LEC 9

# CSE 123

# Exhaustive Search

Questions during Class?
Raise hand or send here

sli.do #cse123

BEFORE WE START

***Talk to your neighbors:***

*What is your favorite Halloween/Spooky/Scary movie?*

**Respond on sli.do!**

**Instructor:** Nathan Brunelle

**TAs:**

| | | | | |
|---|---|---|---|---|
| Cora | Chris | Crystal | Anya | Evan |
| Jonah | Nhan | Saachi | Ishita | Sahana |
| Chloe | Shiven | Nate | Renee | Yuntong |
| Amiya | Shashmit | Akshaya | Vihan | Shreedhaarini |
| Roshan | Adiyan | Nancy | Rachel | Ananya |
| Ryuta | Mimi | Isaac | Megan | |

# Announcements

- Paper Quiz 0 on Tuesday 11/3
  - Practice paper quiz is available in Ed Resources
- Programming Project 1 due Wednesday 11/4
- Computer Quiz 1 begins 11/5

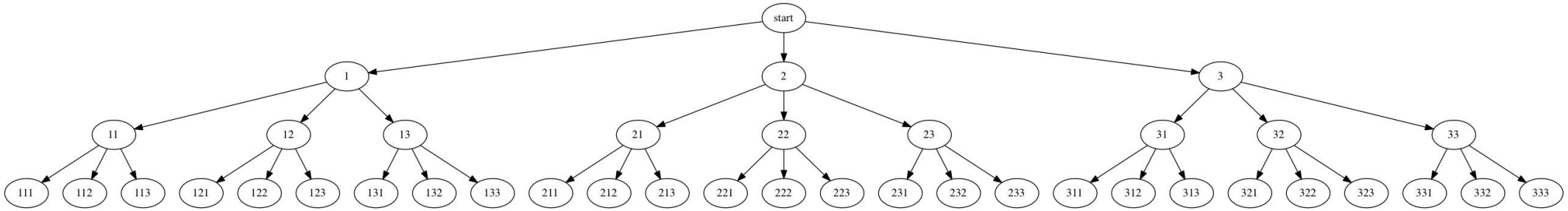
# Exhaustive Search Approach

- We suppose we want to explore the space of all possible solutions…

- So what do we do?
  - We "exhaustively search" through every possibility
  - We need some sort of plan or process to follow to do this programmatically

- What do we need? Recursion + some kind of accumulator
  - public / private pair

# Tracing through printNums

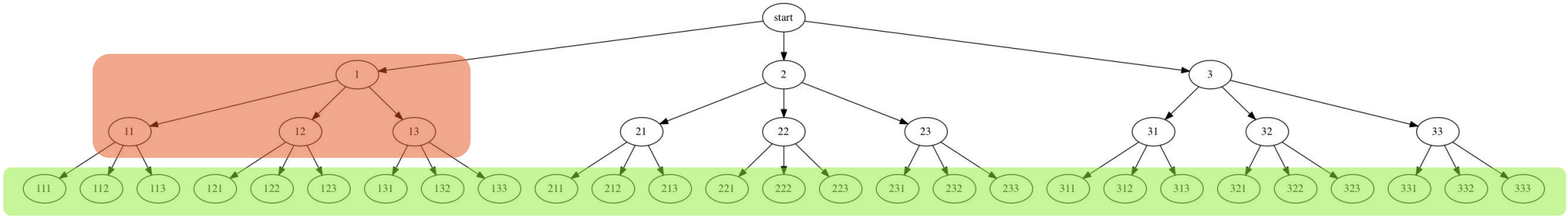



```
public static void printNums() {
    printNums("");
}

private static void printNums(String soFar) {
    if (soFar.length() == 3) {
        System.out.println(soFar);
    } else {
        printNums(soFar+ "1");
        printNums(soFar+ "2");
        printNums(soFar+ "3");
    }
}
```

# Decision Trees

- Visual we use to help understand what our process is
  - Visualization tool, not a data structure
  - If you can draw a decision tree, you can implement exhaustive search




