Thinking Recursively

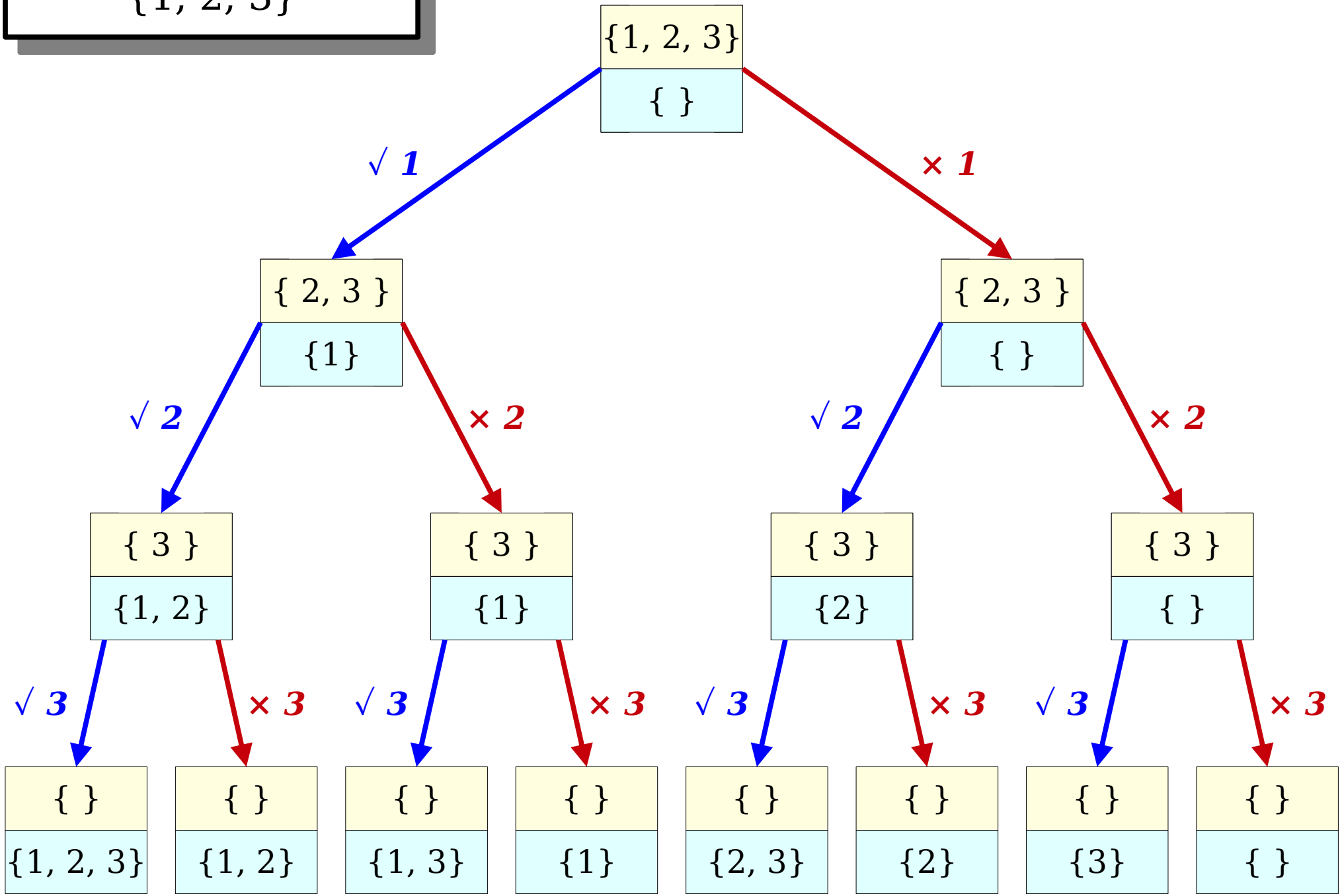
Part III

Outline for Today

- ***Iteration + Recursion***
  - Combining two techniques together.
- ***Enumerating Permutations***
  - What order should we do things?
- ***Enumeration, Generally***
  - How to think about enumeration problems.

