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's your favorite rainy day activity?

Instructors: Brett Wortzman
Miya Natsuhara

TAs:

Arohan	Neha	Rushil	Johnathan	Nicholas
Sean	Hayden	Srihari	Benoit	Isayiah
Audrey	Chris	Andras	Jessica	Kavya
Cynthia	Shreya	Kieran	Rohan	Eeshani
Amy	Packard	Cora	Dixon	Nichole
Trien	Lawrence	Liza	Helena	

Music: CSE 123 25wi Lecture Tunes

Announcements

- Yay Quiz 1 is done!
  - Again, grades before Quiz 2 but we have some makeups we need to take care of…
  - Quiz 2 is scheduled for March 4, so you have a bit of a break!
- Programming Assignment 1 due tonight (Feb 12) at 11:59pm
- Creative Project 2 released tomorrow (Thurs, Feb 13), due in one week (Wed, Feb 19)
  - Focused on recursion!
- Resubmission Cycle 3 is open, closes on Friday, Feb 14
  - ***P0***, C1 eligible
- The CSE 12x/14x TA application is now open for Spring 2025!

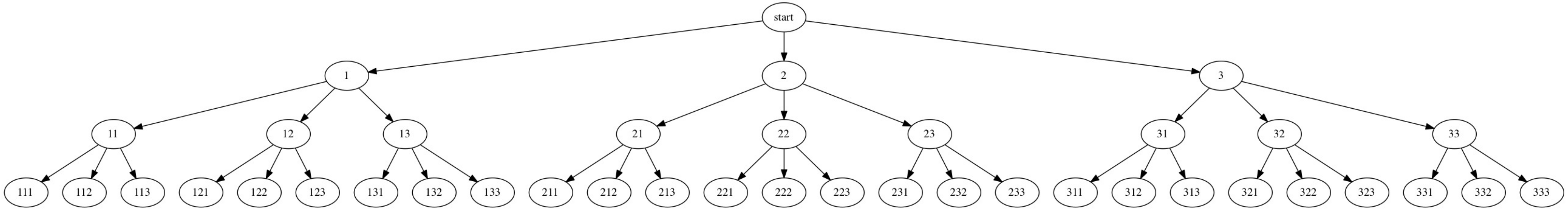
Exhaustive Search

- 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

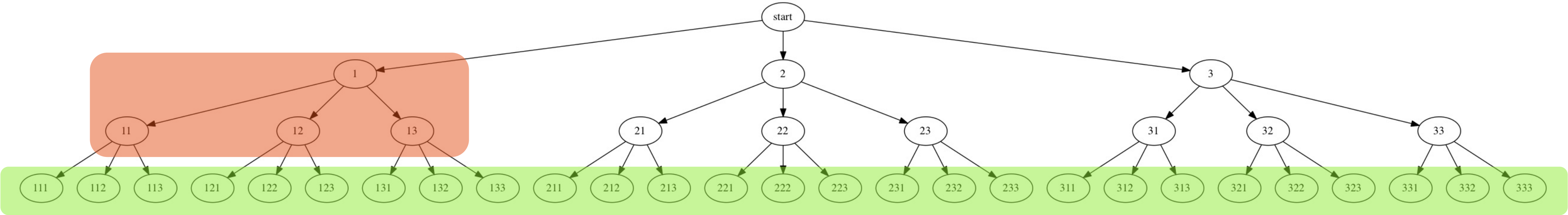


```
public static void printNums() {
    printNums("");
}

private static void printNums(String soFar) {
    if (soFar.length() == 3) {
        count++;
        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);
        }
    }
}
```

