CS 61B Spring 2019

Asymptotics II, Search Trees

Discussion 7: March 4, 2019

Here's a review of some formulas you will find useful when doing asymptotic analysis.

- $\sum_{i=1}^{N} i = 1 + 2 + 3 + 4 + \cdots + N = \frac{N(N+1)}{2} = \frac{N^2+N}{2} \in \mathbf{\Theta(N^2)}$
- $\sum_{i=0}^{N} 2^i = 1 + 2 + 4 + 8 + \cdots + 2^N = 2 \cdot 2^N - 1 \in \mathbf{\Theta(2^N)}$
- $N + \frac{N}{2} + \ldots + 2 + 1 = 2N - 1 \in \mathbf{\Theta(N)}$

1 Space Jam 2

For each of the following recursive functions, give the worst case and best case runtime in $\Theta(\cdot)$ notation.

1.1 Give the running time in terms of N.

```
1 public void andslam(int N) {
2     if (N > 0) {
3         for (int i = 0; i < N; i += 1) {
4             System.out.println("datboi.jpg");
5         }
6         andslam(N / 2);
7     }
8 }
```

1.2 Give the running time for `andwelcome(0, N)` in terms of N.

```
public static void andwelcome(int low, int high) {
    System.out.print("[ ");
    for (int i = low; i < high; i += 1) {
         System.out.print("loyal  ");
    }
    System.out.println("]");
    if (high - low > 0) {
        double coin = Math.random();
        if (coin > 0.5) {
            andwelcome(low, low + (high - low) / 2);
        } else {
            andwelcome(low, low + (high - low) / 2);
            andwelcome(low + (high - low) / 2, high);
        }
    }
}
```

1.3 Give the running time in terms of N.

```
public int tothe(int N) {
    if (N <= 1) {
        return N;
    }
    return tothe(N - 1) + tothe(N - 1);
}
```

1.4 *Extra:* Give the running time in terms of N. An $\mathcal{O}$-bound is sufficient.

```
public static void spacejam(int N) {
    if (N <= 1) {
        return;
    }
    for (int i = 0; i < N; i += 1) {
        spacejam(N - 1);
    }
}
```

2 Is This a BST?

The following method `buggyIsBST` is supposed check if a given binary tree is a BST, though for some binary trees, it is returning the wrong answer. Think about an example of a binary tree for which `buggyIsBST` fails. Then, write `isBST` so that it returns the correct answer for any binary tree. The `TreeNode` class is defined as follows:

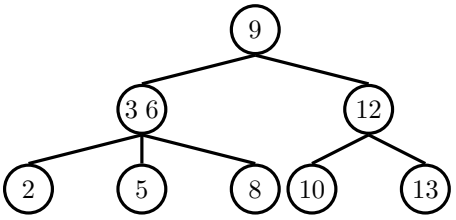
```
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;
}
```

Hint: You will find `Integer.MIN_VALUE` and `Integer.MAX_VALUE` helpful when writing `isBST`.

```
public static boolean buggyIsBST(TreeNode T) {
    if (T == null) {
        return true;
    } else if (T.left != null && T.left.val > T.val) {
        return false;
    } else if (T.right != null && T.right.val < T.val) {
        return false;
    } else {
        return buggyIsBST(T.left) && buggyIsBST(T.right);
    }
}

public static boolean isBST(TreeNode T) {
    return isBSTHelper(                                          );
}

public static boolean isBSTHelper(                                    ) {









}
```

3 … as all Trees Should be

Consider the 2-3 tree below. What order should we insert these numbers so that we get the tree shown? There may be multiple correct answers.



What is the **minimum** number of insertions that one can make to the above tree to cause the root to split? Assume we insert no duplicate items.

Extra: What is the **maximum** number of insertions one can make to the above tree **without** splitting the root? Assume we insert no duplicate items.