Thinking Recursively

Part III

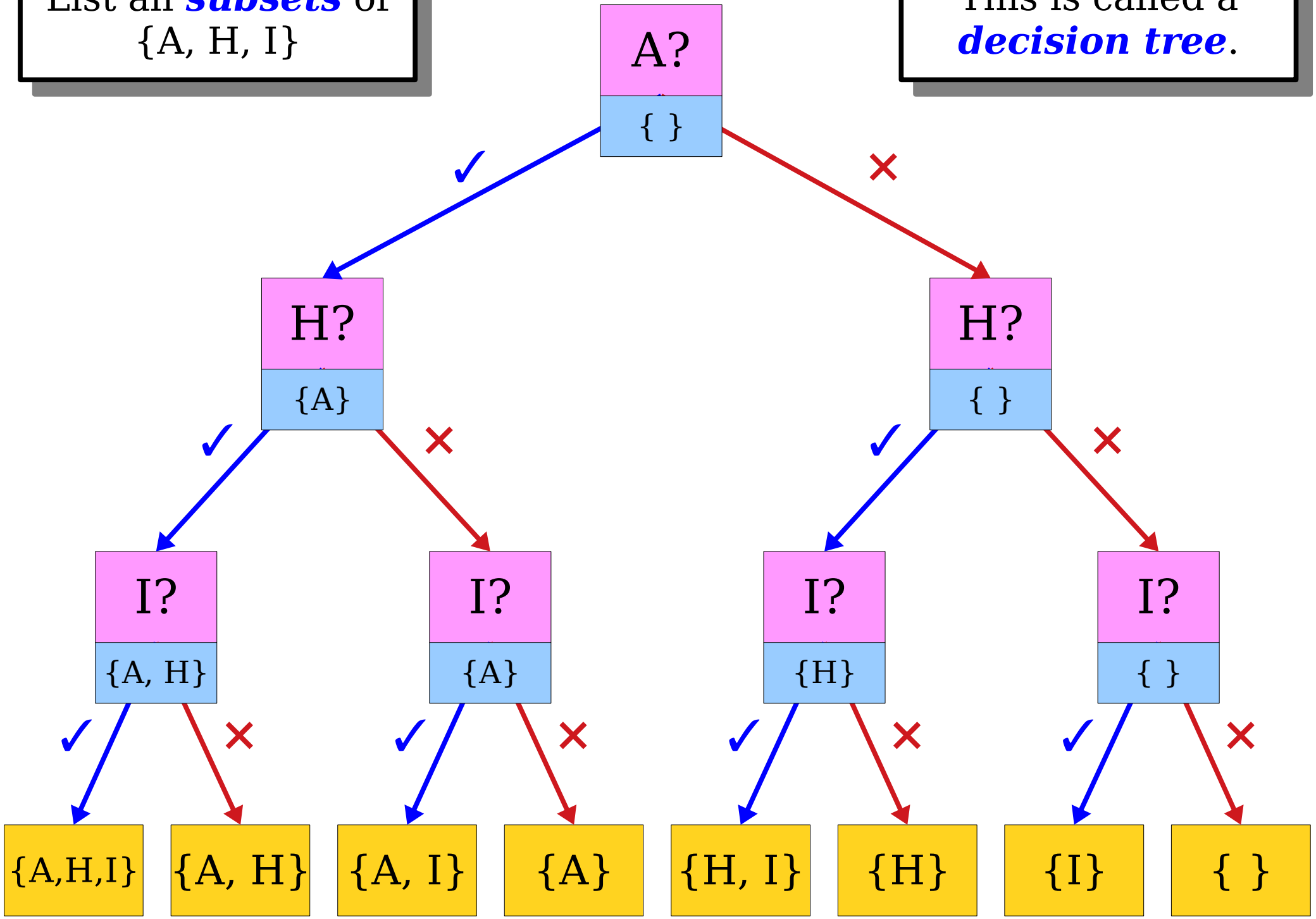
Outline for Today

- ***Recap from Last Time***
  - Where are we, again?
- ***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 {A, H, I}

This is called a ***decision tree***.



Base Case:
No decisions remain.

