CS 61B Spring 2022

Bits & Algorithmic Analysis

Exam Prep Discussion 7: February 28, 2022

1 Asymptotics Introduction

Give the runtime of the following functions in Θ notation. Your answer should be as simple as possible with no unnecessary leading constants or lower order terms.

```
private void f1(int N) {
    for (int i = 1; i < N; i++) {
        for (int j = 1; j < i; j++) {
            System.out.println("hello tony");
        }
    }
}
```

$\Theta($___$)$

```
private void f2(int N) {
    for (int i = 1; i < N; i *= 2) {
        for (int j = 1; j < i; j++) {
            System.out.println("hello hannah");
        }
    }
}
```

$\Theta($___$)$

2 Finish the Runtimes

Below we see the standard nested for loop, but with missing pieces!

```
for (int i = 1; i < ______; i = ______) {
    for (int j = 1; j < ______; j = ______) {
        System.out.println("We will miss you next semester Akshit :(");
    }
}
```

For each part below, **some** of the blanks will be filled in, and a desired runtime will be given. Fill in the remaining blanks to achieve the desired runtime! There may be more than one correct answer.

Hint: You may find `Math.pow` helpful.

(a) Desired runtime: $\Theta(N^2)$

```
for (int i = 1; i < N; i = i + 1) {
    for (int j = 1; j < i; j = ______) {
        System.out.println("This is one is low key hard");
    }
}
```

(b) Desired runtime: $\Theta(log(N))$

```
for (int i = 1; i < N; i = i * 2) {
    for (int j = 1; j < ______; j = j * 2) {
        System.out.println("This is one is mid key hard");
    }
}
```

(c) Desired runtime: $\Theta(2^N)$

```
for (int i = 1; i < N; i = i + 1) {
    for (int j = 1; j < ______; j = j + 1) {
        System.out.println("This is one is high key hard");
    }
}
```

(d) Desired runtime: $\Theta(N^3)$

```
for (int i = 1; i < ______; i = i * 2) {
    for (int j = 1; j < N * N; j = ______) {
        System.out.println("yikes");
    }
}
```

3 Asymptotic Expressions

(a) Which of the following expressions are true? Check all that apply. Equations between asymptotic expressions, such as $O(f) = O(g)$ simply mean that all functions that are $O(f)$ are also $O(g)$ and vice-versa. An expression such as $O(f) \subseteq O(g)$ means that all functions that are $O(f)$ are also $O(g)$.

- ☐ $\Theta(1000 * N^3 + N * \log(N)) = \Theta(N^3))$.
- ☐ For all $k \geq 0$, $O(N^k) \subseteq O(N^{k+1}))$.
- ☐ For all $k \geq 0$, $\Omega(N^k) \subseteq \Omega(N^{k+1}))$.
- ☐ For positive-valued functions f and g, if $f = \Omega(g)$ and $g = O(h)$, $f = \Omega(h)$.
- ☐ For positive-valued functions f and g, if $f = \Omega(g)$ and $h = O(g)$, $f = \Omega(h)$.

(b) For positive-valued functions $f_0 \ldots f_k$, where we define $f_i(n) = 1 + f_{n\%i}(n)$ for $i \geq 1$ and $f_0(n) = 1$, which of the following are true? Check all that apply. Assume that $n > k$.

- ☐ The evaluation of $f_k(n)$ may run forever.
- ☐ $f_k(n) = \Omega(log(k))$, with respect to k.
- ☐ $f_k(n) = O(k)$, with respect to k.
- ☐ $f_k(n) = \Theta(1)$, with respect to n.
- ☐ If $n = k! - 1$, $f_k(n) = \Theta(k)$, with respect to k.

4 Prime Factors

Determine the best and worst case runtime of `prime_factors` in $\Theta(.)$ notation as a function of `N`.

```
1  int prime_factors(int N) {
2      int factor = 2;
3      int count = 0;
4      while (factor * factor <= N) {
5          while (N % factor == 0) {
6              System.out.println(factor);
7              count += 1;
8              N = N / factor;
9          }
10         factor += 1;
11     }
12     return count;
13 }
```

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