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: Miya Natsuhara

TAs:

Arohan	Shiven	Yuntong	Anya
Sreshta	Vrinda	Amiya	Anisha
Rushil	Gavin	Sahana	Trien
Sean B	Shreya	Anirudh	Neal
Chloe	Jonah	Rohan	Evan
Jenny	Renee	Crystal	Rena
Nate	Chris	Eeshani	
Saachi	Ishita	Prakshi	
Hawa	Kuhu	Aidan	
Maggie	Kavya	Cora	
Sean E	Misha	Nhan	

Music: CSE 123 26sp 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 May 25, so you have a bit of a break!
- Programming Assignment 1 due tonight (5/6) at 11:59pm
- Creative Project 2 released tomorrow due in one week (Wed,5/13)
  - Focused on recursion!
- Resubmission Cycle 3 is open, closes on 5/8
  - ***P0***, C1 eligible
- CSE 12x TA application for Autumn 2026 is open! Consider applying!

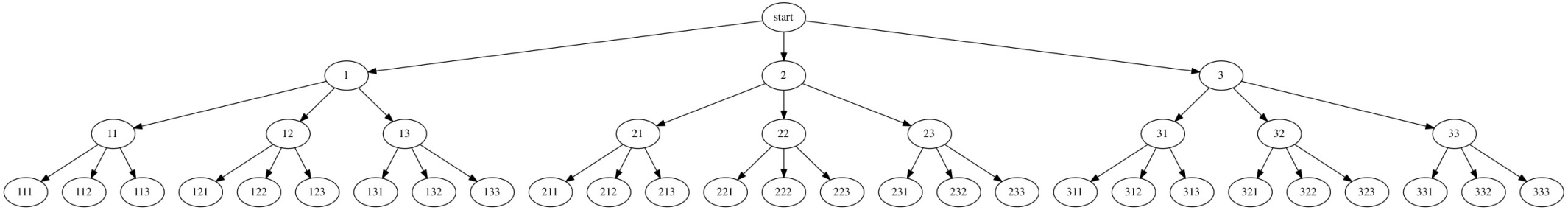
Exhaustive Search Approach

- 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

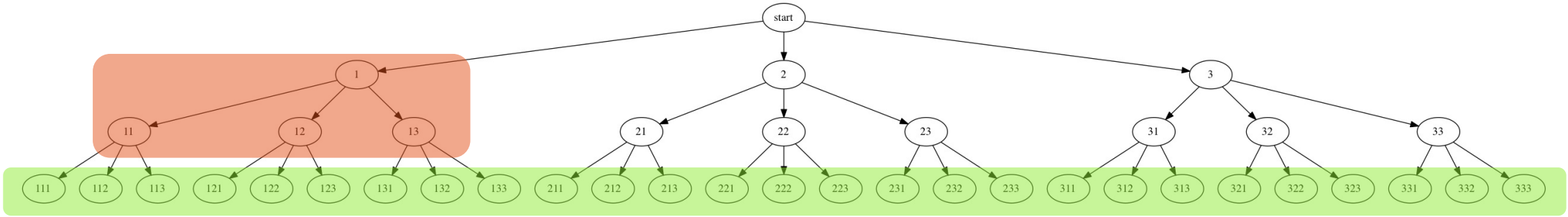



```
public static void printNums() {
    printNums("");
}

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);
        }
    }
}
```

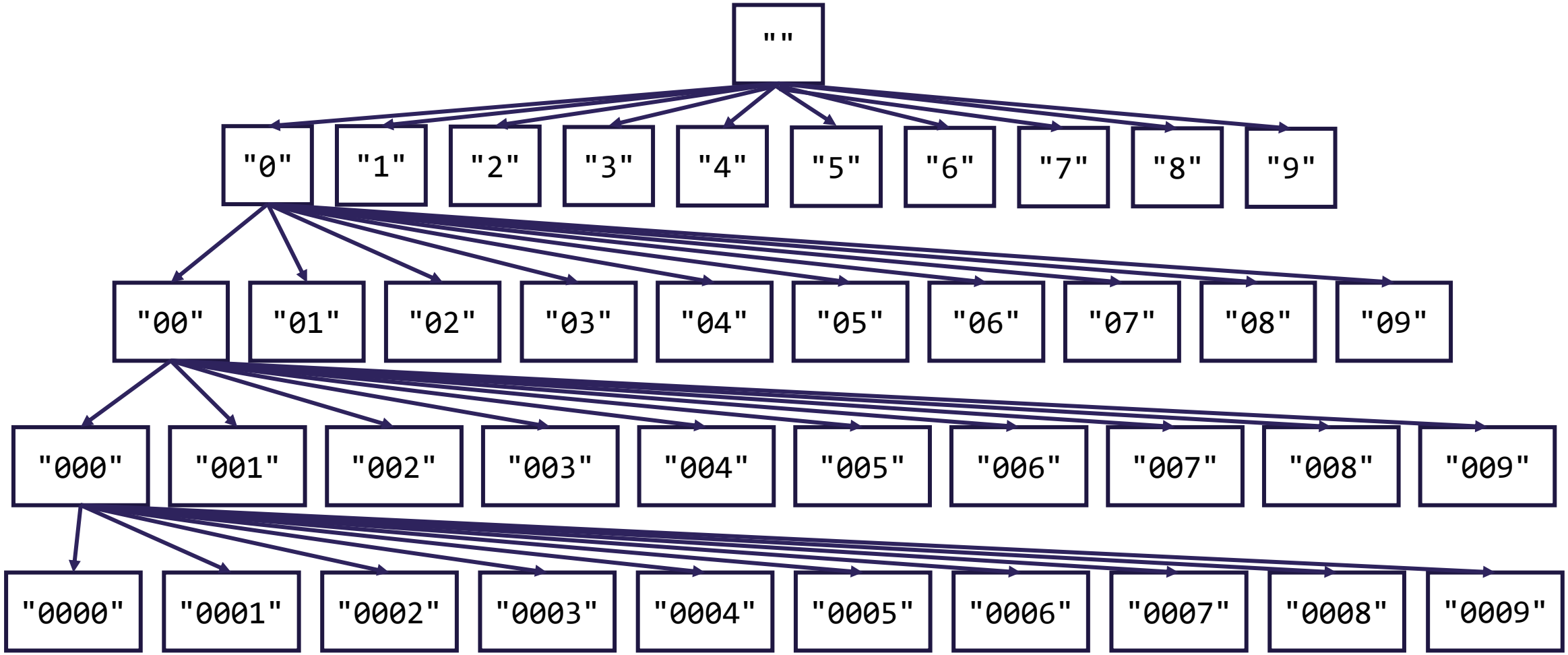
Exhaustive Search Pattern (printNums)

```
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);
        }
    }
}
```