Decisions yet to be made

```
void listSubsetsRec(const HashSet<int>& elems,
                    const HashSet<int>& chosen) {

  if (elems.isEmpty()) {
    cout << chosen << endl;
  } else {
    int elem = elems.first();
    HashSet<int> remaining = elems - elem;

    /* Option 1: Include this element. */
    listSubsetsRec(remaining, chosen + elem);

    /* Option 2: Exclude this element. */
    listSubsetsRec(remaining, chosen);
  }
}
```

Decisions already made

Recursive Case:
Try all options for the next decision.

Base Case:
No decisions remain.

Decisions yet to be made

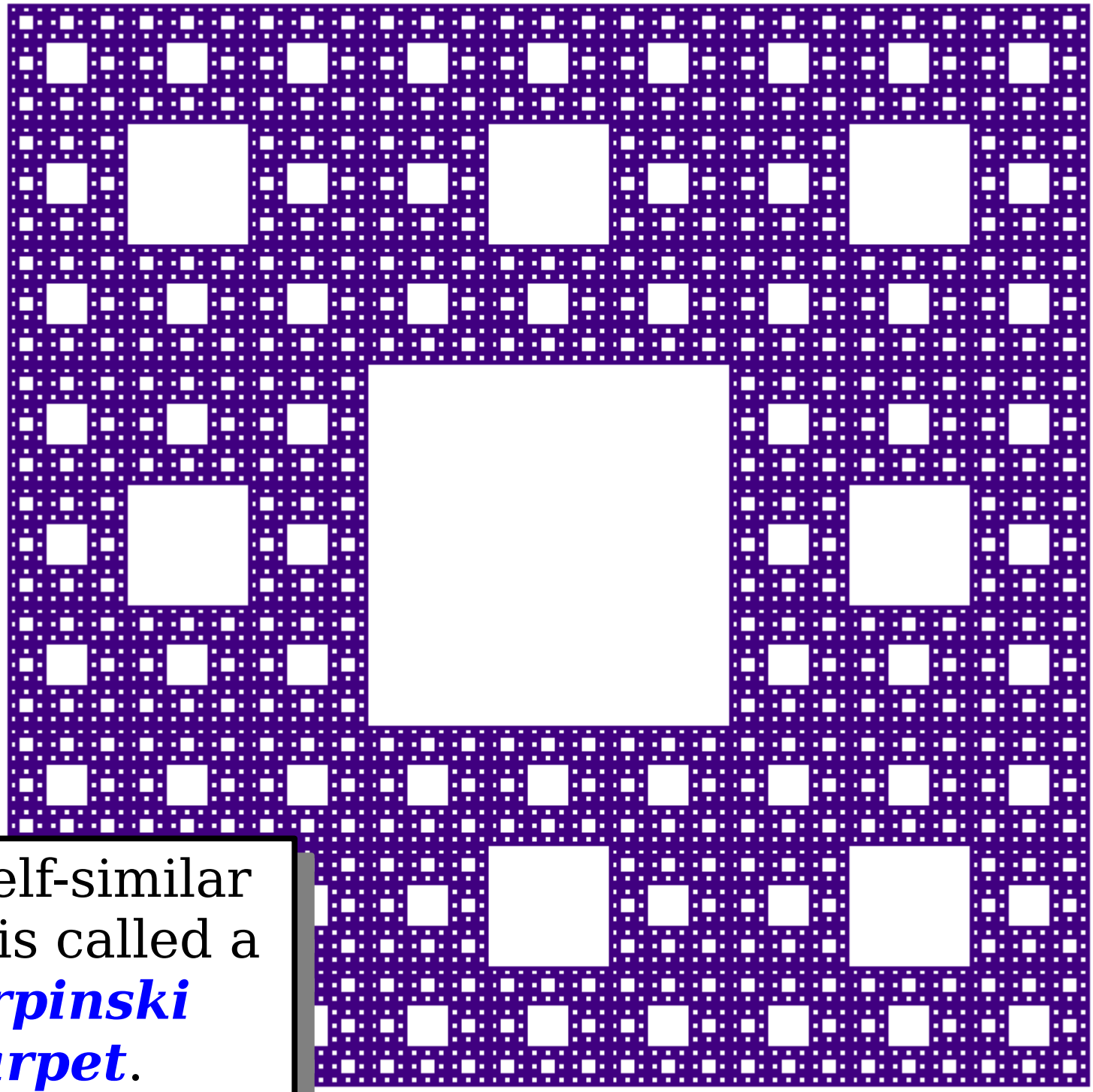
```
HashSet<string> subsetsRec(const string& str,
                           const string& chosen) {

  if (str == "") {
    return { chosen };
  } else {
    string remaining = str.substr(1);

    /* Either include the first character, or don't. */
    return subsetsRec(remaining, chosen + str[0]) +
           subsetsRec(remaining, chosen);

  }
}
```

