Section 03: Asymptotic Analysis

1. Analysis

For each of the following code blocks, what is the worst-case runtime? Give a big-Θ bound.

(a)
```
public IList<String> repeat(DoubleLinkedList<String> list, int n) {
    IList<String> result = new DoubleLinkedList<String>();
    for(String str : list) {
        for(int i = 0; i < n; i++) {
            result.add(str);
        }
    }
    return result;
}
```

(b)
```
public void foo(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 5; j < i; j++) {
            System.out.println("Hello!");
        }

        for (int j = i; j >= 0; j -= 2) {
            System.out.println("Hello!");
        }
    }
}
```

(c)
```
public int num(int n){
    if (n < 10) {
        return n;
    } else if (n < 1000) {
        return num(n - 2);
    } else {
        return num(n / 2);
    }
}
```

(d)
```
public int foo(int n) {
    if (n <= 0) {
        return 3;
    }
    int x = foo(n - 1);
    System.out.println("hello");
    x += foo(n - 1);
    return x;
}
```

2. Recurrences

For each of the following recurrences, use the unfolding method to first convert the recurrence into a summation. Then, find a big-Θ bound on the function in terms of n. Assume all division operations are integer division.

(a) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n/2) + 3 & \text{otherwise} \end{cases}$

(b) $T(n) = \begin{cases} 1 & \text{if } n = 0 \\ T(n-1) + 2 & \text{otherwise} \end{cases}$

(c) $T(n) = \begin{cases} 1 & \text{if } n = 0 \\ T(n/3) + 4 & \text{otherwise} \end{cases}$

(d) $T(n) = \begin{cases} 1 & \text{if } n = 0 \\ 2T(n/3) + n & \text{otherwise} \end{cases}$

(e) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n-1) + 1 & \text{otherwise} \end{cases}$

(f) $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/2) + 100 & \text{otherwise} \end{cases}$

3. Modeling recursive functions

Consider the following method.

```
public static int f(int n) {
    if (n == 0) {
        return 0;
    }

    int result = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            result += j;
        }
    }
    return 5 * f(n / 2) + 3 * result + 2 * f(n / 2);
}
```

(a) Find a recurrence $T(n)$ modeling the worst-case runtime of `f(n)`.

(b) Find a recurrence $W(n)$ modeling the *integer output* of `f(n)`.

4. Modeling recursive functions 2

```
public static int g(n) {
    if (n <= 1) {
        return 1000;
    }
    if (g(n / 3) > 5) {
        for (int i = 0; i < n; i++) {
            System.out.println(”Hello”);
        }
        return 5 * g(n / 3);
    } else {
        for (int i = 0; i < n * n; i++) {
            System.out.println(”World”);
        }
        return 4 * g(n / 3);
    }
}
```

(a) Find a recurrence $S(n)$ modeling the worst-case runtime of `g(n)`.

(b) Find a recurrence $X(n)$ modeling the *integer output* of `g(n)`.

5. Modeling recursive functions 3

Consider the following set of recursive methods.

```
public int test(int n) {
    IDictionary<Integer, Integer> dict = new AvlDictionary<>();
    populate(n, dict);
    int counter = 0;
    for (int i = 0; i < n; i++) {
        counter += dict.get(i);
    }
    return counter;
}

private void populate(int k, IDictionary<Integer, Integer> dict) {
    if (k == 0) {
        dict.put(0, k);
    } else {
        for (int i = 0; i < k; i++) {
            dict.put(i, i);
        }
        populate(k / 2, dict);
    }
}
```

(a) Write a mathematical function representing the *worst-case runtime* of `test`.

You should write two functions, one for the runtime of `test` and one for the runtime of `populate`.

(b) Write a mathematical function representing the *integer output* of `test`.

6. AVL Trees

(a) Draw an AVL Tree as each of the following keys are added in the order given. Show intermediate steps.

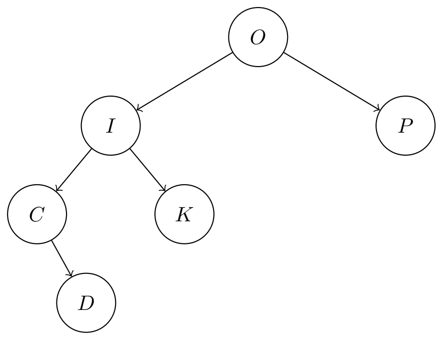
$$\{13, 17, 14, 19, 22, 18, 11, 10, 21\}$$

(b) Draw an AVL Tree as each of the following keys are added in the order given. Show intermediate steps.

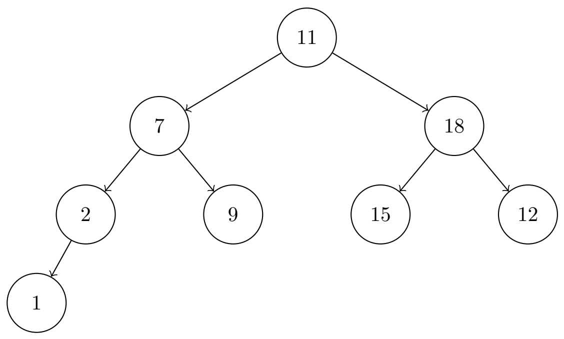
$$\{1, 2, 3, 4, 5, 6\}$$

7. More AVL Trees

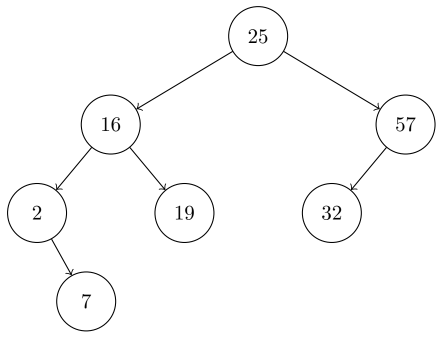
(a) Is this a valid AVL tree? Explain your answer.



(b) Is this a valid AVL tree? Explain your answer.



(c) Is this a valid AVL tree? Explain your answer.



8. Algorithm Design

(a) Given a binary search tree, describe how you could convert it into an AVL tree with worst-case time $\mathcal{O}(n\log(n))$. What is the best case runtime of your algorithm?

(b) Given an AVL tree, describe how would you do a level order tree traversal. What is the worst-case runtime of your algorithm?