

LEC 10

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 current favorite podcast or
YouTube channel?*

[Respond on sli.do!](#)

Instructor: Brett Wortzman

TAs:

Arohan	Jonah	Kavya	Eeshani	Trien
Ashar	Brice	Misha	Aidan	Evan
Sean	Chris	Kieran	Cora	Rena
Chloe	Elden	Sahana	Dixon	Katharine
Jenny	Ishita	Anirudh	Nhan	Anyia
Nate	Kuhu	Crystal		

Now playing:  [CSE 123 26wi Lecture Tunes](#) 

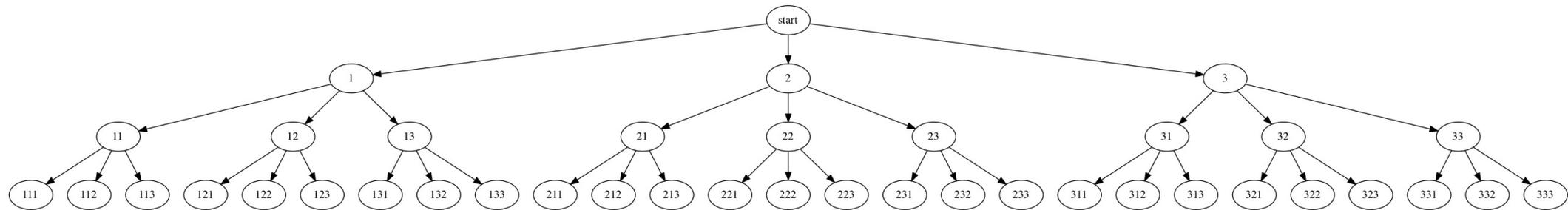
Announcements

- Yay Quiz 1 is done!
 - Again, grades before Quiz 2 but we have makeups to take care of...
 - Quiz 2 is scheduled for March 3, so you have a bit of a break!
- Programming Assignment 1 due tonight (2/11) at 11:59pm
- Creative Project 2 released tomorrow due in one week (Wed, 2/18)
 - Focused on recursion!
- Resubmission Cycle 3 is open, closes on 2/13
 - PO, C1 eligible

Exhaustive Search

- 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

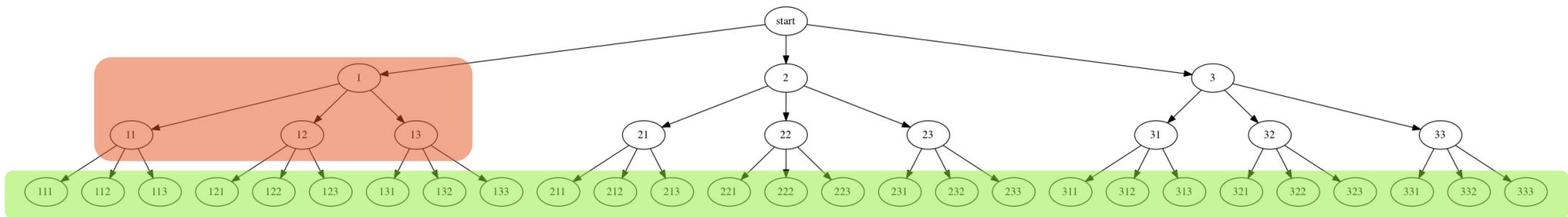


```
public static void printNums() {  
    printNums("");  
}
```

```
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);
        }
    }
}
```

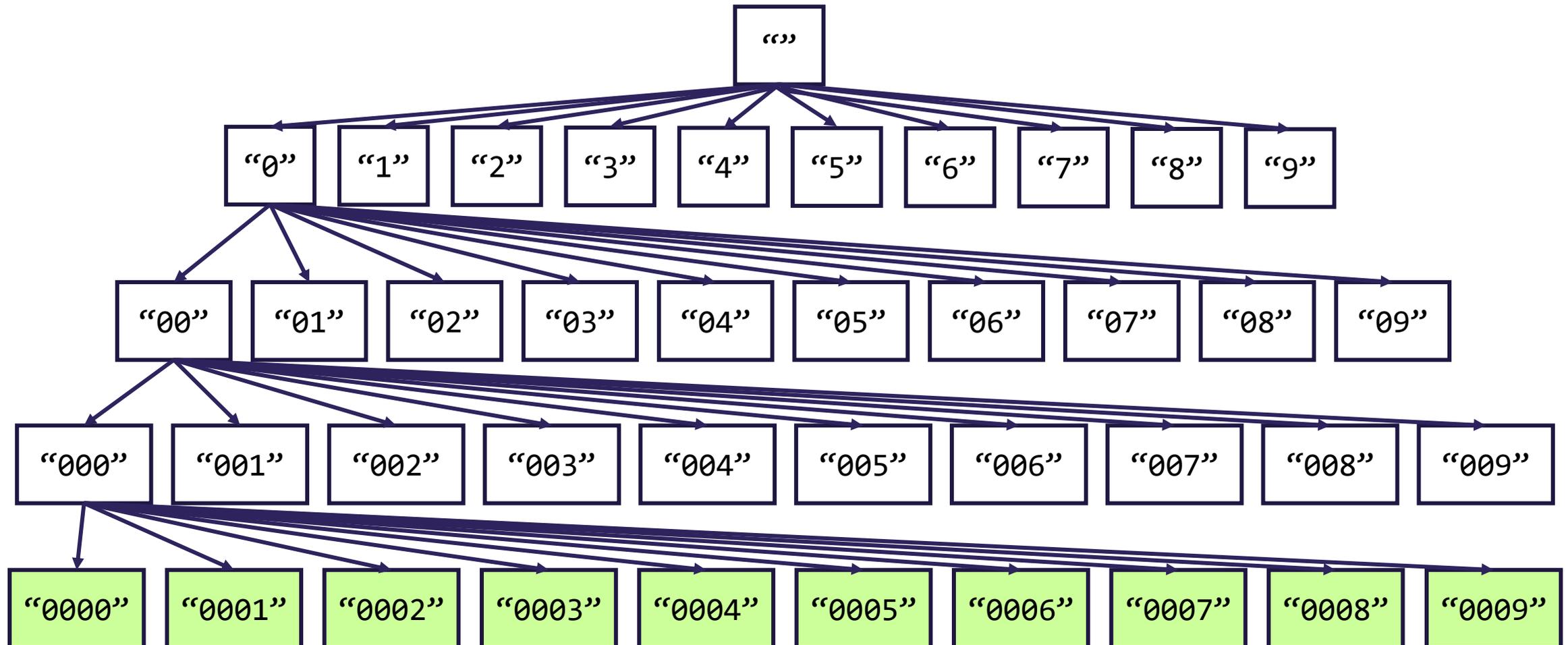
Exhaustive Search Pattern (printNums)

```
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

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



Password Cracker

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

