CS 61B Spring 2018

Asymptotics II

Exam Prep 8: March 5, 2018

1 Warmup

Given the following method on a sorted array, what is the worst-case runtime? There is an approach to make this algorithm faster. What is that approach and what is the worst-case runtime of the faster algorithm?

```
1  public static int f1(int i, int[] numList) {
2      for (int j = 0; j < numList.length; j++) {
3          if (numList[j] == i) {
4              return j;
5          }
6      }
7      return -1;
8  }
```

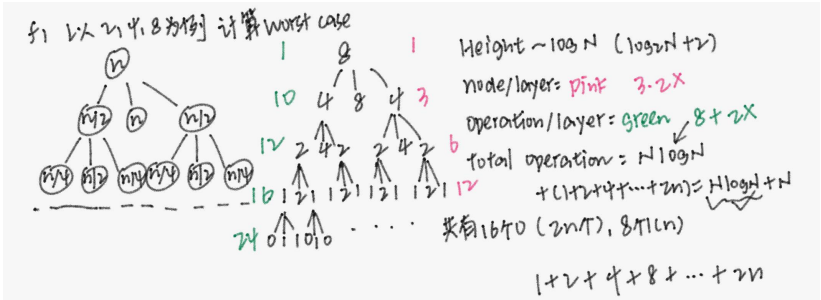
Worst case: O(N)
Approach: using a low index and a high index; compare i to the middle value, and change low/high accordingly
Worst case: O(log N)

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.

Big Theta(n log n)

```
1  public static void f1(int n) {
2      if (n == 0) {return;}
3      f1(n/2);
4      f(n);
5      f1(n/2);
6  }
```

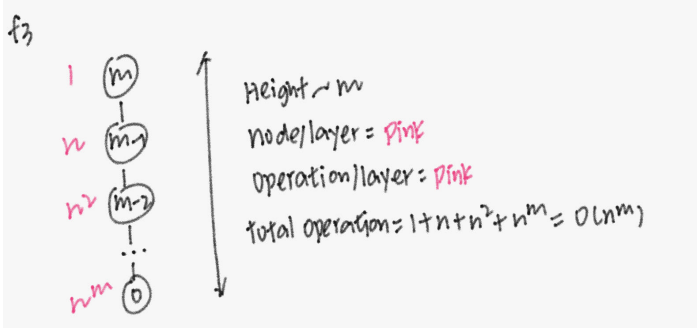


Big Theta(2^n)

```
1  public static void f2(int n) {
2      if (n == 0) {return;}
3      f2(n-1);
4      f(17);
5      f2(n-1);
6  }
```

Big Theta(n^m)

```
1  public static void f3(int n, int m) {
2      if (m <= 0) {
3          return;
4      } else {
5          for (int i = 0; i < n; i +=1) {
6              f3(n, m - 1);
7          }
8      }
9  }
```

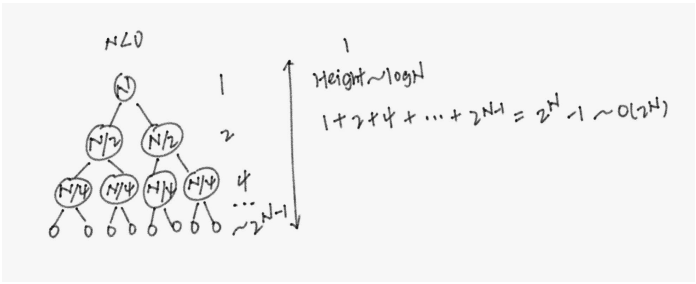


3 It's Fall '16 And I'm Still Doing Asymptotics

1. Give best- and worst-case runtime bounds for the call foo2(N,N) as a function of N. Assume that cnst() is some function that runs in constant time.

```
1  public static void foo2(int i, int N) {
2      if (i==0) {return;}
3      for (int j = 0; j < i; j = j+1) {
4          cnst();
5      }
6      if (i > N/2) {
7          foo2(i-1, N);
8      } else {
9          foo2(i/2, N) + foo2(i/2, N);
10     }
11 }
```

Best: Big Theta(N^2)
Worst: Big Theta(2^N)



Best case: $\Theta($ $)$ Worst case: $\Theta($ $)$

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: both g and f are constant time

