Programming Abstractions

CS106B

Cynthia Bailey Lee
Julie Zelenski

Today's topics:

- Recursion ~~Week~~ Fortnight continues!
- Today:
  - Loops + recursion for *generating sequences and combinations*
- Upcoming:
  - Loops + recursion for *recursive backtracking*

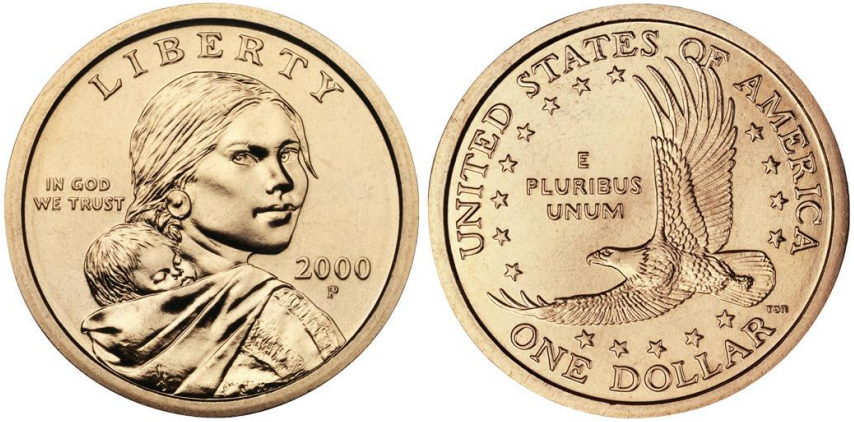
- **Assignment 3 "YEAH" Session TONIGHT**
  - Monday Oct 11, 6:30pm in Durand 410
  - YEAH = Your Early Assignment Help
  - Orientation to what the assignment is asking, tips for getting started, common questions, etc.

Heads or Tails?

GENERATING SEQUENCES

Heads or Tails?

- You flip a coin 5 times
- What are all the possible heads/tails sequences you could observe?
  - TTTTT
  - HHHHH
  - THTHT
  - HHHHT
  - etc…
- We want to write a program to fill a Vector with strings representing each of the possible sequences.



Generating all possible coin flip sequences

```
void generateAllSequences(int length, Vector<string>& allSequences)
{
    string sequence;
    generateAllSequences(length, allSequences, sequence);
}

void generateAllSequences(int length, Vector<string>& allSequences, string sequence)
{
    // base case: this sequence is full-length and ready to add
    if (sequence.size() == length) {
        allSequences.add(sequence);
        return;
    }
    // recursive cases: add H or T and continue
    sequence += "H";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "T";
    generateAllSequences(length, allSequences, sequence);
}
```

Your Turn: coin flip sequences


```
void generateAllSequences(int length, Vector<string>& allSequences, string sequence)
{
    // base case: this sequence is full-length and ready to add
    if (sequence.size() == length) {
        allSequences.add(sequence);
        return;
    }
    // recursive cases: add H or T and continue
    sequence += "H";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "T";
    generateAllSequences(length, allSequences, sequence);
}
```

- **Q: Of these sequences (all of which should be included in `allSequences`), which sequence appears first in `allSequences`? Last?**
  - TTTTT, HHHHH, THTHT, HHHHT

Your Turn: coin flip sequences


```
void generateAllSequences(int length, Vector<string>& allSequences, string sequence)
{
    // base case: this sequence is full-length and ready to add
    if (sequence.size() == length) {
        allSequences.add(sequence);
        return;
    }
    // recursive cases: add H or T and continue
    sequence += "H";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "T";
    generateAllSequences(length, allSequences, sequence);
}
```

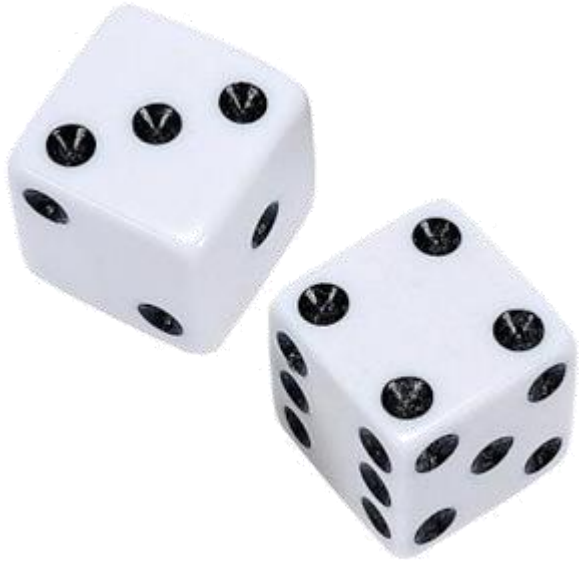
- **Q: What would happen if we didn't do the erase (highlighted above)? Which of the following sequences would we NOT generate? Which additional sequences would we generate (that we shouldn't)?**
  - TTTTT, HHHHH, THTHT, HHHHT

Roll the Dice!

