CS 61B
Fall 2020

More Asymptotics

Exam Prep Discussion 7: October 5, 2020

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): $\Theta(\quad)$
g(N, 2): $\Theta(\quad)$

(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(x) 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 }
```

g(N, 2): $\Omega(\quad)$, $O(\quad)$
g(N, N): $\Omega(\quad)$, $O(\quad)$

2 Flip Flop

For each part, give the best and worst case runtime in $\Theta(.)$ notation as a function of N. Your answer should be simple with no unnecessary leading constants or summations.

```java
public static void flip(int N) {
    if (N <= 100) {
        return;
    }
    for (int i = 1; i < N; i++) {
        // Assume g(i, N) will be equal to i for at least one i
        if (g(i, N) == i) {
            flop(i, N);
            return;
        }
    }
}
```

Given the method `flip` defined above, we will determine the best and worst case runtime when `flop` is defined as:

```java
public static void flop(int a, int b) {
    flip(b - a);
}
```

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

```java
public static void flop(int a, int b) {
    int low = Math.min(a, b - a);
    flip(low);
    flip(low);
}
```

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

```java
public static void flop(int a, int b) {
    flip(a);
    flip(b - a);
}
```

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

3 Prime Factors

What is the best and worst case runtime of the function below?

```
int prime_factors(int N) {
    int factor = 2;
    int count = 0;
    while (factor * factor <= N) {
        while (N % factor == 0) {
            System.out.println(factor);
            count += 1;
            N = N / factor;
        }
        factor += 1;
    }
    return count;
}
```

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