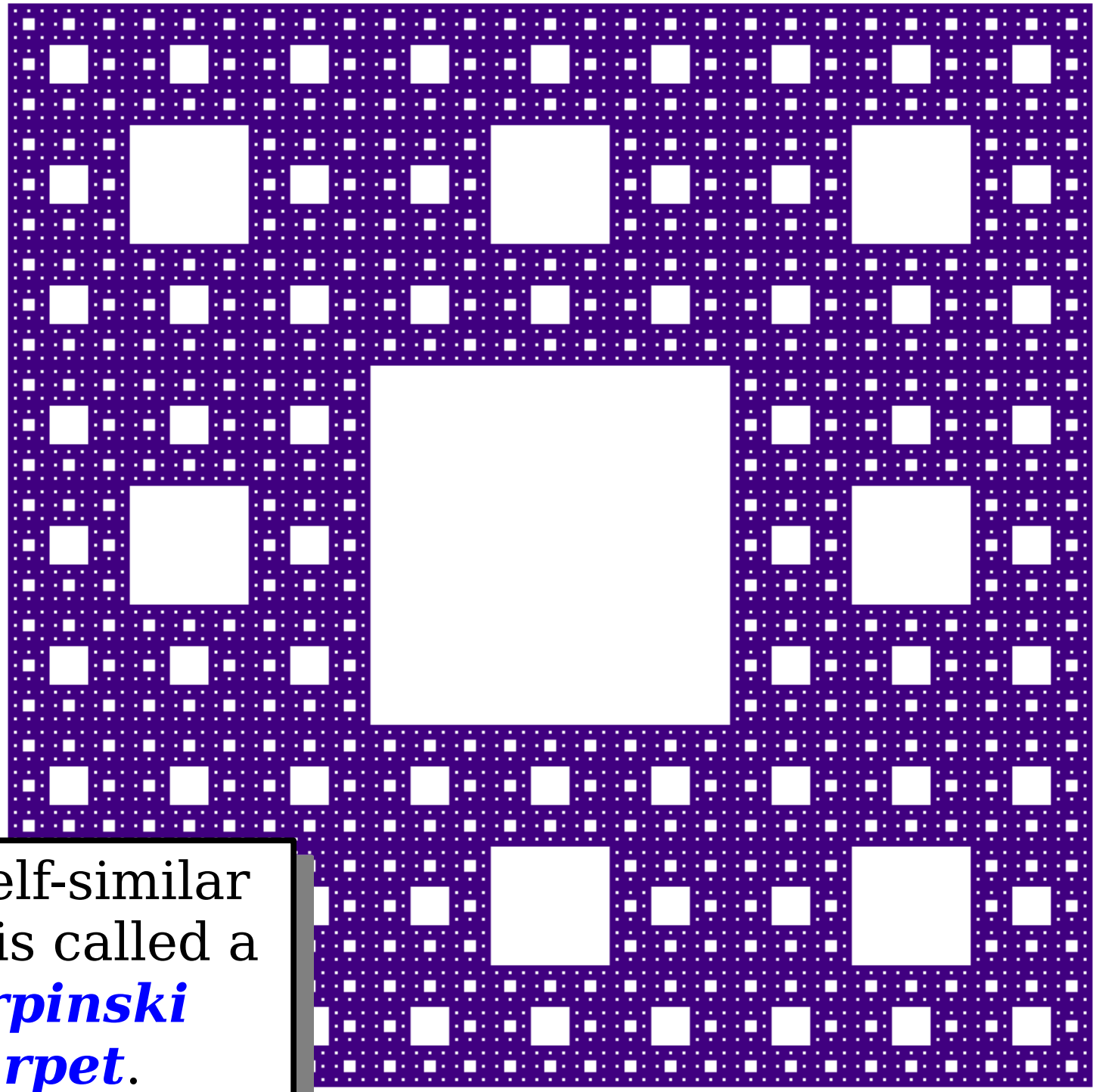
Recap from Last Time

List all ***subsets*** of {1, 2, 3}

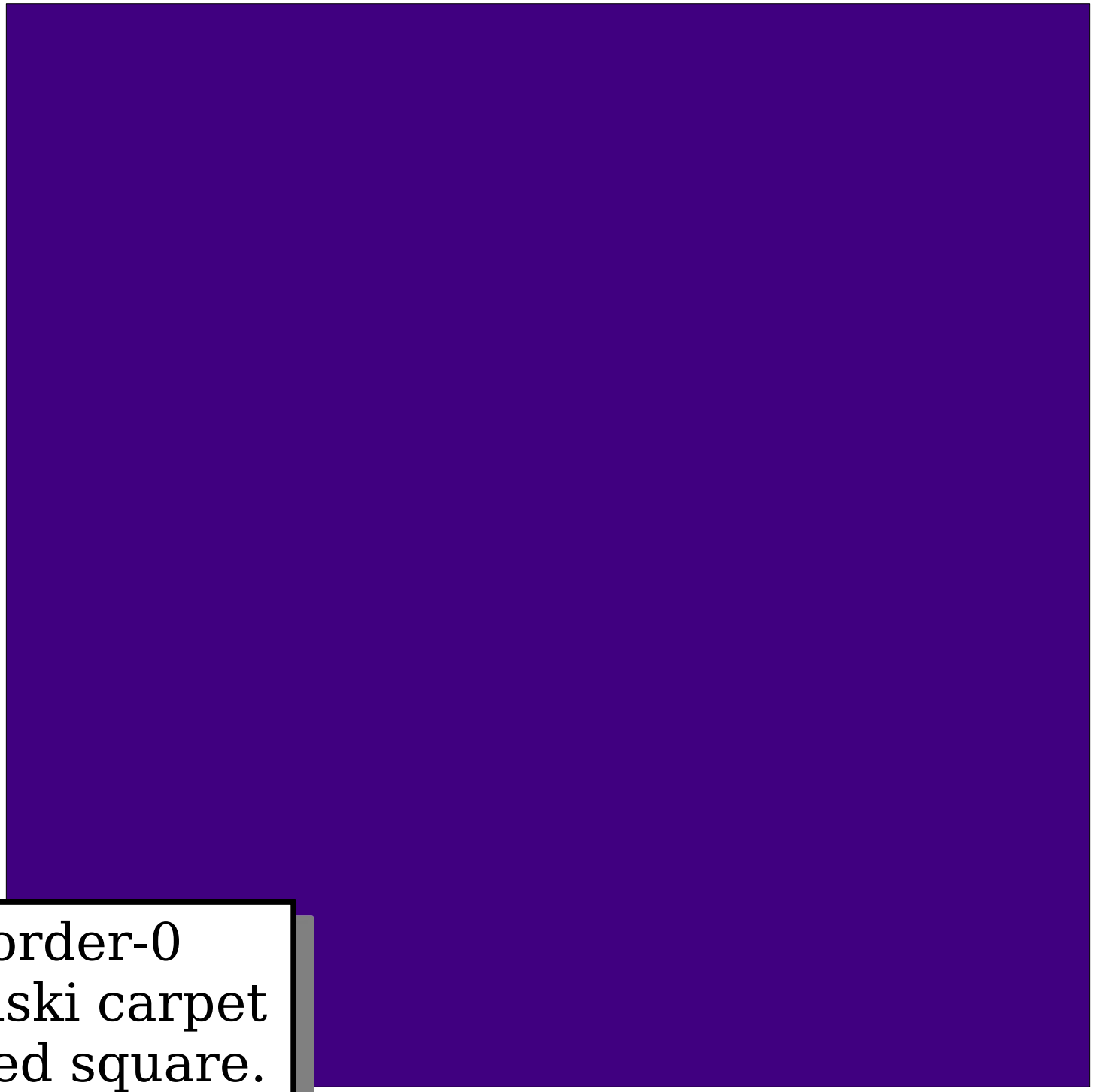