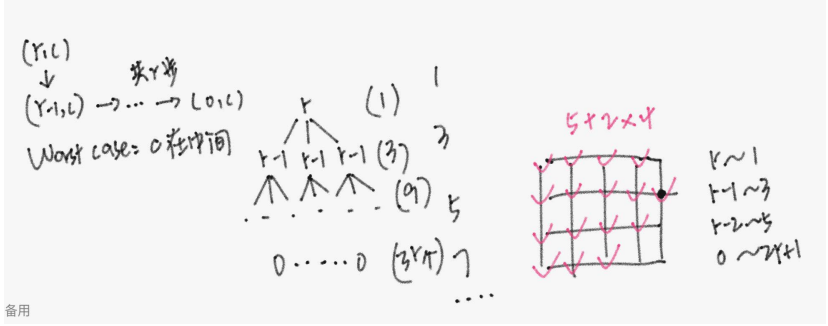
3. True or false: if $f(N) \in \Theta(N)$ and $g(N) \in \Theta(N^2)$, and both functions are non-negative, then $|g(N) - f(N)| \in \Omega(N)$. If true, explain why; otherwise give a counterexample.

True

4. What is a tight big-O bound for the worst case running time of the following algorithm, as a function of the parameter r?

```
1  /** Assumes that VALS is a square array, and that 0 <= R, C < vals.length. */
2  double best(double vals[][], int r, int c) {
3      if (r == 0) {
4          return vals[r][c];
5      }
6      double v = best(vals, r-1, c);
7      if (c > 0) {
8          v = Math.max(v, best(vals, r-1, c-1));
9      }
10     if (c < vals[r].length - 1) {
11         v = Math.max(v, best(vals, r-1, c+1));
12     }
13     return v + vals[r][c];
14 }
```

Worst case: $O($ 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.

   I think it's the "addFront" operation? But I'm not sure

   great intuition; adding new elements to the front means shifting all indexes by one; adding to the back will be much simpler

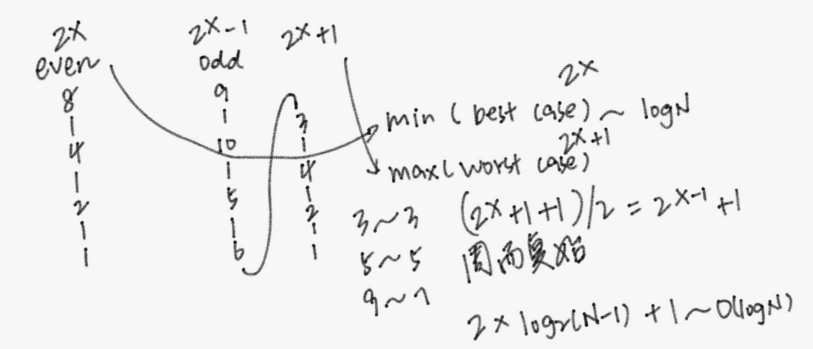
   Now describe and justify a change in the data structure that would improve runtime without requiring a change in actions taken.

   the resizing operation takes much time; so he can use LinkedList

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

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

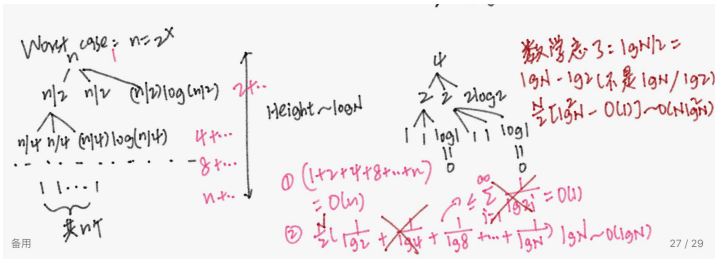
```
1  public int f1(n):
2      if (n == 1){return 0;}
3      if (n is even){
4          return f3(n/2);
5      } else {
6          return f3(n+1);
7      }
```



   Answer: $\Theta($log N$)$

2. Give a tight Θ bound on the running time, where process() is a method that runs in $\Theta(n \log n)$

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



   Answer: $\Theta($ N $)$

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

   this is false; which I don't quite get is that , 2^n cannot be marked as O(2^(n/2))