More Algorithmic Analysis

Discussion 08

Announcement

- Congratulations on surviving Engima!
- Weekly Survey due Tuesday 03/08
- Homework 5 due Tuesday 03/08
- Lab 8 due Friday 03/11

Review

Best Case vs. Worst Case

Best Case: Restrict examined situation to only the best case (independent of input size)
Worst Case: Restrict examined situation to only the worst case (independent of input size)

Best case, worst case, and average case can ALL be bounded by Theta, O, or Omega.

CS 61B Spring 2022 — Recursive Algorithmic Analysis

Exam Prep Discussion 8: March 7, 2022

1 Asymptotics is Fun!

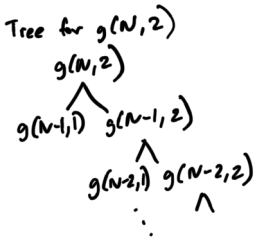
(a) Using the function g defined below, what is the runtime of the following function calls? Write each answer in terms of N.

```
1  void g(int N, int x) {
2      if (N == 0) {
3          return;
4      }
5      for (int i = 1; i <= x; i++) {
6          g(N - 1, i);
7      }
8  }
```

g(N, 1): $\Theta(N)$ — $g(N), g(N-1), \ldots, g(0)$ is $\Theta(N)$

g(N, 2): $\Theta(N^2)$ — $g(N-1,1) + g(N-2,1) + \ldots + g(0,1) \Rightarrow \Theta(N^2)$

Tree for g(N, 2):



(b) Suppose we change line 6 to g(N - 1, x) and change the stopping condition in the for loop to i <= f(x) where f returns a random number between 1 and x, inclusive. For the following function calls, find the tightest Ω and big O bounds.

```
1  void g(int N, int x) {
2      if (N == 0) {
3          return;
4      }
5      for (int i = 1; i <= f(x); i++) {
6          g(N - 1, x);
7      }
8  }
```

Best Case: f(x) = 1

g(N) → g(N-1) → g(N-2) → ⋮ → g(0)

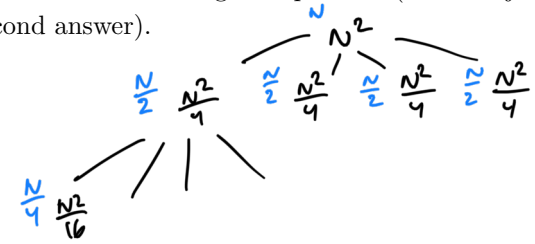
g(N, 2): $\Omega(N)$, $O(2^N)$ ← Apply previous problem to find the 1 and 2 cases

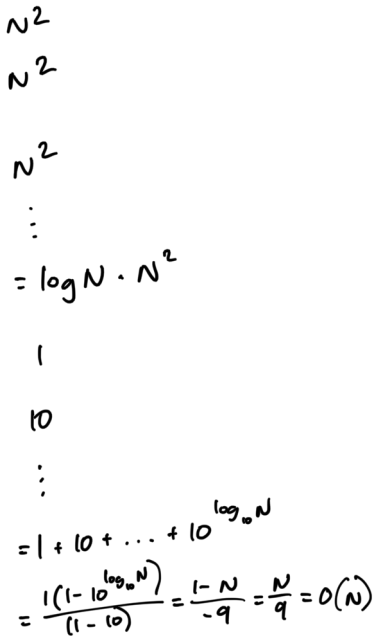
g(N, N): $\Omega(N)$, $O(N^N)$

2 Slightly Harder (Spring 2017, MT2)

Give the runtime of the following functions in Θ or O notation as requested. Your answer should be as simple as possible with no unnecessary leading constatns or lower order terms. For f5, your bound should be as tight as possible (so don't just put $O(N^{NM!})$ or similar for the second answer).

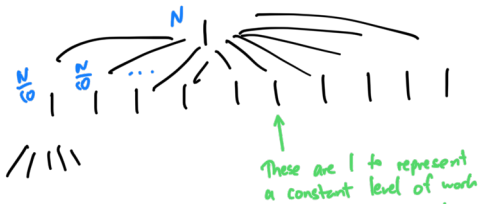
```
1  public static void f4(int N) {
2      if (N == 0) {return;}
3      f4(N / 2);
4      f4(N / 2);
5      f4(N / 2);
6      f4(N / 2);
7      g(N); // runs in Θ(N^2) time
8  }
```





Runtime: $\Theta(N^2 \log N)$

```
1  public static void f5(int N, int M) {
2      if (N < 10) {return;}
3      for (int i = 0; i <= N % 10; i++) {
4          f5(N / 10, M / 10);
5          System.out.println(M);
6      }
7  }
```



Runtime: $O($ N $)$ Worst case is N is a bunch of 9's

3 Flip Flop

Suppose we have the `flip` function as defined below. Assume the method `unknown` returns a random `integer` between 1 and `N`, exclusive, and runs in constant time. For each definition of the `flop` method below, give the best and worst case runtime of `flip` in $\Theta(.)$ notation as a function of `N`.

```
1  public static void flip(int N) {
2      if (N <= 100) {
3          return;
4      }
5      int stop = unknown(N);
6      for (int i = 1; i < N; i++) {
7          if (i == stop) {
8              flop(i, N);
9              return;
10         }
11     }
12 }
```

(a)
```
public static void flop(int i, int N) {
    flip(N - i);
}
```

$\frac{N}{stop}$ calls, each doing stop work for a total of N

N, N - stop, N - 2stop, ...

Best Case: $\Theta($ N $)$, Worst Case: $\Theta($ N $)$

(b)
```
public static void flop(int i, int N) {
    int minimum = Math.min(i, N - i);
    flip(minimum);
    flip(minimum);
}
```

Use this as the hint for the stop values in the best and worst cases

If stop = 1

If stop = $\frac{N}{2}$

Best Case: $\Theta($ 1 $)$, Worst Case: $\Theta(N \log N)$

(c)
```
public static void flop(int i, int N) {
    flip(i);
    flip(N - i);
}
```

If stop = 1

If stop = N - 1

Best Case: $\Theta($ N $)$, Worst Case: $\Theta($ N^2 $)$