Decisions already made

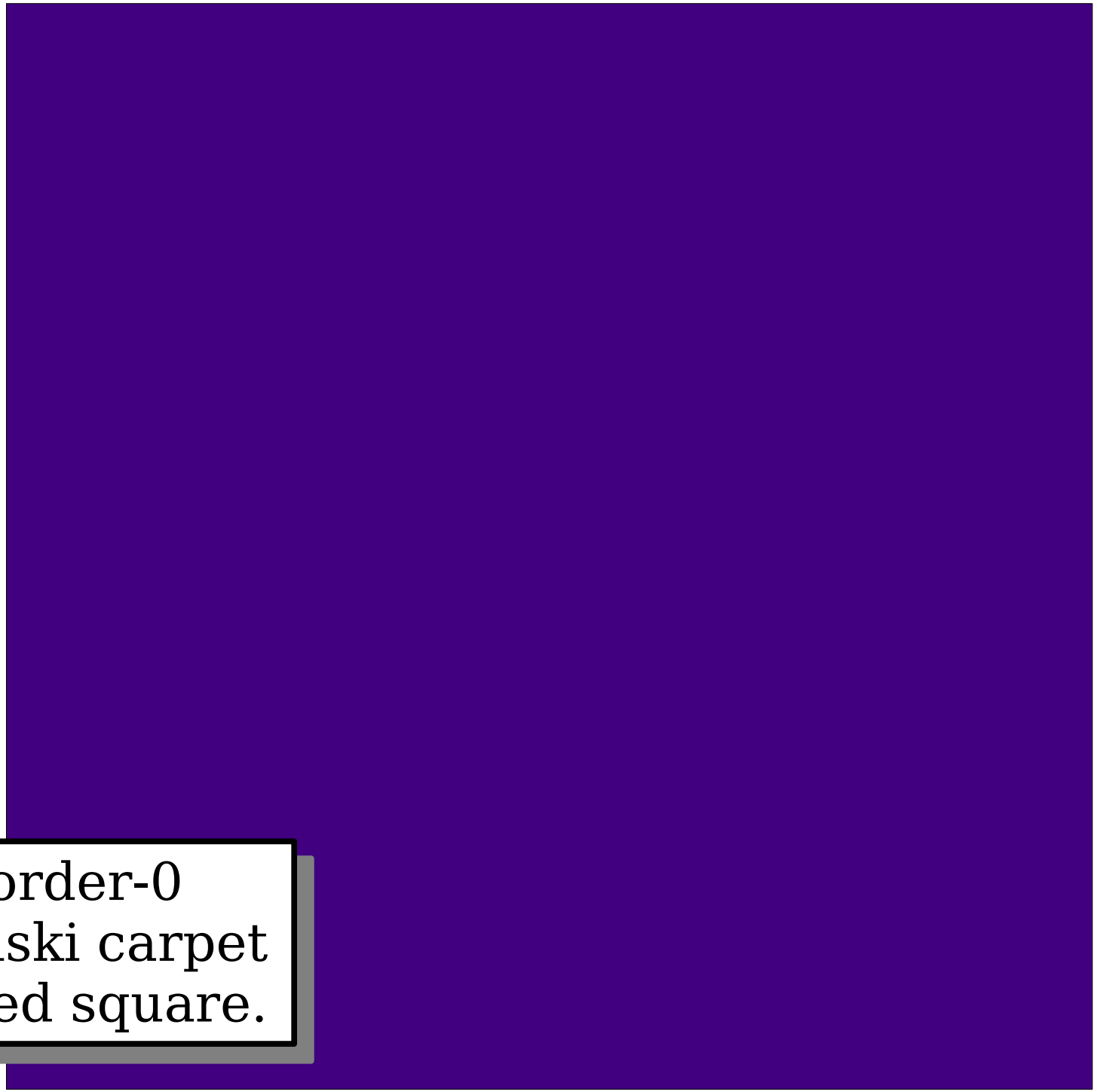
Recursive Case:
Try all options for the next decision.