- Can glean important information
  - **Base case (end nodes)**
  - **Recursive case (middle nodes)**
  - **"Dead end" case (more on this later…)**

# Exhaustive Search Pattern (search)

```
public static void search(input) {
    search(input, "");
}

private static void search(input, String soFar) {
    if (base case) {
        // Do something with soFar (e.g. print it out)
        System.out.println(soFar);
    } else {
        // Might not be a loop, but 1 recursive call for each option
        for (each option) {
            search(input, soFar + option);
        }
    }
}
```

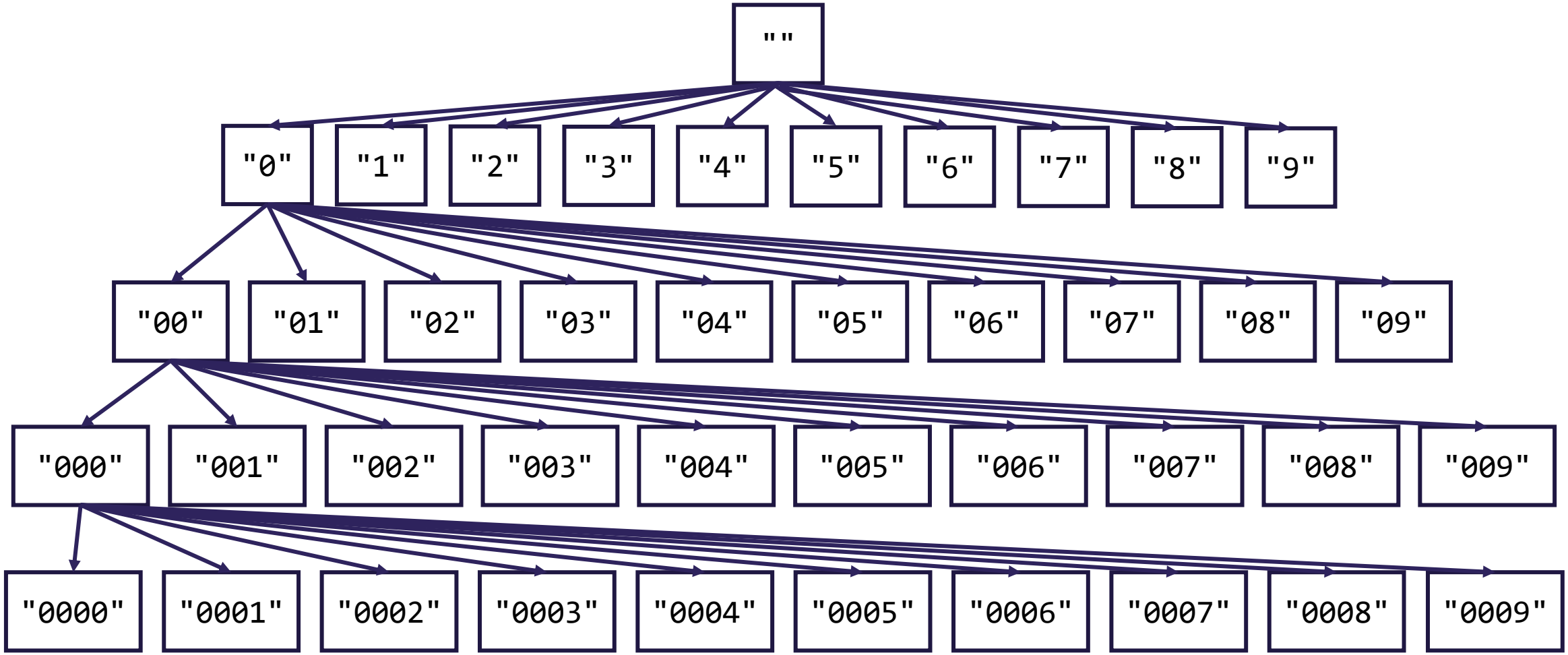
# Exhaustive Search Pattern (printNums)

```
public static void printNums() {
    printNums("");
}

private static void printNums(String soFar) {
    if (soFar.length() == 3) {
        // Do something with soFar (e.g. print it out)
        System.out.println(soFar);
    } else {
        // Might not be a loop, but 1 recursive call for each option
        for (int i = 1; i <= 3; i++) {
            printNums(soFar + i);
        }
    }
}
```