GENERATING MORE SEQUENCES

Roll the Dice!

- You roll a single die 5 times
- What are all the possible 1/2/3/4/5/6 sequences you could observe?
  - 11111
  - 66666
  - 12345
  - 21655
  - etc…
- We want to write a program to fill a Vector with strings representing each of the possible sequences.



Generating all possible ~~coin flip~~ die roll sequences

```
void generateAllSequences(int length, Vector<string>& allSequences)
{
    string sequence;
    generateAllSequences(length, allSequences, sequence);
}

void generateAllSequences(int length, Vector<string>& allSequences, string sequence)
{
    // base case: this sequence is full-length and ready to add
    if (sequence.size() == length) {
        allSequences.add(sequence);
        return;
    }
    // recursive cases: add H or T and continue
    sequence += "H";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "T";
    generateAllSequences(length, allSequences, sequence);
}
```

To adapt for die rolls, we need to change this from 2 options (H/T) to 6 options (1-6).

Generating all possible ~~coin flip~~ die roll sequences

```
    // recursive cases: add 1 or 2 or 3 or 4 or 5 or 6 and continue
    sequence += "1";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "2";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "3";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "4";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "5";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "6";
    generateAllSequences(length, allSequences, sequence);
}
```

This works, but YIKES!!
So much copy-paste!!

Generating all possible ~~coin flip~~ die roll sequences

```
    // recursive cases: add 1 or 2 or 3 or 4 or 5 or 6 and continue
    sequence += "1";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "2";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "3";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "4";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "5";
    generateAllSequences(length, allSequences, sequence);
    sequence.erase(sequence.size() - 1);
    sequence += "6";
    generateAllSequences(length, allSequences, sequence);
}
```

Let's take the repeated actions and put them in a for-loop from 1 to 6.

Generating all possible ~~coin flip~~ die roll sequences

```
void generateAllSequences(int length, Vector<string>& allSequences)
{
    string sequence;
    generateAllSequences(length, allSequences, sequence);
}

void generateAllSequences(int length, Vector<string>& allSequences, string sequence)
{
    // base case: this sequence is full-length and ready to add
    if (sequence.size() == length) {
        allSequences.add(sequence);
        return;
    }
    // recursive cases: add 1-6 and continue
    for (int i = 1; i <= 6; i++) {
        sequence += integerToString(i);
        generateAllSequences(length, allSequences, sequence);
        sequence.erase(sequence.size() - 1);
    }
}
```

Much nicer!!

Generating all possible ~~coin flip~~ die roll sequences

```
void generateAllSequences(int length, Vector<string>& allSequences)
{
    string sequence;
    generateAllSequences(length, allSequences, sequence);
}

void generateAllSequences(int length, Vector<string>& allSequences, string sequence)
{
    // base case: this sequence is full-length and ready to add
    if (sequence.size() == length) {
        allSequences.add(sequence);
        return;
    }
    // recursive cases: add 1-6 and continue
    for (int i = 1; i <= 6; i++) {
        sequence += integerToString(i);
        generateAllSequences(length, allSequences, sequence);
        sequence.erase(sequence.size() - 1);
    }
}
```

Notice that this loop **does not replace** the recursion. It just controls how many times the recursion launches.

Your Turn: die roll sequences

```
void generateAllSequences(int length, Vector<string>& allSequences, string se
{
    // base case: this sequence is full-length and ready to add
    if (sequence.size() == length) {
        allSequences.add(sequence);
        return;
    }
    // recursive cases: add 1-6 and continue
    for (int i = 1; i <= 6; i++) {
        sequence += integerToString(i);
        generateAllSequences(length, allSequences, sequence);
        sequence.erase(sequence.size() - 1);
    }
}
```

- **Q: Of these sequences (all of which should be included in `allSequences`), which sequence appears first in `allSequences`? Last?**
  - 11111, 66666, 12345, 21655