CS 61B
Spring 2026

Asymptotics I, Disjoint Sets

Exam-Level Discussion 06: February 23, 2026

1 xStep

Analyze the following loops and determine the asymptotic runtime of each using big Theta notation with respect to `N`. Assume that `System.out.println(...)` runs in constant time.

```
int x = 7;
while (x < N + 14) {
  System.out.println("tidal wave");
  x++;
}
```

Runtime: $\Theta(N)$

```
for (int x = 1; x < N; x++) {
  for (int y = 1; y < N; y++) {
    System.out.println("amethyst");
  }
}
```

Runtime: $\Theta(N^2)$

```
for (int x = 1; x < N; x *= 2) {
  for (int y = 1; y < N; y++) {
    System.out.println("flamewall");
  }
}
```

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

```
for (int x = 2; x < N; x += 2) {
  for (int y = 1; y < 1000000; y) {
    System.out.println("anathema");
  }
}
```

Runtime: $\Theta(N)$

```
for (int x = 3; x < N * N * N; x *= 3) {
  System.out.println("nullscapes");
}
```

Runtime: $\Theta(\log N^3)$

```
for (int x = 6; x < N; x += 6) {
  for (int y = x; y < N; y += 6) {
    System.out.println("grief");
  }
}
```

Runtime: $\Theta(N^2)$

2 I am Speed

For each code block below, fill in the blank(s) so that the function has the desired runtime. Do not use any commas. If the answer is impossible, just write "impossible" in the blank. Assume that `System.out.println` runs in constant time. You may use Java's `Math.pow(x, y)` to raise `x` to the power of `y`.

(a) Desired Runtime: $\Theta(N)$

```
public static void f1(int N) {

    for (int i = 1; i < N; i += 1){

        System.out.println("hi Dom");
    }
}
```

Note the solution could be `i += C`, where `C` is some constant independent of `N`. This is because even if we did for example, `i += 10`, we would do $\frac{N}{10}$ work in total, which is still $\Theta(N)$.

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

```
public static void f2(int N) {

    for (int i = 1; i < N; i *= C) {

        System.out.println("howdy Ergun");
    }
}
```

Here, the solution could be `i *= C`, where `C` is some constant independent of `N`. This is because even if we did for example, `i *= 5`, we would do $\log_5 N$ work in total, and in general $\log_i N$ work, which is still $O(\log n)$.

(c) Desired Runtime: $\Theta(1)$

```
public static void f3(int N) {

    for (int i = 1; i < 1000; i += 1) {

        System.out.println("hello Anniyat");
    }
}
```

Again, the solution is actually just `i < C`, where `C` is some constant independent of the input `N`. Any other syntactically valid constant would work.

(d) Desired Runtime: $\Theta(2^N)$

This one is tricky! *Hint: think about the dominating term in* $1 + 2 + 4 + 8 + ... + f(N)$

```
public static void f4(int N) {

    for (int i = 1; i < Math.pow(2, N); i *= 2) {

        for (int j = 0; j < i; j += 1) {
            System.out.println("what's up Alyssa");
        }
    }
}
```

3 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.**

There are three criteria here that invalidate a representation:

- If there is a cycle in the parent-link.
- For each parent-child link, the tree rooted at the parent is smaller than the tree rooted at the child before the link (you would have merged the other way around).
- The height of the tree is greater than $\log_2 n$, where n is the number of elements.

(a)

1	2	3	0	1	1	1	4	4	5

Impossible: has a cycle 0-1, 1-2, 2-3, and 3-0 in the parent-link representation.

(b)

9	0	0	0	0	9	9	9	−10

Impossible: the nodes 1, 2, 3, 4, and 5 must link to 0 when 0 is a root; hence, 0 would not link to 9 because 0 is the root of the larger tree.

(c)

1	2	3	4	5	6	7	8	9	−10

Impossible: tree rooted at 9 has height $9 > \log_2 10$.

(d)

−10	0	0	0	0	1	1	1	6	2

Possible: 8-6, 7-1, 6-1, 5-1, 9-2, 3-0, 4-0, 2-0, 1-0.

(e)

−10	0	0	0	0	1	1	1	6	8

Impossible: tree rooted at 0 has height $4 > \log_2 10$.

(f)

−7	0	0	1	1	3	3	−3	7	7

Impossible: tree rooted at 0 has height $3 > \log_2 7$.