New Stuff!

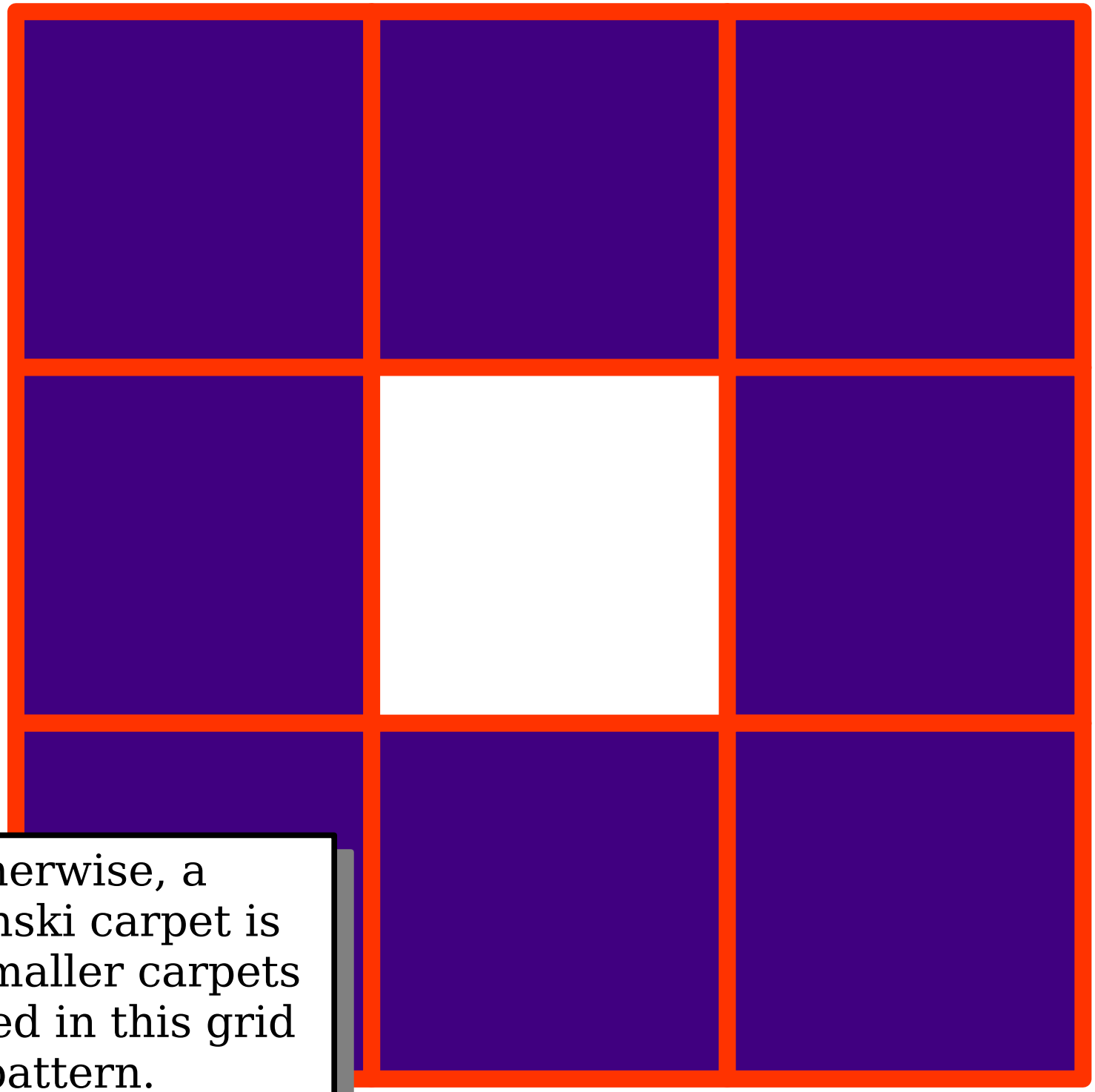
More On Self-Similarity



