Programming Abstractions

CS106B

Cynthia Lee

Today's topics:

- Previous lectures:
  - Introduction to recursion with Factorial
  - Mechanics of recursion: looking at the stack frames
  - Visual example: Boxy "snowflake" fractal
  - Classic, widely-used CS algorithm example: Binary Search
- Today and Monday:
  - New patterns of recursion application: adding loops
    - Loops + recursion for *generating sequences and combinations*
    - Loops + recursion for *recursive backtracking*

Generating sequences and combinations

Recursion pattern: generating sequences and combinations

- Example problems:
  - Given a deck of cards, output all possible distinct 5-card poker hands
  - Generate all possible passwords of length N
  - Print all possible 7-digit phone numbers
  - Generate all possible 4-digit PINs
  - › *These are all variants of "generate all strings of length N," but with different alphabets to choose from*

Problem: generating sequences (approach #1 BUG)

- How can we code generating all possible 4-digit PINs?
- Let's try: **loops**

```
Vector<Vector<int>> allPINs;
for (int dig1 = 0; dig1 <= 9; dig1++) {// try all possible 1st digits
    for (int dig2 = 0; dig2 <= 9; dig2++) {// for each, try all possible 2nd digits
        for (int dig3 = 0; dig3 <= 9; dig3++) {// for each, try all possible 3rd...
            for (int dig4 = 0; dig4 <= 9; dig4++) {// and all possible 4th
                Vector<int> PIN(4);
                PIN[0] = dig1;
                PIN[1] = dig2;
                PIN[2] = dig3;
                PIN[3] = dig4;
                allPINs.add(PIN);
            }
        }
    }
}
```

- **Question:** how do we get this code to work for any N-digit PINs, not just 4-digit?
  ANSWER: ☹ Can't dynamically retype code!

Recursion pattern: generating sequences

```
void generateAllPINs(int length, Vector<Vector<int>> &allPINs,
                     Vector<int> currentPIN) {
    if (currentPIN.size() == length) {
        allPINs.add(currentPIN);
        return;
    }
    currentPIN.add(0); // add a placeholder last digit
    for (int digit = 0; digit <= 9; digit++) {
        // keep overwriting last digit with all possible digits
        currentPIN[currentPIN.size()-1] = digit;
        // recursively try all possible remaining digits
        generateAllPINs(length, allPINs, currentPIN);
    }
}
```

Recursion pattern: designing a “wrapper” function

```
// PUBLIC-FACING FUNCTION – USE THIS ONE (called the "wrapper")
void generateAllPINs(int length, Vector<Vector<int>> &allPINs) {
    Vector<int> currentPIN;
    generateAllPINs(length, allPINs, currentPIN); // wrapper's job is to call recursive function
}                                                 // with all supporting arguments set up

// SECRET BEHIND THE SCENES FUNCTION – (the actual recursive function)
void generateAllPINs(int length, Vector<Vector<int>> &allPINs,
                     Vector<int> currentPIN) {
    if (currentPIN.size() == length) {
        allPINs.add(currentPIN);
        return;
    }
    currentPIN.add(0); // add a placeholder last digit
    for (int digit = 0; digit <= 9; digit++) {
        // keep overwriting last digit with all possible digits
        currentPIN[currentPIN.size()-1] = digit;
        // recursively try all possible remaining digits
        generateAllPINs(length, allPINs, currentPIN);
    }
}
```