Edwin Mak

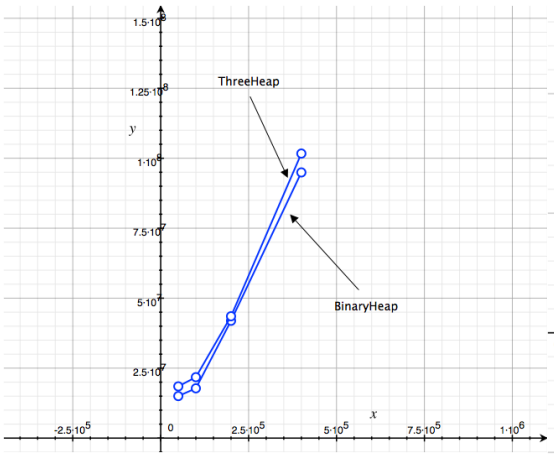
Homework #3 Write-Up

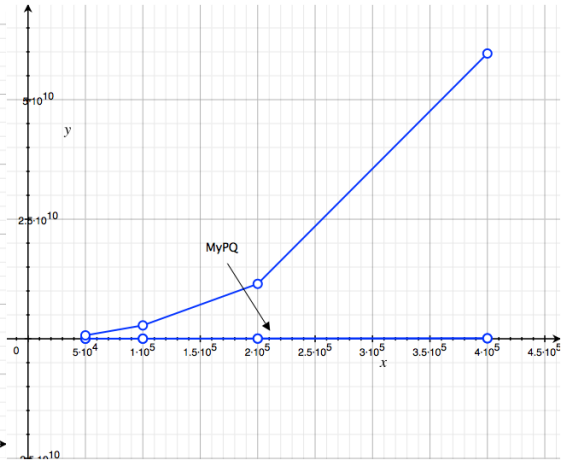
1)BinaryHeap: isEmpty O(1)
               size O(1)
               insert O(logN)
               findMin O(1)
               deleteMin O(logN)

ThreeHeap:
               isEmptyO(1)
               size O(1)
               insert O(logN)
               findMin O(1)
               deleteMin O(logN)

MyPQ(Unsorted Array):
               isEmpty O(1)
               size O(1)
               insert O(1)
               findMin O(N)
               deleteMin O(N)

2)





Binary Tree:
5e4 1.502e7
1e5 1.7791e7
2e5 4.1923e7
4e5 9.493e7

ThreeHeap:
5e4 1.8488e7
1e5 2.1776e7
2e5 4.3541e7
4e5 1.01669e8

```
MyPQ (Unsorted Array):
5e4   7.18582e8
1e5   2.79739e9
2e5   1.1484935e10
4e5   5.9650473e10
```

```
for(int timing = 0; timing < NUM_TIMINGS; ++timing) {
                long startTime = System.nanoTime();
                //start code
                PriorityQueue z = new MyPQ();
                Random random = new Random();
                for(int i = 0; i < 400000; i++){
                  double rand = random.nextDouble();
                   z.insert(rand);
                }
                for(int i = 0; i < 400000; i++){
                   z.deleteMin();
                }
                // end code
                long endTime = System.nanoTime();
                long elapsedTime = endTime - startTime;
                // 1 second = 1000000000 (10^9) nanoseconds.
                System.out.println(elapsedTime + "
nanoseconds or " + elapsedTime/(1000000000.0) + " seconds
elapsed");
            }
```

3) a)The Asymptopic analysis was pretty useful because the measured analysis what I expected.

b)The analysis matched what I expected in that the unsorted array with an O(N) insert/deleteMin had the (mostly) linear runtime and grew much faster compared to the ThreeHeap and the BinaryHeap, which both had insert/deleteMin operations of O(logN).

c)I would recommend someone to use the BinaryHeap over the other two options. The unsorted list PQ has an insert of O(1) and a delete of O(N), which makes insert/delete O(N). ThreeHeap on the other hand requires fewer comparisons to go through the entire heap, but can actually do more comparisons because it needs to compare three children when percolating compared to binary heaps that only require two comparisons when percolating. Also, the ThreeHeap goes through the heap faster than a BinaryHeap by a constant factor of $\log_3(2)$, which is a minimal difference.

I would not consider using an unsorted array because of the O(N) operation of deleteMin. Perhaps the only time consider using an unsorted array was if I knew that I would be inserting a lot of elements while not calling deleteMin much at all.

4) I tested my priority queue's insert() and deleteMin() methods by drawing trees and calling insert() and deleteMin() according to my trees. I wrote a simple test program by generating random doubles to insert into the priority queues. After calling insert() a number of times, I called deleteMin() until the pq was empty. I checked to make sure the deleteMin was working properly by putting the deleteMin calls into a new array and making sure that the array was in chronological order.

Below is some test code:

```
import java.util.Random;

public class TestCode{

   public static void main (String[] args){
      PriorityQueue z = new BinaryHeap();
      //z.deleteMin();
      z.insert(5);
      z.insert(100);
      z.insert(6);
      z.insert(5);
      z.insert(8);
      z.insert(1.5);
      z.insert(0.0);
      z.insert(5.);
      z.insert(100.0);
      z.insert(11);
      for(int i = 0; i < 10; i++){
         System.out.println("+"+z.findMin());
         System.out.println(z.size());
         System.out.println(z.deleteMin());
      }
      //z.makeEmpty();
      //System.out.println(z.isEmpty());
      System.out.println(z.size());
   }
       public static final int NUM_TIMINGS = 5;

       public static void main(String[] args) {
            PriorityQueue z = new MyPQ();
```

```
            z.insert(11);
            z.insert(3);
            z.insert(2);
            z.insert(3);
            z.insert(4);
            z.insert(7);
            z.insert(9);
            for(int i = 0; i < 100000; i++){
               Random random = new Random();
               double rand = random.nextDouble();
               z.insert(rand);
               //System.out.println(z.deleteMin());
            }
            double [] temp = new double[z.size()];
            for(int i = 0; i < z.size() ; i++){
               temp[i] = z.deleteMin();
            }
            int count = 0;
            for(int i = 0; i< z.size(); i++){
               if(temp[i] > temp[i+1]){
                  count = 2;
               }
            }
          System.out.println(count);

          }
}
```

5)

a)

BinaryHeap				i*2	i*2+1
ThreeHeap			i*3-1	i*3	i*3+1
FourHeap		i*4-2	i*4-1	i*4	i*4+1
FiveHeap	i*5-3	i*5-2	i*4-1	i*5	i*5+1

b)

The mathematical formula to find the leftmost child of a d-heap is:

(d*i)*d-2