New Stuff!

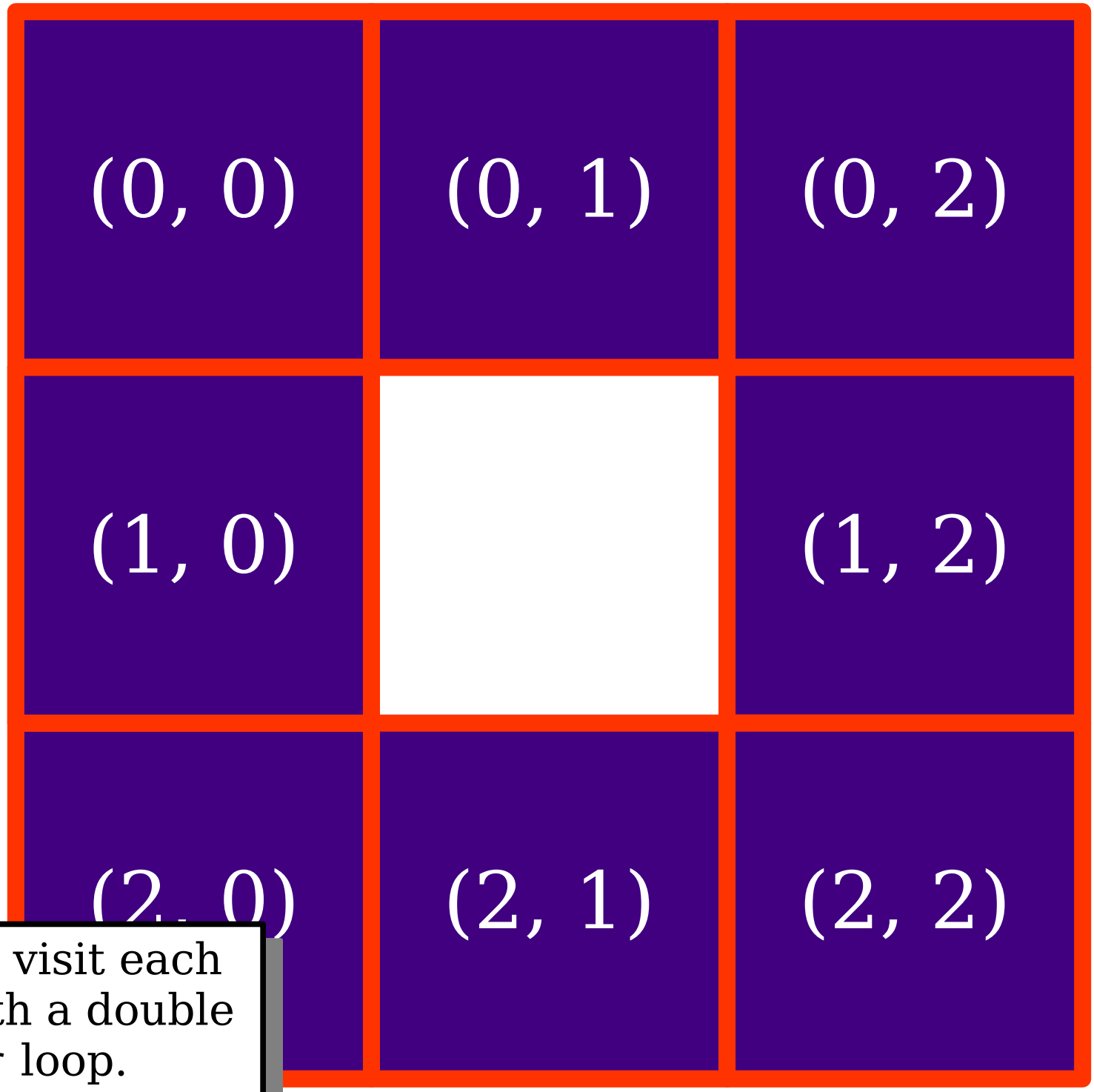
More On Self-Similarity



This self-similar shape is called a ***Sierpinski carpet***.



