

## 1 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("Circle is the best TA");  
    }  
}
```

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 = j+1) {  
        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 < 2_____; j = j * 2) {  
        System.out.println("This is one is mid key hard");  
    }  
}
```

- (c) Desired runtime:  $\Theta(2^N)$ .  $\frac{2^N}{N}$  is a valid answer, could you think of another?

```
for (int i = 1; i < N; i = i + 1) {  
    for (int j = 1; j < 2^n-1/n; j = j + 1) {  
        System.out.println("This is one is high key hard");  
    }  
}
```

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

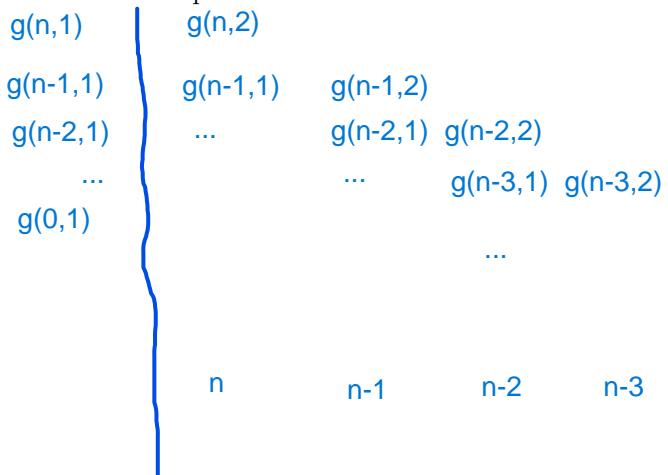
format problem : `Math.pow(2, N)`

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

## 2 Asymptotics is Fun!

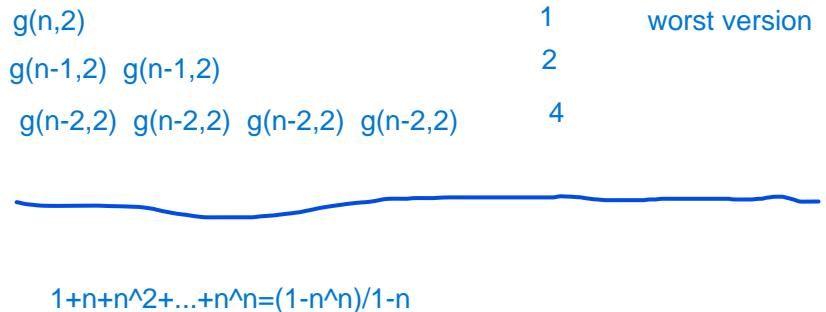
- (a) Using the function `g` defined below, what is the runtime of the following function calls? Write each answer in terms of  $N$ . Feel free to draw out the recursion tree if it helps.

```
void g(int N, int x) {
    if (N == 0) {
        return;
    }
    for (int i = 1; i <= x; i++) {
        g(N - 1, i);
    }
}
g(N, 1): Θ(n)
g(N, 2): Θ(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  $\Omega$  and big O bounds. Feel free to draw out the recursion tree if it helps.

```
void g(int N, int x) {
    if (N == 0) {
        return;
    }
    for (int i = 1; i <= f(x); i++) {
        g(N - 1, x);
    }
}
g(N, 2): Ω(n), O(2^n)
g(N, N): Ω(n), O(n^n)
```



### 3 Asymptotics Proofs already done

As a reminder, the formal definitions of  $\Omega$ ,  $\Theta$ , and  $O$  are provided below:

Let  $f, g$  be real-valued functions. Then:

$f(x) \in \Theta(g(x))$  if there exists  $a, b, N_0 > 0$  such that for all  $N > N_0$ ,  $|ag(N)| \leq |f(N)| \leq |bg(N)|$ .

$f(x) \in O(g(x))$  if there exists  $b, N_0 > 0$  such that for all  $N > N_0$ ,  $|f(N)| \leq |bg(N)|$ .

$f(x) \in \Omega(g(x))$  if there exists  $a, N_0 > 0$  such that for all  $N > N_0$ ,  $|ag(N)| \leq |f(N)|$ .

Informally, we say that  $f(x) \in O(g(x))$  approximately means that  $f(x) \leq g(x)$ , and similarly,  $f(x) \in \Theta(g(x))$  means  $f(x) = g(x)$  and  $f(x) \in \Omega(g(x))$  means  $f(x) \geq g(x)$ . This problem will explore why we can make this informal statement, by showing that the  $O$  relation shares many properties with the  $\leq$  relation.

For this problem, let  $f$ ,  $g$ , and  $h$  be real-valued functions, and let  $x$ ,  $y$ , and  $z$  be real numbers. You won't be expected to write full proofs on exams, but this thinking style will be helpful on exams and especially in later classes.

(a) If  $x \leq y$ , then  $y \geq x$ . Show that if  $f(x) \in O(g(x))$ , then  $g(x) \in \Omega(f(x))$

(b) If  $x \leq y$  and  $y \leq x$ , then  $x = y$ . Show that if  $f(x) \in O(g(x))$  and  $g(x) \in O(f(x))$ , then  $f(x) \in \Theta(g(x))$

(c) For any real number,  $x \leq x$ . Show that for any function,  $f(x) \in O(f(x))$ .

(d) If  $x \leq y$  and  $y \leq z$ , then  $x \leq z$ . Show that if  $f(x) \in O(g(x))$  and  $g(x) \in O(h(x))$ , then  $f(x) \in O(h(x))$

(e) For any pair of real numbers  $x$  and  $y$ , either  $x < y$ ,  $x = y$ , or  $x > y$ . Show that this is NOT a property of  $O$ ; that is, find functions  $f$  and  $g$  such that  $f(x) \notin O(g(x))$  and  $g(x) \notin O(f(x))$ .