Daniel Chai

CSE 373

HW 2 – Write-Up

January 27th, 2017

1. I wrote methods in MyClient that uses methods inside ThreeHeap to test the functions.
2. **BuildHeap** – O(n) - the first for-loop traverses through all of the list, which means it is at least O(n).

For the second for-loop, we only iterate through (n + 1) / 3 elements, which is like adding n/3 to the run time.

**isEmpty** – O(1), it is a constant runtime since we are keeping track of size

**size** – O(1), we are keeping track of size

**deleteMin, insert** – Percolating up will take as many steps as the height of the tree, which is ceiling() – 1. So the runtime will be O(()

**findMin** – O(1), just returning the first element

1. A) **O(log n)** – This is log n, because of the worst case scenario where n will be very large. The second for-loop will then have to iterate log n times to increase the sum count by 1.

B) **O(n3)** – The first loop tells us that it is at least O(n). The second for-loop happens (n2+ n) / 3 times which can be written as O(n2). So, the worst case would be n3.

C) **O(n2)** – This method is like having nested for-loops. Because you are iterating over n, n times so the total runtime is n2.

1. A) int difference //largest difference

for (int i, every number up to size of the array) {

for(int j, every number up to i){

int temp = i – j

set difference to the bigger of temp and difference

}

}

Return difference;

B) int min

Int max

for (every number up to size of the array){

set min to smaller of min and array[i]

set max to bigger of max and array[i]

}

Return max – min ;

C) In order to make this O(1), we must keep track of min and max while we are adding the elements into the array.

1. a) **BuildHeap** – O(n2) – the runtime of the new algorithm is O(n). This is because of lack of amortized cost since the algorithm is not doing enough cheap processes to “save up”. So, since I have to run the algorithm n times, it will be O(n2).

**isEmpty** – new algorithm does not affect this

**size** – new algorithm does not affect this

**deleteMin** – new algorithm does not affect this

**insert** – O(n) – The total runtime becomes N + log3N. So, that becomes O(n).

**findMin** – new algorithm does not affect this

b) **BuildHeap** – It will be worse than original one but better than previous algorithm. This one copies over arrays more often than the original method of increasing the size by 2 times.

**insert** – Similarly, this method also copies over more than the original algorithm.

c) **BuildHeap** – Same runtime as before.

**insert** – The runtime is now same as before.

1. I implemented the Floyd’s method of building a heap.