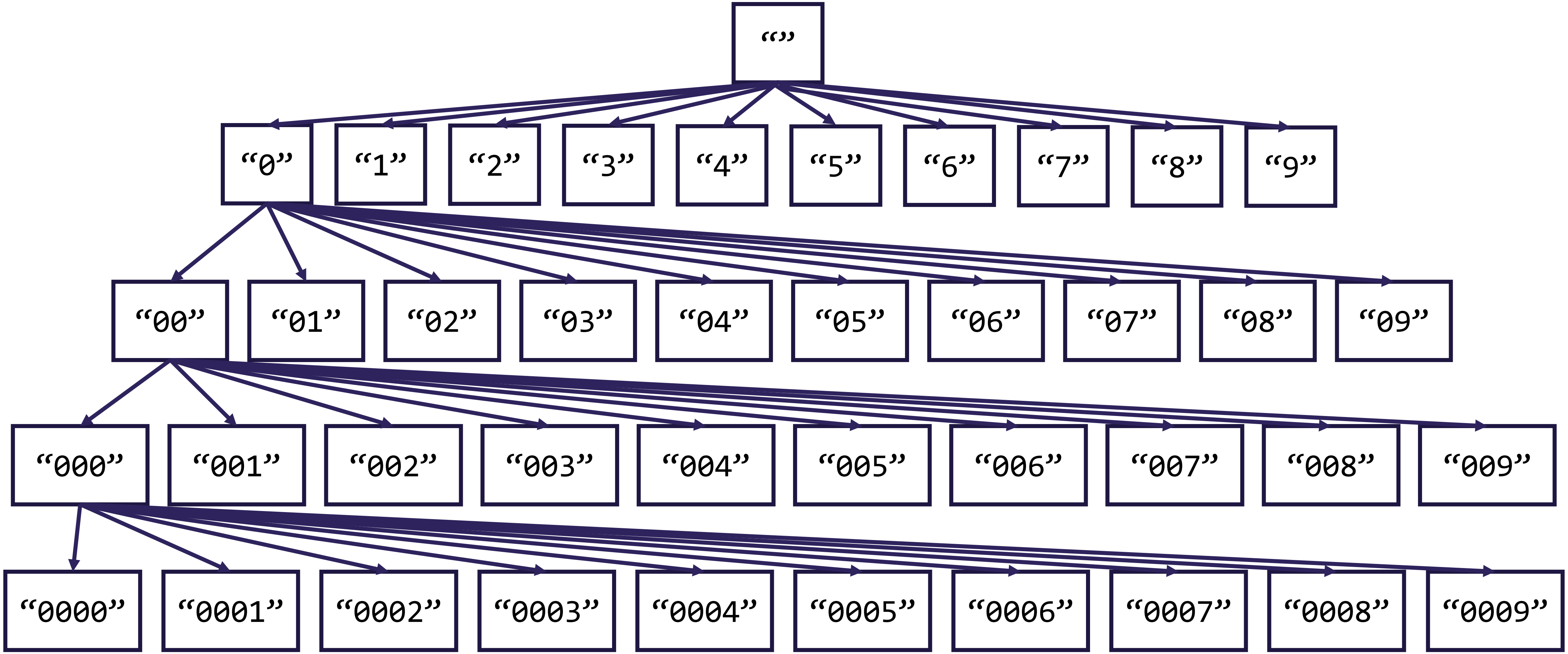
Exhaustive Search Pattern (printNums)

```java
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);
        }
    }
}
```

