CS 61B Discussion 8 EP Solutions Spring 2017

1 Warmup

Given the following method on a sorted array, what is the worst-case runtime?.

```java
public static int f1(int i, int[] numList) {
    for (int j = 0; j < numList.length; j++) {
        if (numList[j] == i) {
            return j;
        }
    }
    return -1;
}
```

Rewrite this method to improve the runtime. What is the runtime of your new method?

Linear time, $\theta(n)$ where n is the length of the list. Binary search on the sorted array, **for** $\theta(logn)$

2 You wanna hang out this Spring '15? Asymptotes!

For each of the pieces of code below, give the runtime in θ notation as a function of the given parameters. Let $f(x)$ be a function that runs in time linear to the size of its input x.

```java
public static void f1(int n) {
    if (n == 0) {return;}
    f1(n/2);
    f(n);
    f1(n/2);
}
```

$\theta(nlogn)$

```java
public static void f2(int n) {
    if (n == 0) {return;}
    f2(n-1);
    f(17);
    f2(n-1);
}
```

$\theta(2^n)$

```java
public static void f3(int n, int m) {
    if (m <= 0) {
        return;
    } else {
        for (int i = 0; i < n; i +=1) {
            f3(n, m - 1);
        }
    }
}
```

$\theta(n^m)$

3 It's Fall '16 And I'm Still Doing Asymptotics

In the following, assume that cnst() is a constant time function call. When giving bounds, give bounds that are as tight and simple as possible.

1. Give best- and worst-case runtime bounds for the call foo2(N,N) as a function of N

```
public static void foo2(int i, int N) {f
    if (i==0) {return;}
    for (int j = 0; j < i; j = j+1) {
        cnst();
    }
    if (i > N/2) {
        foo2(i-1, N);
    } else {
        foo2(i/2, N) + foo2(i/2, N);
    }
}
```

Best **case:** θ(N^2) Worst **case:** θ(N^2)

2. True or false: if $f(N) \in O(N)$ and $g(N) \in O(N^2)$, and both functions are non-negative, then $|g(N) - f(N)| \in \Omega(N)$. if true, explain why; otherwise, give a counterexample.
False: Consider $f(N) = g(N) = N$

3. True or false: if $f(N) \in \theta(N)$ and $g(N) \in \theta(N^2)$, and both functions are non-negative, then $|g(N) - f(N)| \in \Omega(N)$. If true, explain why; otherwise give a counterexample.
True: Since $g(N) \in \theta(N^2)$, it is also in $\Omega(N)$, **while** $f(N) \in O(N)$. Since g grows at least as N^2, and O(N) quantity is eventually negligible in comparison.

4. What is the running time of the following algorithm, as a function of the parameter r?

```
/** Assumes that VALS is a square array, and that 0 <= R, C <
    vals.length. */
double best(double vals[][], int r, int c) {
    if (r == 0) {
        return vals[r][c];
    }
    double v = best(vals, r-1, c);
    if (c > 0) {
        v = Math.max(v, best(vals, r-1, c-1));
    }
    if (c < vals[r].length - 1) {
        v = Math.max(v, best(vals, r-1, c+1));
    }
    return v + vals[r][c];
}
```

$\theta(3^r)$

4 Fall '16: I'm more than just a runtime

1. Your friend, a budding politician, meets several hundred people a day and places their names onto the front of an ArrayList. Once there, he never removes a name, but he sometimes looks through the list to see the order in which he met people. At least one aspect of his procedures is slower than it could be. Describe and justify a small change in your friend's use of the data structure that would improve runtime without changing the data structure involved.

   Putting items at the end of the list instead of the front would speed up addition to the list.

   Now describe and justify a change in the data structure that would improve runtime without requiring a change in actions taken.

   Using a linked list would allow adding to the front of the list in $O(1)$ time.

5 (Most 61B Problems) $\in$ O(These Problems)

1. Give a tight θ bound on the running time.

```
public int f1(n):
    if (n == 1){return 0;}
    if (n is even){
        return f1(n/2);
    } else {
        return f1(n+1);
    }
```

   Answer: θ(log(n))
   Make an O() and Ω() bound by considering the odd and even cases. Use **this** information to establish a θ bound

2. Give a tight θ bound on the running time, where process() is a method that runs in $\theta(nlogn)$

```
function f2(n):
    if (n = 1){return 1;}
    int a = f2(n/2);
    int b = f2(n/2);
    x = process(a, b)
    return x;
```

   Answer: $\theta(nlog^2n)$
   The recursion tree should give you a summation that looks like **this:**
   $n\log n + n\log(\frac{n}{2}) + \log(\frac{n}{4}) + \dots$
   $= n(\log n + \log(\frac{n}{2}) + \log(\frac{n}{4}) + \dots)$
   $= n(\log n + (\log n - \log 2) + (\log n - \log 4) + \dots)$
   $= n((\log n + \log n + \dots + \log n) - (\log 2 + \log 4 + \dots + \log n))$
   $= n((\log n)^2 - (1 + 2 + \dots + \log n))$
   $\approx n(\log^2 n - \frac{\log^2 n}{2})$
   $= \frac{n\log^2 n}{2}$
   $= \Theta(n\log^2 n)$

3. True or False: If $f(n) = O(g(n))$, then $2^{f(n)} = O(2^{g(n)})$

   False; Consider $f(n) = n$ and $g(n) = n/2$, then use what you know about exponents to mess around with the base.