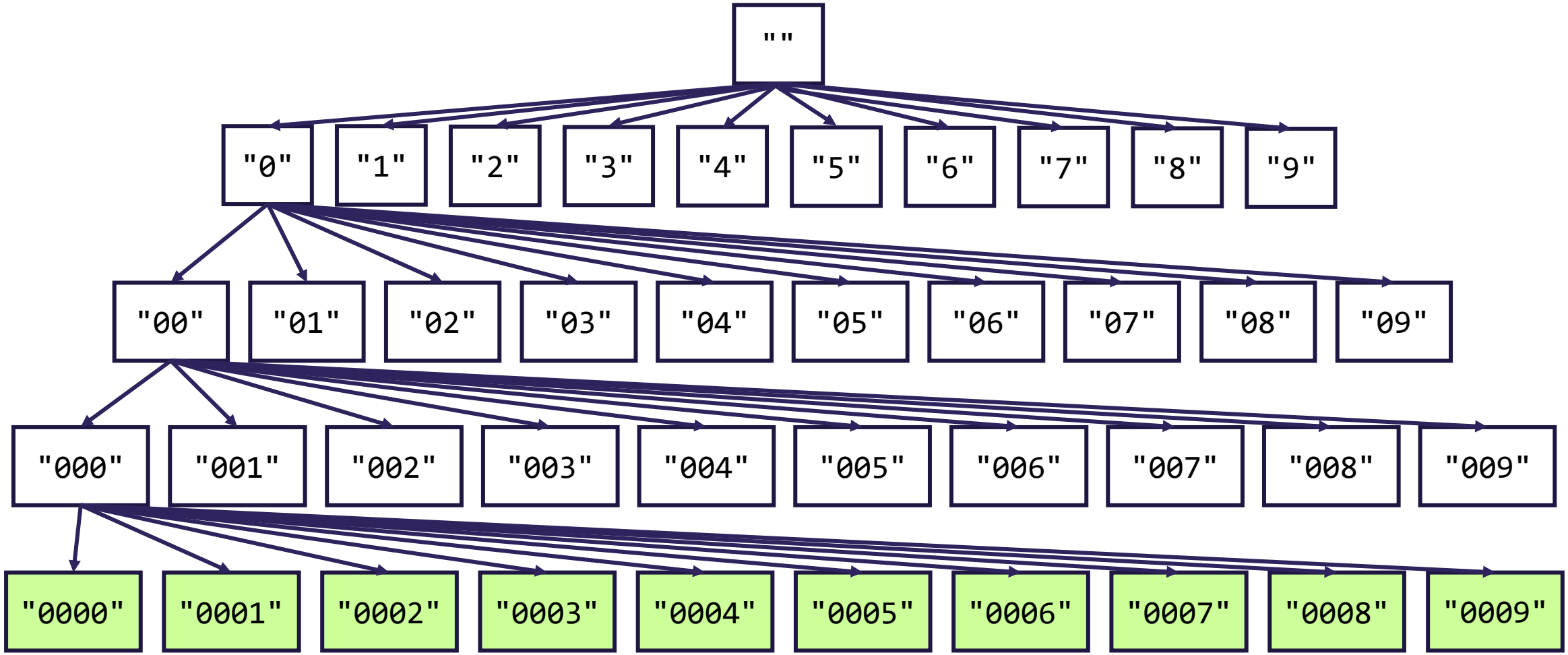
Password Cracker: Decision Tree

- 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