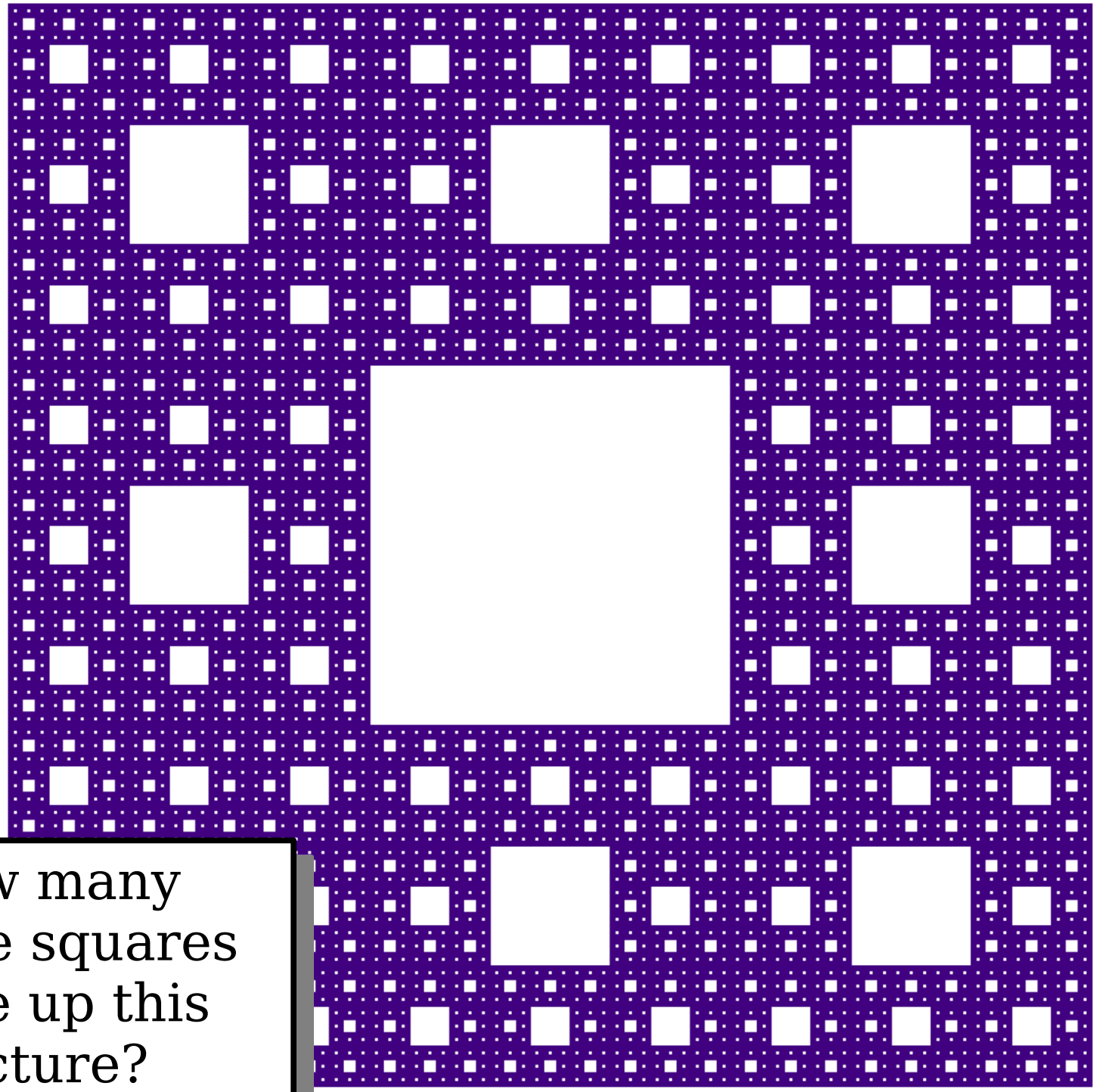
An order-0 Sierpinski carpet is a filled square.



We can visit each spot with a double `for` loop.

Iteration + Recursion

- It's completely reasonable to mix iteration and recursion in the same function.
- Here, we're firing off eight recursive calls, and the easiest way to do that is with a double `for` loop.
- Recursion doesn't mean "the absence of iteration." It just means "solving a problem by solving smaller copies of that same problem."



How many purple squares make up this picture?

Time-Out for Announcements!