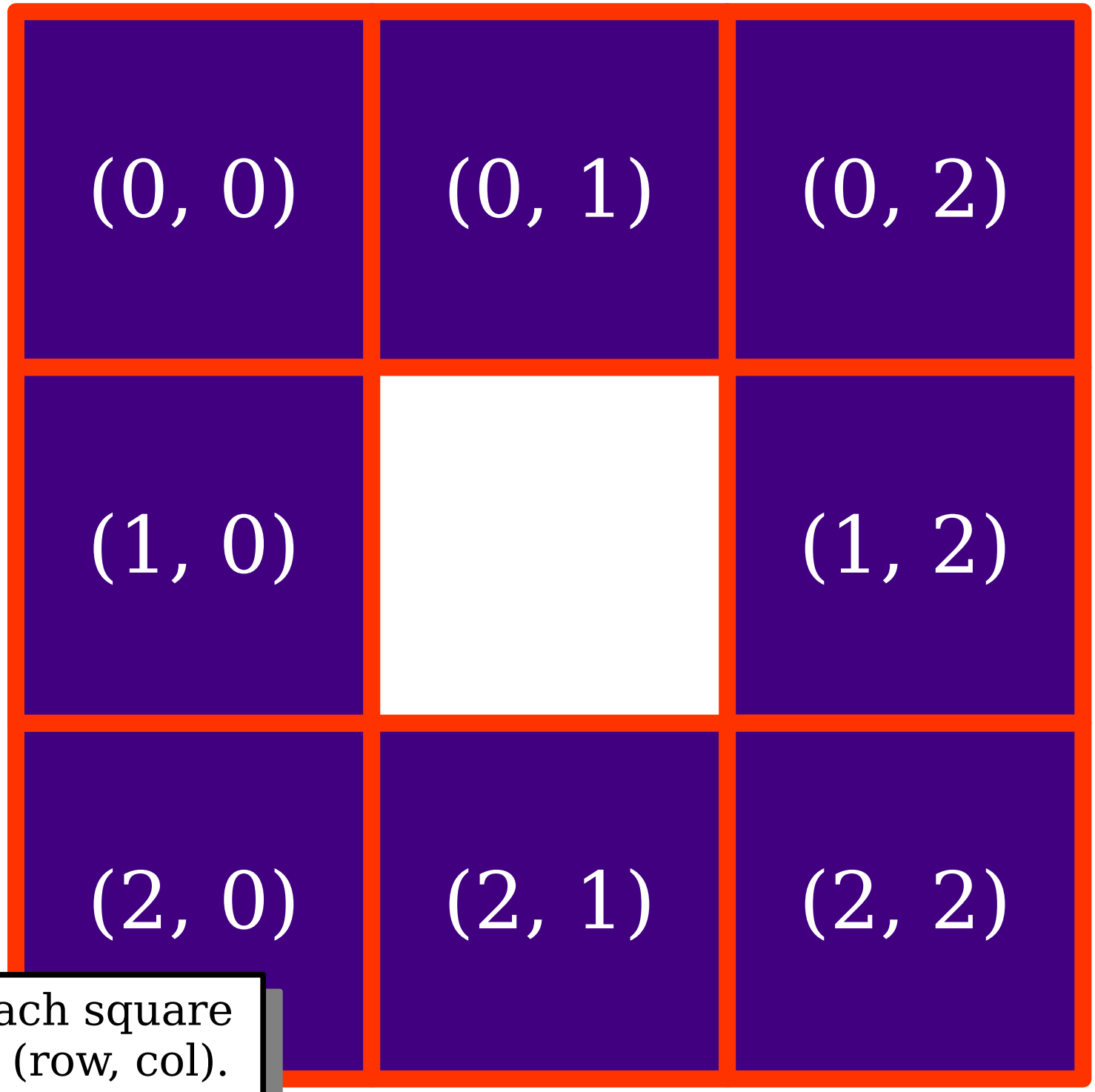
This self-similar shape is called a ***Sierpinski carpet***.



