CS 61BL
Summer 2026

Asymptotics and Trees

Exam Prep 01: July 10th, 2026

1 The Space-Time Continuum

For each growth function on the left, Theta bound it, simplify it, and draw a line to the matching Theta bound on the right.

Recall the rules for simplifying asymptotic bounds and logarithm rules!

$\log(N) + N$

$N^2 + 3N\log(N)$

$\log(2^N)$

$2^N * 2^{-100}$

$\log(9)$

2^{N+4}

$2 + \frac{1}{N}$

$\log(N^2)$

$5N^2$

$6\log(N)$

$\Theta(1)$

$\Theta(\log N)$

$\Theta(N)$

$\Theta(N\log N)$

$\Theta(N^2)$

$\Theta(2^N)$

2 Asymptotics

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.

In CS 61B, we always assume ***IO.println*** *runs in* $\Theta(1)$

```
private void f1(int N) {
  for (int i = 1; i < N; i++) {
    for (int j = 1; j < i; j++) {
        IO.println("dwang 1.0");
    }
  }
}
```

Runtime: _____

```
private void f2(int N) {
  for (int i = 1; i < N; i *= 2) {
    for (int j = 0; j < i; j++) {
        IO.println("dwang 2.0");
    }
  }
}
```

Runtime: _____

3 I Am Speed

For each code block below, fill in the blank(s) so that the function has the desired runtime. 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; _____________________________){
        System.out.println("hi Selina");
    }
}
```

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

```
public static void f2(int N) {
    for (int i = 1; i < N; _____________________________) {
        System.out.println("howdy Caitlin");
    }
}
```

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

```
public static void f3(int N) {
    for (int i = 1; _____________________________; i += 1) {
        System.out.println("hello Dawn");
    }
}
```

(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 *= 2) {
        for (int j = 0; j < i; j += 1) {
            System.out.println("what's up Alyssa");
        }
    }
}
```

4 Re-cursed with Asymptotics

(a) What is the runtime of the code below in terms of N?

```
public static int curse(int n) {
    if (n <= 0) {
        return 0;
    } else {
        return n + curse(n - 1);
    }
}
```

Runtime: _____

(b) Can you find a runtime bound for the code below? We can assume the **System.arraycopy** method takes $\Theta(N)$ time, where N is the number of elements copied. The official signature is **System.arrayCopy(Object sourceArr, int srcPos, Object dest, int destPos, int length)**. Here, **srcPos** and **destPos** are the starting points in the source and destination arrays to start copying and pasting in, respectively, and **length** is the number of elements copied.

```
public static void silly(int[] arr) {
    if (arr.length <= 1) {
        System.out.println("You won!");
        return;
    }

    int newLen = arr.length / 2;
    int[] firstHalf = new int[newLen];
    int[] secondHalf = new int[newLen];

    System.arraycopy(arr, 0, firstHalf, 0, newLen);
    System.arraycopy(arr, newLen, secondHalf, 0, newLen);

    silly(firstHalf);
    silly(secondHalf);
}
```

Runtime: _____

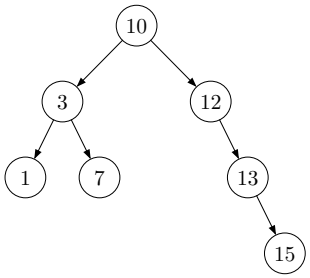
(c) Given that **exponentialWork** runs in $\Theta(3^N)$ time with respect to input N, what is the runtime of **amy**?

```
public void amy(int N) {
    if (N <= 1) {
        return;
    }
    amy(N - 2);
    amy(N - 2);
    amy(N - 2);
    exponentialWork(N); // Runs in Theta(3^N) time
}
```

Runtime: _____

5 Numerical Trees

(a) Write the DFS pre-order, DFS in-order, DFS post-order, and BFS traversals of the following binary tree. For all traversals, process child nodes left to right.



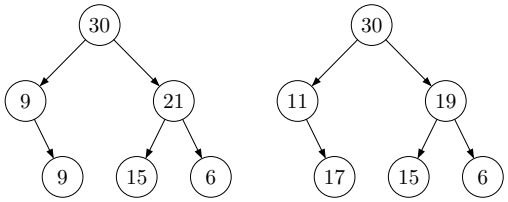
Pre-order: _____

In-order: _____

Post-order: _____

BFS: _____

(b) Complete isSumTree below. In a sum tree, the value at each non-leaf node equals the sum of its children. For example, the left binary tree is a sum tree, but the right one is not.




```
public class Tree {
  public int value;
  public Tree left, right;
  public Tree() {}
  public Tree(int val) { value = val; }
}

public boolean isSumTree(Tree t) {
  if (_____________________ || ____________________________________________) {
    return true;
  }

  int left = 0, right = 0;
  if (_____________________________) {
    left = ____________________________________________;
  }
  if (_____________________________) {
    right = ____________________________________________;
  }

  boolean leftValid = ____________________________________________;
  boolean rightValid = ____________________________________________;

  return ____________________________ && leftValid && rightValid;
}
```

6 Binary Search Trees

(a) We implement a binary search tree with the following methods:

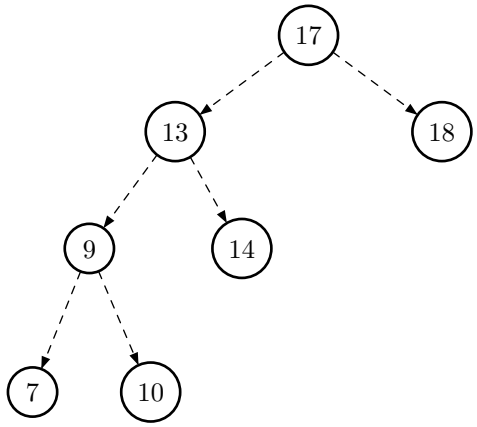
```
// Inserts an item into the binary search tree.
public void insert(T item) {
    // Implementation has been omitted
}

// Deletes an item from the binary search tree.
public void delete(T item) {
    // Implementation has been omitted
}
```

Draw the binary search tree that results from the following operations. Assume we start from an empty tree.

```
insert(5)
insert(7)
insert(10)
insert(6)
insert(3)
insert(1)
insert(4)
delete(10)
delete(7)
```

(b) Given the following binary search tree:



Suppose we delete the root node. Which node(s) can we replace 17 with as the new root node?

(c) Suppose we create a BST by inserting the nodes $V_0, V_1, ..., V_n$, where V_i is strictly smaller than $V_{\{i+1\}}$, in order. That is, we first insert V_0, then V_1, and so on. What is the runtime to find an element in this BST in the worst case, where N is the number of nodes?