Assignment 3

- Assignment 2 was due today at 1PM.
  - You can use late days to extend the deadline by 24 or 48 hours. Remember that you only get four late days to use over the quarter.
- Assignment 3 (***A Visit to Recursia***) goes out today. It's due next Friday at 1:00PM.
  - Play around with recursion and recursive problem-solving!
  - Use recursion to generate mountain ranges!
  - Generate all words in a fictional language!
  - Make a neat building with recursion!
- YEAH hours are today, 4:30PM – 5:30PM, in Hewlett 101. Purely optional, but highly recommended.

Recursive Drawing Contest

- Our (optional, just for fun) Recursive Drawing contest ends on Monday at 1PM.
- If you're interested in participating, visit ***http://recursivedrawing.com/***, draw something, and post it to EdStem.
- We're very impressed with the submissions you've made so far! If you haven't yet done so, go check them out.

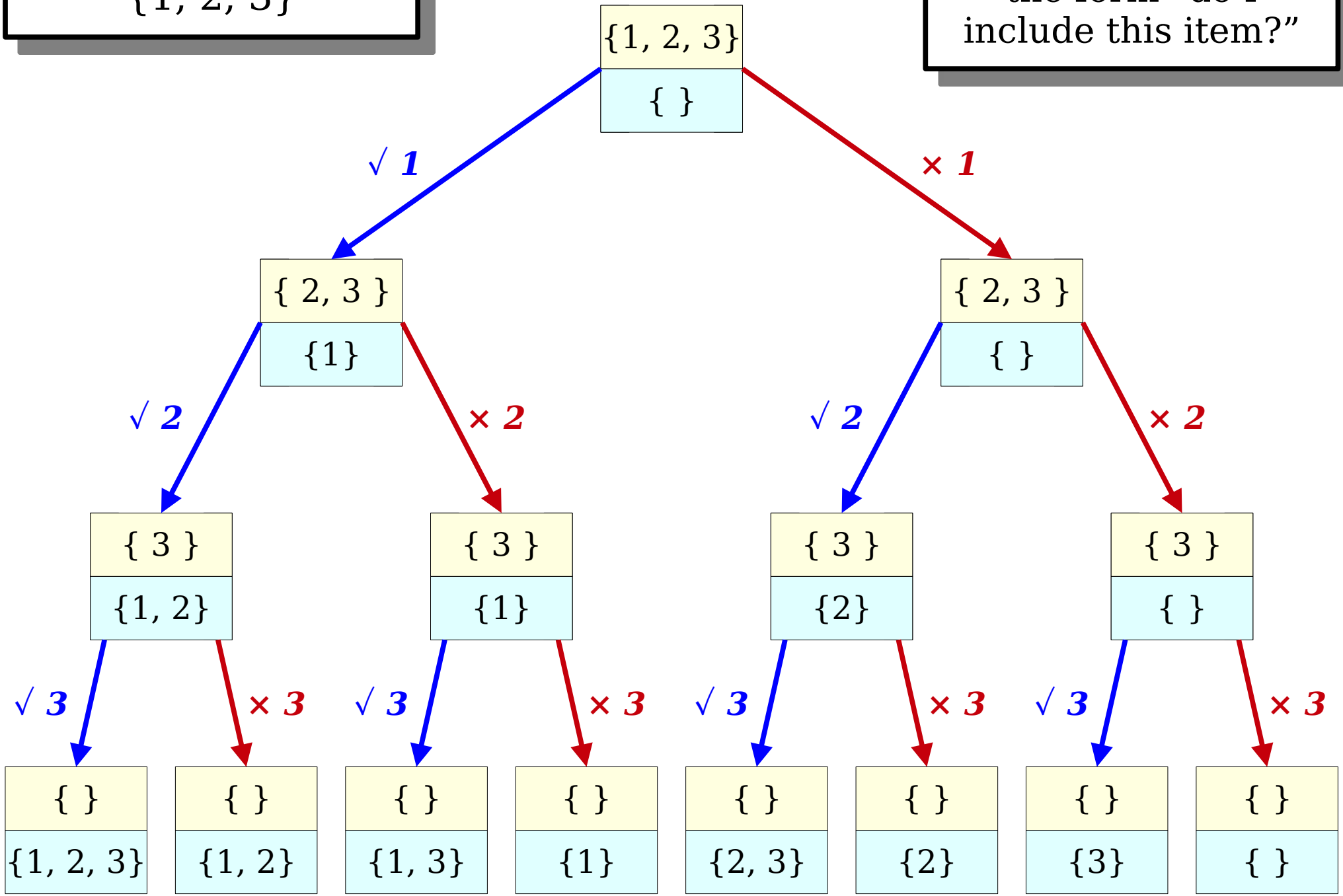
(The Curtain Rises on Act II)

Enumerating Permutations

A ***permutation*** is a rearrangement of the elements of a sequence.