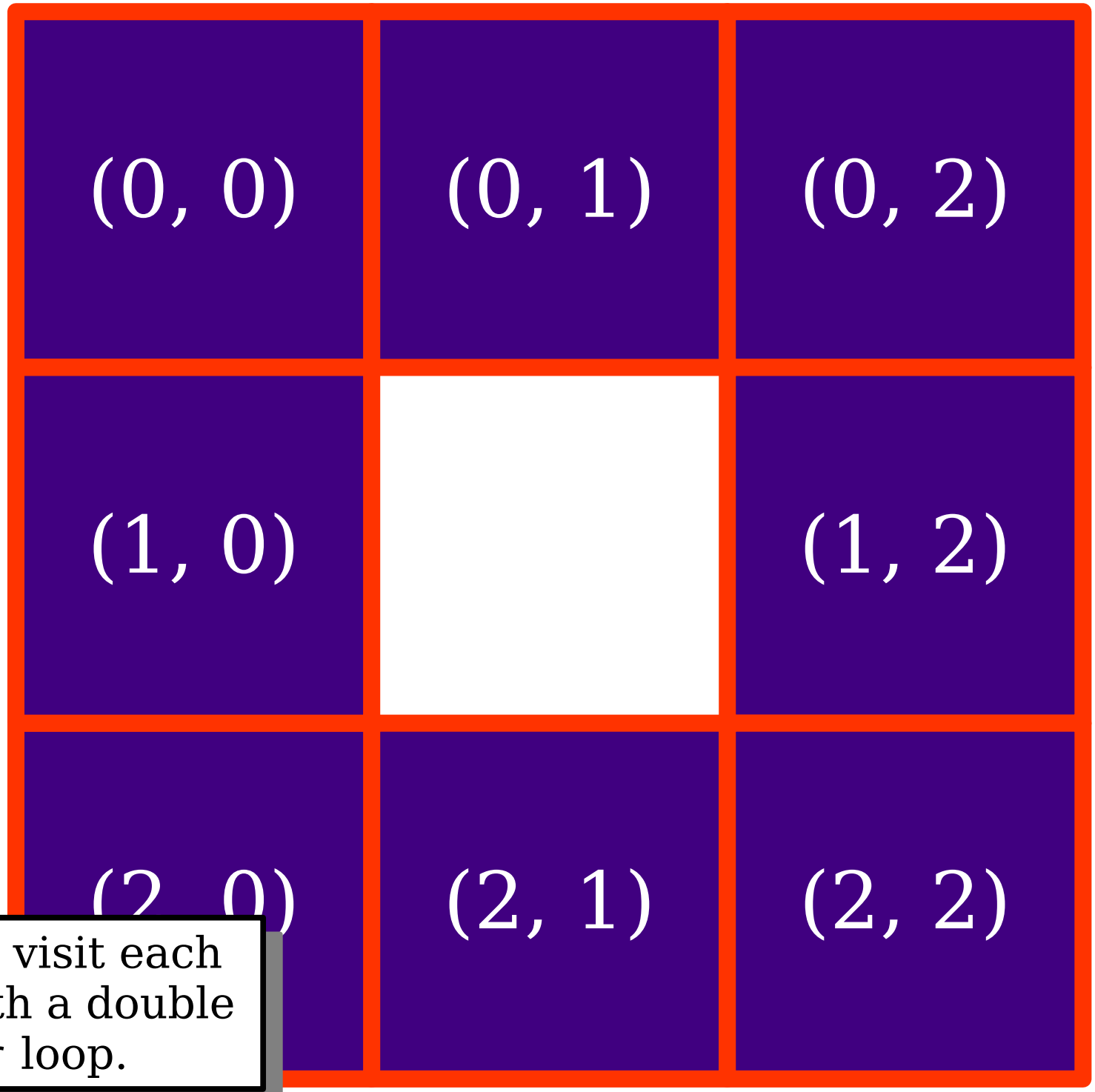
An order-0 Sierpinski carpet is a filled square.