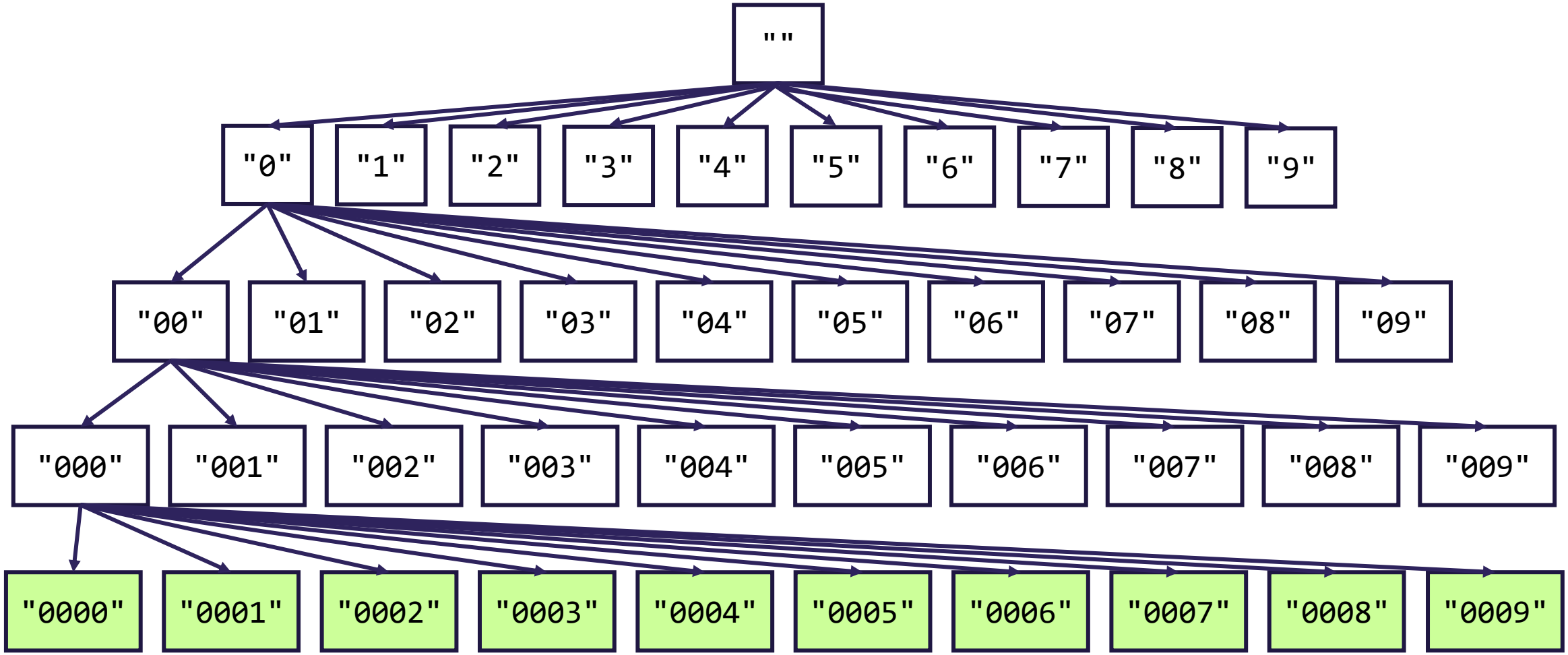
# Password Cracker: Decision Tree

- Let's say we want to crack the password of a 4 digit combination lock



# Password Cracker: End nodes will be handled by the base case

- Let's say we want to crack the password of a 4 digit combination lock



# Password Cracker: Intermediate nodes are handled by the recursive case

- Let's say we want to crack the password of a 4 digit combination lock