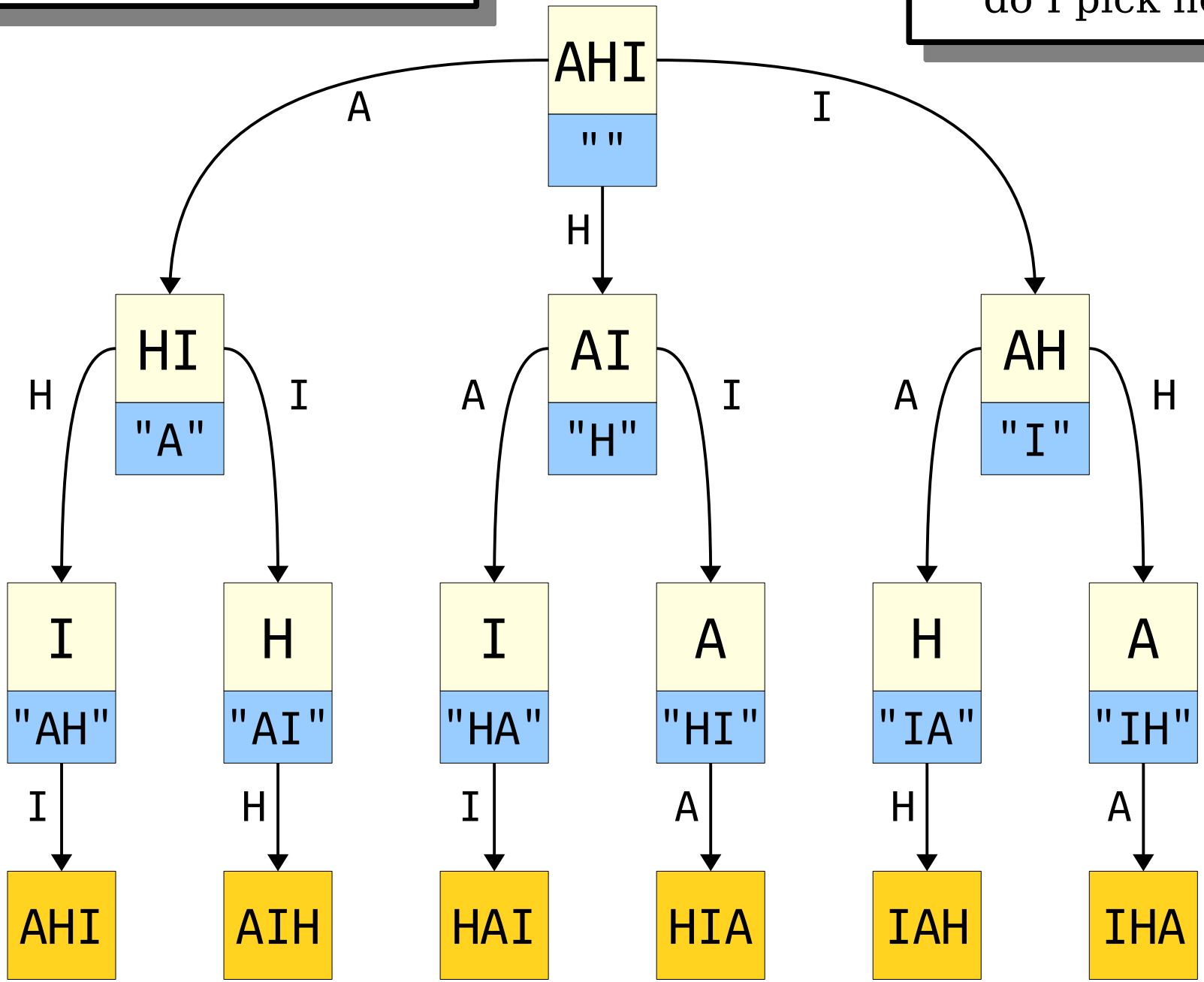
List all ***subsets*** of {1, 2, 3}

Each decision is of the form “do I include this item?”



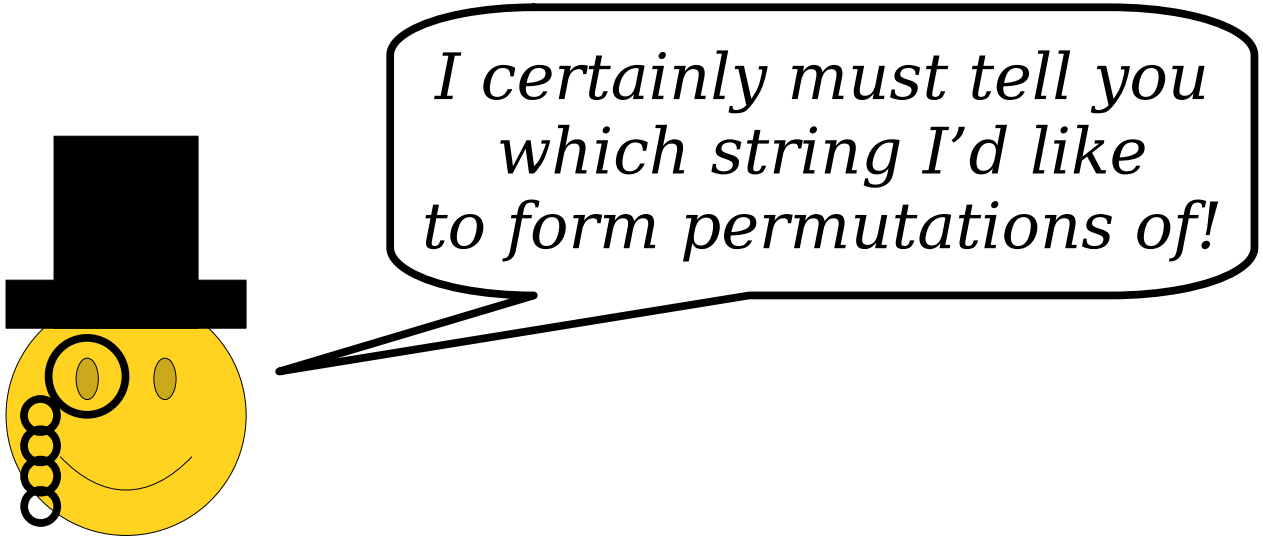
List all ***permutations*** of {A, H, I}

Each decision is of the form “which item do I pick next?”

