture and the arrays for the following:

Add the following elements to the Tracker (in this order):

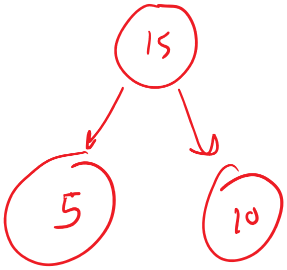
~~5~~, ~~10~~, ~~15~~, ~~20~~, ~~25~~, ~~30~~, ~~35~~

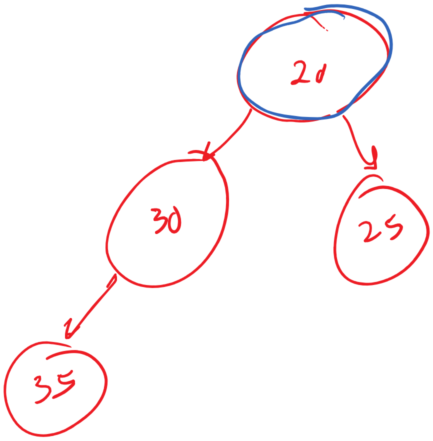
max heap pq1

15	5	10	

min heap pq2

20	30	25	35





Value of get()? $\boxed{20}$

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

20

What is the run-time for the tracker?

add() get()

Worst Case $\Theta(\log_2(N))$ $\Theta(1)$

Best Case:

Write a method to use the tracker:

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