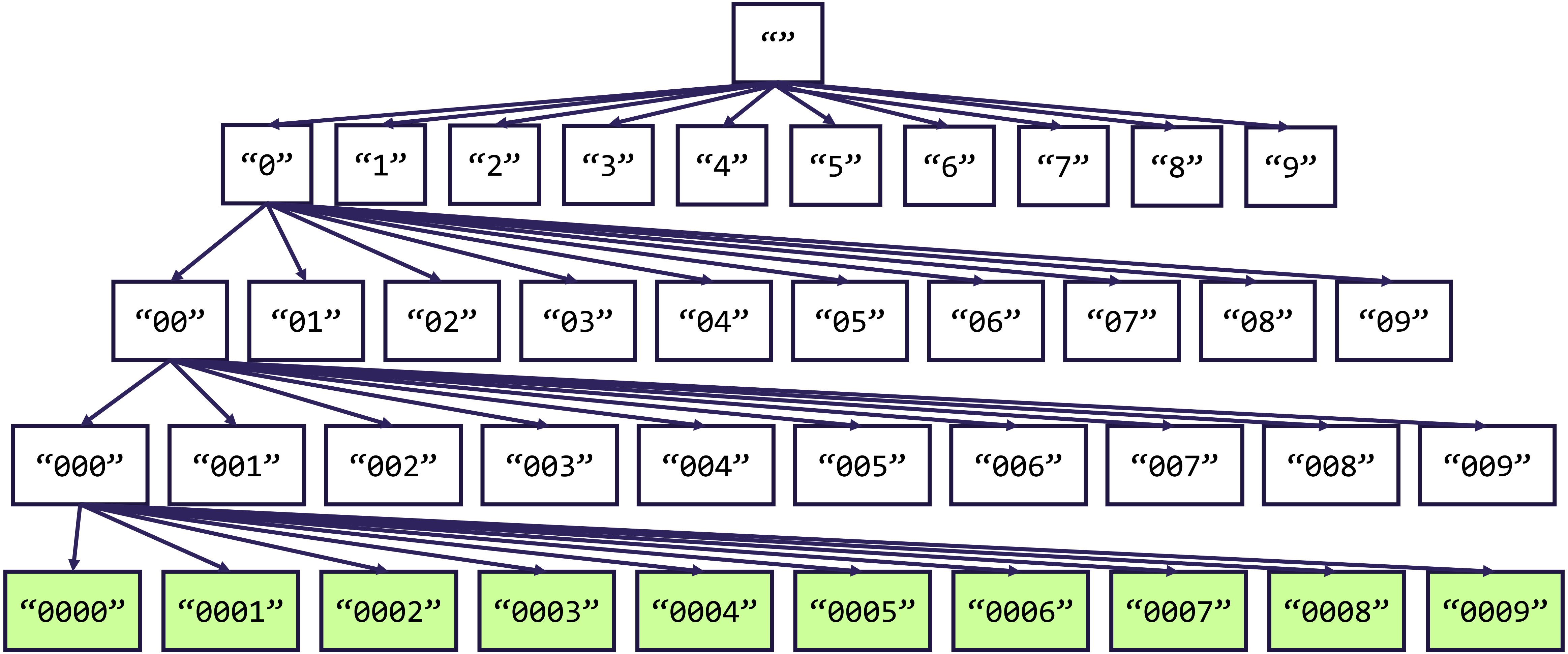
Password Cracker

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



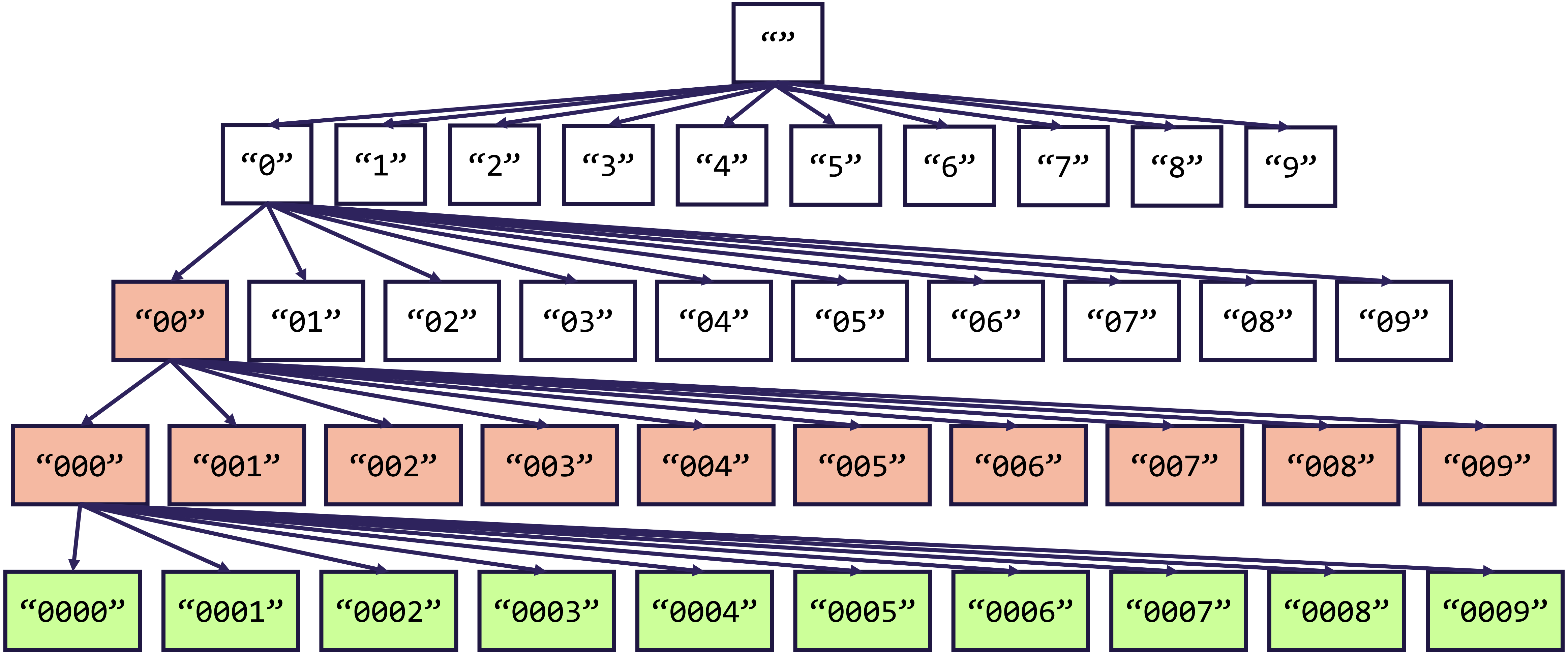
Password Cracker

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



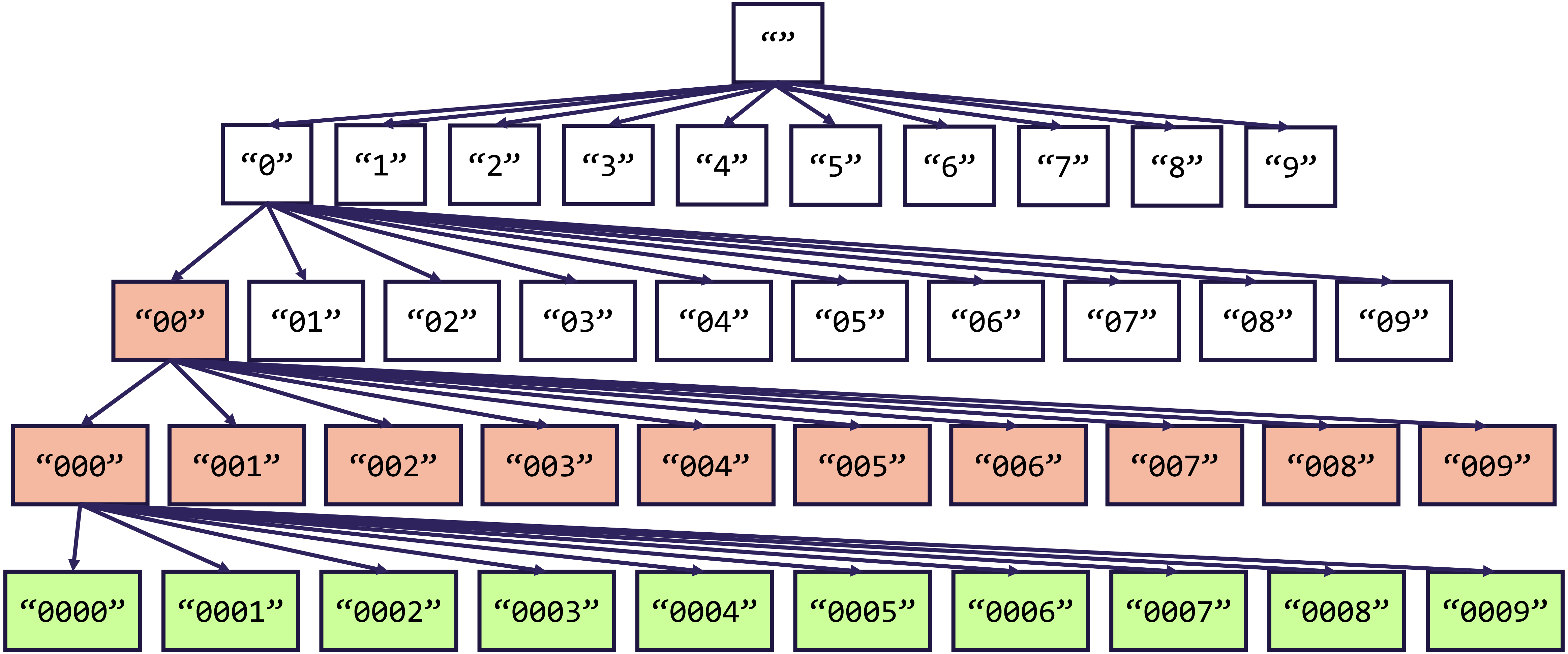
Password Cracker

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



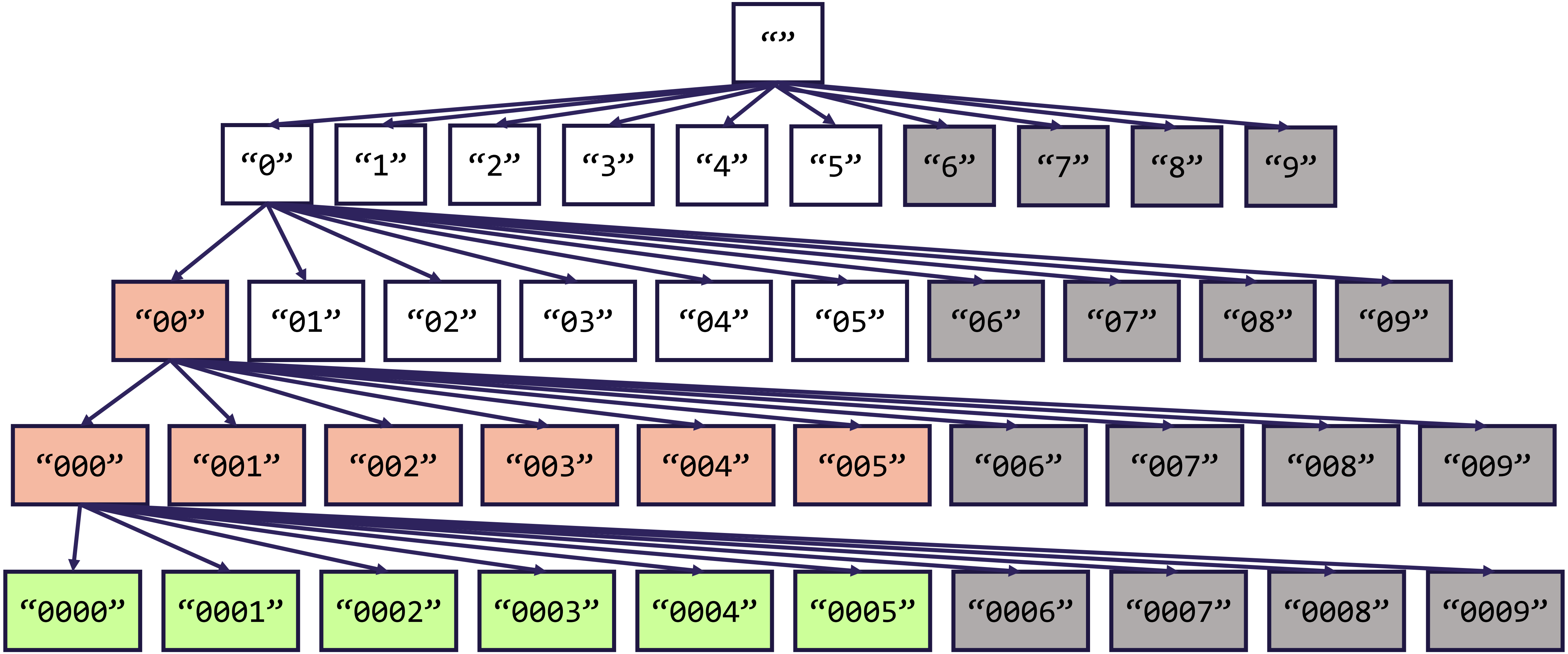
Password Cracker

- Now, what if we knew the sum of all digits was 5?



Password Cracker

- Now, what if we knew the sum of all digits was 5?



Updated Exhaustive Search Pattern

```
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 if (not dead end) {
        // Might not be a loop, but 1 recursive call for each option
        for (each option) {
            search(input, soFar + option);
        }
    }
}
```

Sidenote:

- There are some problems computers can solve, but not very cleverly…

- Two "classes" of problems…
  - **P**olynomial
    - Problems with a polynomial-time solution
  - **N**ondeterministic **P**olynomial
    - *Problems that can be solved by a non-deterministic Turing machine in polynomial time…*
    - Problems that we don't think have polynomial-time solutions…
    - Often these solutions are *exponential* time because we are sort of "brute-forcing" a solution…
      - Generative every possible solution and see if it works!

- Open problem: P = NP?