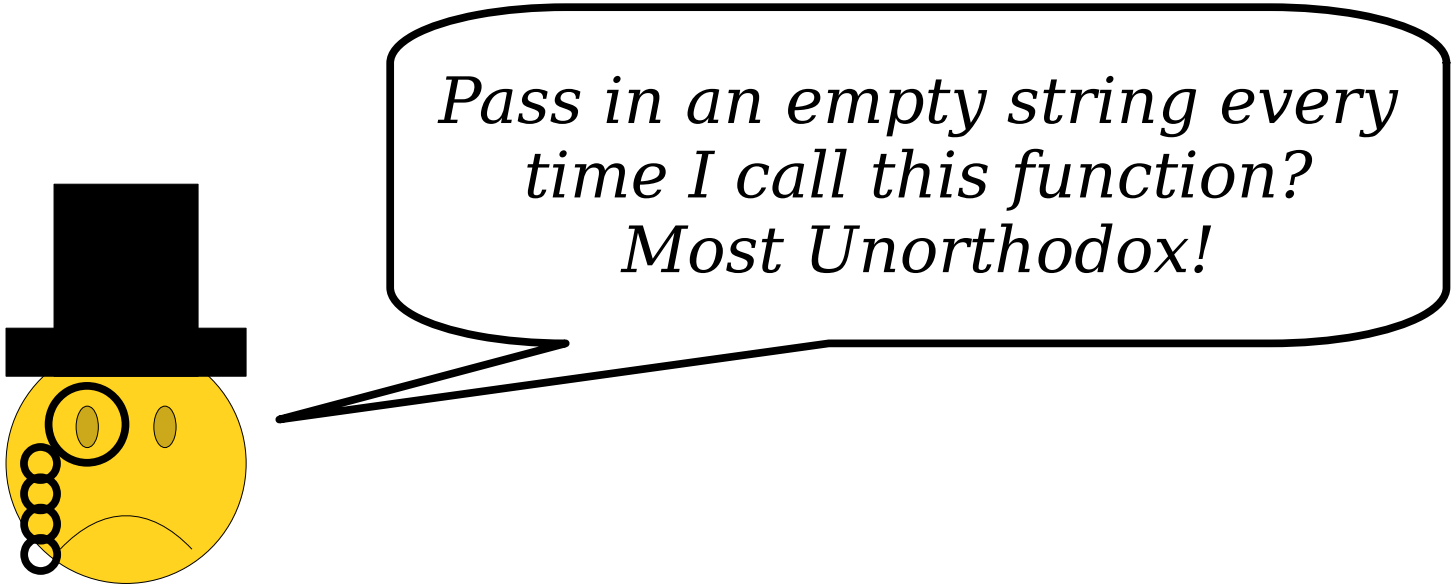


A Question of Parameters

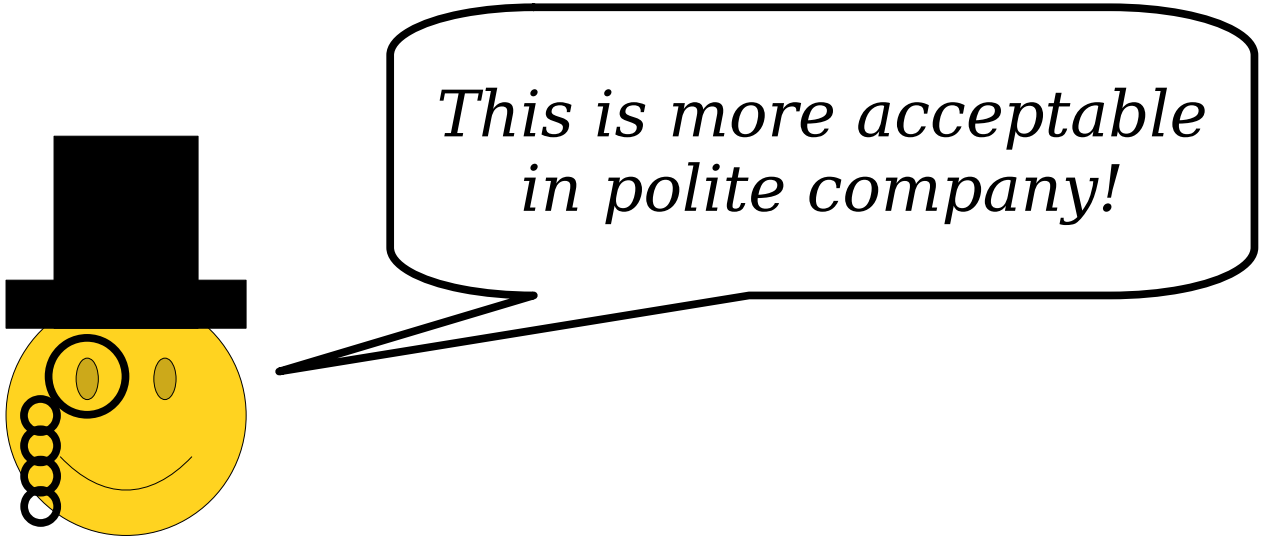
```
listPermutationsOf("AHI", "");
```



```
listPermutationsOf("AHI", "");
```

