

1 Finish the Runtimes

Below we see some standard nested for loops, but with missing pieces!

For each part, **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");  
    }  
}
```

2 Disjoint Sets

For each of the arrays below, write whether this could be the array representation of a weighted quick union with path compression and explain your reasoning. **Break ties by choosing the smaller integer to be the root.**

(a)

1	2	3	0	1	1	1	4	4	5
---	---	---	---	---	---	---	---	---	---

(b)

9	0	0	0	0	9	9	9	-10
---	---	---	---	---	---	---	---	-----

(c)

1	2	3	4	5	6	7	8	9	-10
---	---	---	---	---	---	---	---	---	-----

(d)

-10	0	0	0	0	1	1	1	6	2
-----	---	---	---	---	---	---	---	---	---

(e)

-10	0	0	0	0	1	1	1	6	8
-----	---	---	---	---	---	---	---	---	---

(f)

-7	0	0	1	1	3	3	-3	7	7
----	---	---	---	---	---	---	----	---	---

3 This is NOT an Interview!

Given an `int x` and a *sorted* array `A` of N distinct integers, design an algorithm to find if there exists indices `i` and `j` such that `A[i] + A[j] == x`.

Let's start with the naive solution.

```
public static boolean findSum(int[] A, int x) {
    for (int i = 0; i < A.length; i++){
        for (int j = 0; j < A.length; j++) {
            if (A[i] + A[j] == x) return true;
        }
    }
    return false;
}
```

(a) How can we improve this solution? *Hint*: Does order matter here?

(b) What is the runtime of both the original and improved algorithm?