Thinking Recursively
Part III
Assignment 3
Assignment 3

• Assignment 3 (Recursion!) goes out today. It’s due one week from today at the start of class.
  
  • You are permitted to work with a partner on this assignment. Please make sure you understand the requirements for doing so before beginning. They’re on the website.
  
  • There are two optional warm-up problems. We’ll release solutions on Wednesday.
  
• Anton is holding YEAH hours (Your Early Assignment Help hours) tonight in 420-040 from 7PM – 8PM. Highly recommended!
Tracing the Recursion

\[
\{ A, H, I \} \\rightarrow \{ A, H, I \}, \{ A, H \}, \{ A, I \}, \{ A \}, \{ H, I \}, \{ H \}, \{ I \}, \{ \} \\
\{ H, I \} \\rightarrow \{ H, I \}, \{ H \}, \{ I \}, \{ \} \\
\{ I \} \\rightarrow \{ I \}, \{ \} \\
\{ \} \\rightarrow \{ \} 
\]
Analyzing Our Function

- **Useful fact**: Given any $n$-element set, there are $2^n$ subsets of that set.
- The returned collection of sets will need to have space for at least $2^n$ sets.
- For a modest value of $n$ (say, $n = 50$), this will completely exceed system resources!
Reducing Memory Usage

• In many cases, we need to perform some operation on each subset, but don't need to actually store those subsets.

• **Idea:** Generate each subset, process it, and then discard it.

• **Question:** How do we do this?
A Decision Tree

A?

{H?}

{I?} {H?

{I?} {H, I?

{I?} {A}

{I?} {A, I?

{I?} {A, H}

{I?} {A, H, I}
void exploreFrom(current state, decisions made) {
    if (all decisions have been made) {
        output the result of the decisions we’ve made;
    } else {
        for (each decision we can make) {
            exploreFrom(result of making that decision, decisions made + this decision);
        }
    }
}

void exploreAllTheThings(initial state) {
    exploreFrom(initial state, {});
}
You own a classy print shop. You’ve got a list of jobs you print. Each job requires some amount of time and has a hard deadline. Which jobs should you pick to maximize your profit?
Permutations

• A permutation of a sequence is a sequence with the same elements, though possibly in a different order.

• For example:
  • E Pluribus Unum
  • E Unum Pluribus
  • Pluribus E Unum
  • Pluribus Unum E
  • Unum E Pluribus
  • Unum Pluribus E
Generating Permutations

<table>
<thead>
<tr>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
<th>X_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_1</td>
<td>X_2</td>
<td>X_3</td>
<td>X_4</td>
</tr>
<tr>
<td>X_2</td>
<td>X_1</td>
<td>X_3</td>
<td>X_4</td>
</tr>
<tr>
<td>X_3</td>
<td>X_1</td>
<td>X_2</td>
<td>X_4</td>
</tr>
<tr>
<td>X_4</td>
<td>X_1</td>
<td>X_2</td>
<td>X_3</td>
</tr>
<tr>
<td>X_1</td>
<td>X_2</td>
<td>X_3</td>
<td>X_4</td>
</tr>
<tr>
<td>X_2</td>
<td>X_3</td>
<td>X_1</td>
<td>X_4</td>
</tr>
<tr>
<td>X_3</td>
<td>X_4</td>
<td>X_1</td>
<td>X_2</td>
</tr>
<tr>
<td>X_4</td>
<td>X_3</td>
<td>X_1</td>
<td>X_2</td>
</tr>
<tr>
<td>X_1</td>
<td>X_2</td>
<td>X_3</td>
<td>X_4</td>
</tr>
<tr>
<td>X_2</td>
<td>X_3</td>
<td>X_4</td>
<td>X_1</td>
</tr>
<tr>
<td>X_3</td>
<td>X_4</td>
<td>X_2</td>
<td>X_1</td>
</tr>
<tr>
<td>X_4</td>
<td>X_3</td>
<td>X_2</td>
<td>X_1</td>
</tr>
<tr>
<td>X_1</td>
<td>X_2</td>
<td>X_3</td>
<td>X_4</td>
</tr>
<tr>
<td>X_2</td>
<td>X_3</td>
<td>X_1</td>
<td>X_4</td>
</tr>
<tr>
<td>X_3</td>
<td>X_4</td>
<td>X_1</td>
<td>X_2</td>
</tr>
<tr>
<td>X_4</td>
<td>X_3</td>
<td>X_2</td>
<td>X_1</td>
</tr>
</tbody>
</table>
Generating Permutations

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>$X_2$</td>
<td>$X_3$</td>
<td>$X_4$</td>
<td></td>
</tr>
<tr>
<td>$X_1$</td>
<td>$X_2$</td>
<td>$X_3$</td>
<td>$X_4$</td>
<td></td>
</tr>
<tr>
<td>$X_2$</td>
<td>$X_1$</td>
<td>$X_3$</td>
<td>$X_4$</td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td>$X_1$</td>
<td>$X_2$</td>
<td>$X_4$</td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td>$X_2$</td>
<td>$X_1$</td>
<td>$X_4$</td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td>$X_4$</td>
<td>$X_1$</td>
<td>$X_2$</td>
<td></td>
</tr>
<tr>
<td>$X_3$</td>
<td>$X_4$</td>
<td>$X_2$</td>
<td>$X_1$</td>
<td></td>
</tr>
</tbody>
</table>
A Decision Tree
```java
void exploreFrom(current state, decisions made) {
    if (all decisions have been made) {
        output the result of the decisions we’ve made;
    } else {
        for (each decision we can make) {
            exploreFrom(result of making that decision, decisions made + this decision);
        }
    }
}

void exploreAllTheThings(initial state) {
    exploreFrom(initial state, {});
}
```
Your Action Items

• Start working on Assignment 3.
  
  • *Don’t put this one off!* It’s going to require some thought.

• Stop by YEAH Hours to get some help on how to get started on this assignment.

• Read Chapter 8, if you haven’t yet done so.

• Start reading Chapter 9 in preparation for Wednesday’s lecture.
Next Time

• **Generating Combinations**
  • How do we find the best group of people to pick for a task?

• **Recursive Backtracking**
  • How do we determine whether something is feasible?