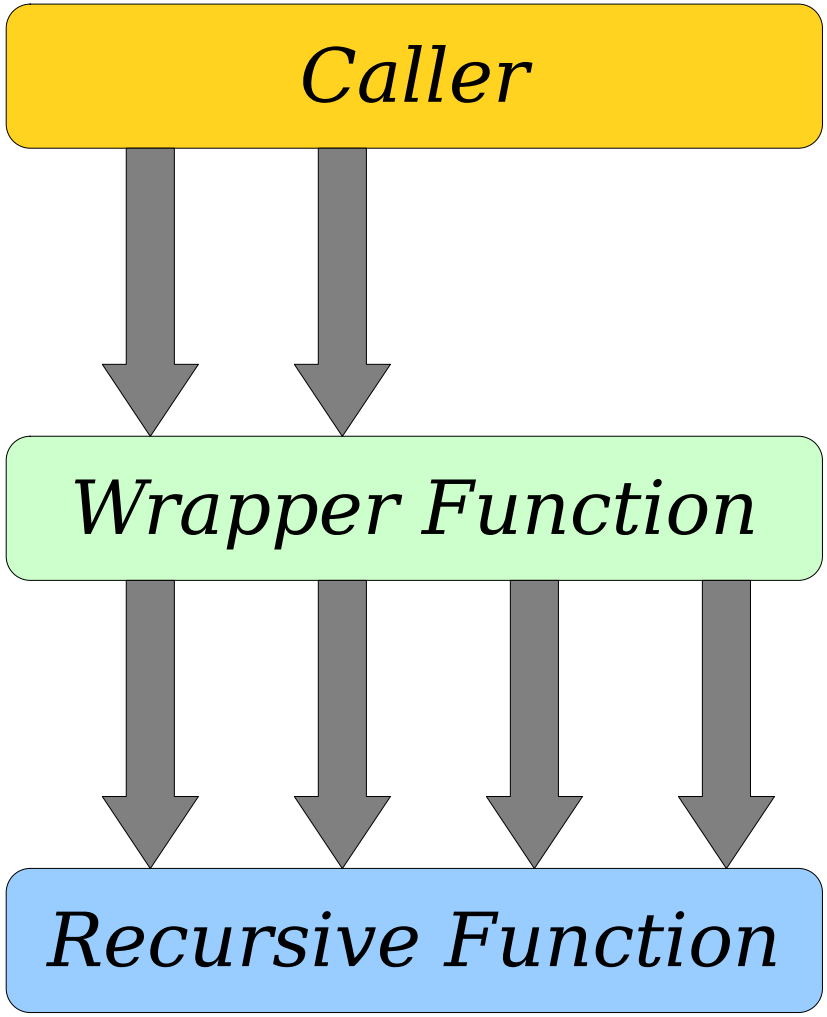


```
listPermutationsOf("AHI");
```



Wrapper Functions

- Some recursive functions need extra arguments as part of an implementation detail.
  - In our case, the string of letters ordered so far is not something we want to expose.
- A ***wrapper function*** is a function that does some initial prep work, then fires off a recursive call with the right arguments.



Storing Permutations

```
Set<string> permutationsOf(const string& str);
```

Base Case: No decisions remain.

```
ResultType exploreRec(decisions remaining,
                      decisions already made) {
  if (no decisions remain) {
    return decisions made;
  } else {
    ResultType result;
    for (each possible next choice) {
      result += exploreRec(all remaining decisions,
                           decisions made + that choice);
    }
    return result;
  }
}
```

Recursive Case: Try all options for the next decision.

```
ResultType exploreAllTheThings(initial state) {
  return exploreRec(initial state, no decisions made);
}
```

Summary for Today

- Recursion and iteration aren't mutually exclusive and are frequently combined.
- We can enumerate subsets using a decision tree of "do I pick this?" We can enumerate permutations using a decision tree of "what do I pick next?"
- Recursive functions can both print all objects of some type and return all objects of some type.

Your Action Items

- ***Read Chapter 8***
  - There are so many goodies there, and it's a great way to complement what we're discussing here.
- ***Work on Assignment 3***
  - Aim to complete the Flag of Recursia and Mountains of Recursia by Monday, and aim to start working on Speaking Recursian.

Next Time

- ***Enumerating Combinations***
  - Can you build the Dream Team?
- ***Recursive Backtracking***
  - Finding a needle in a haystack.
- ***The Great Shrinkable Word Problem***
  - A fun language exercise with a cute backstory.