Otherwise, a Sierpinski carpet is eight smaller carpets arranged in this grid pattern.



Label each square with its (row, col).



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.”

(And, just for fun…)

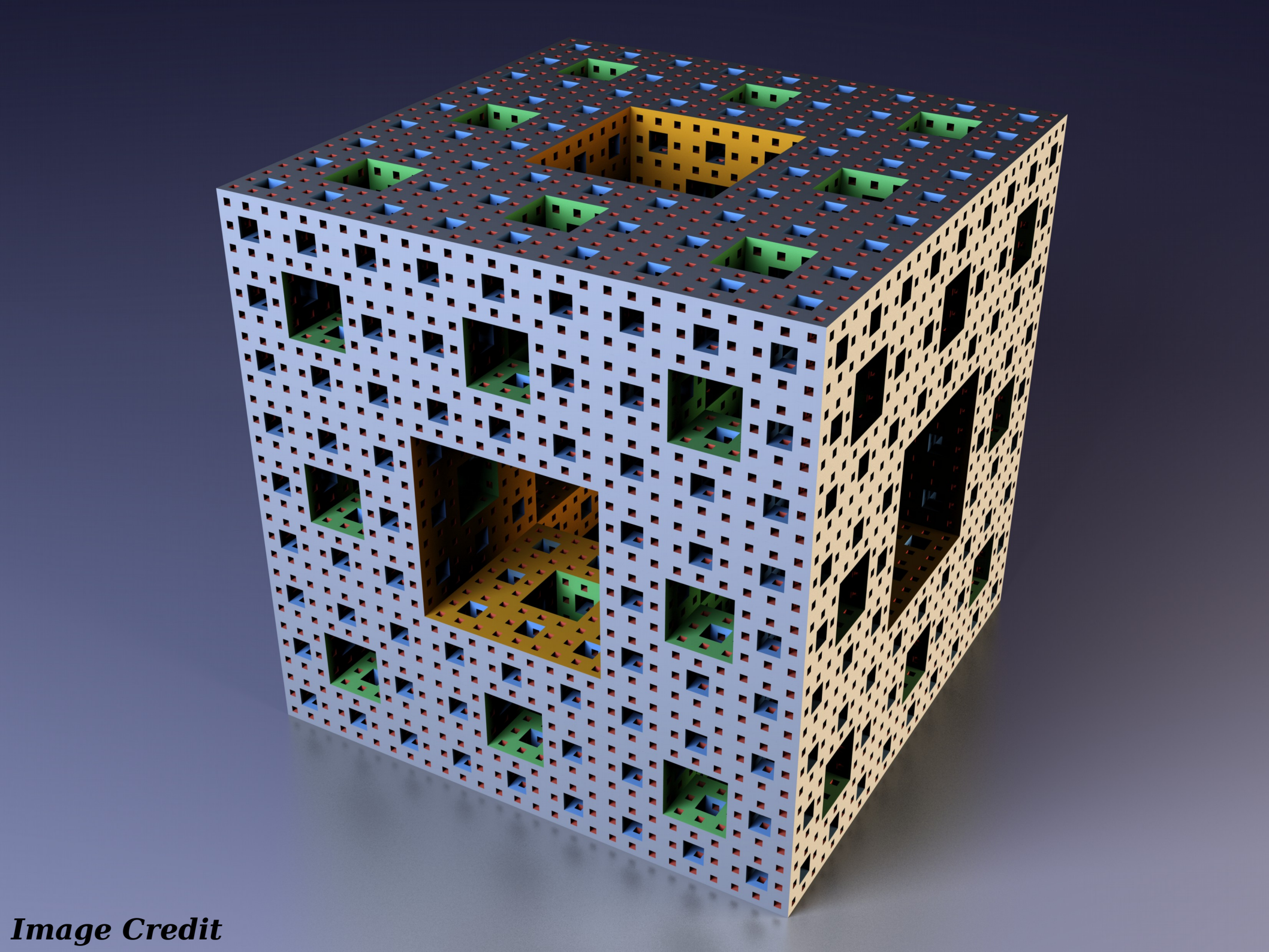


Image Credit

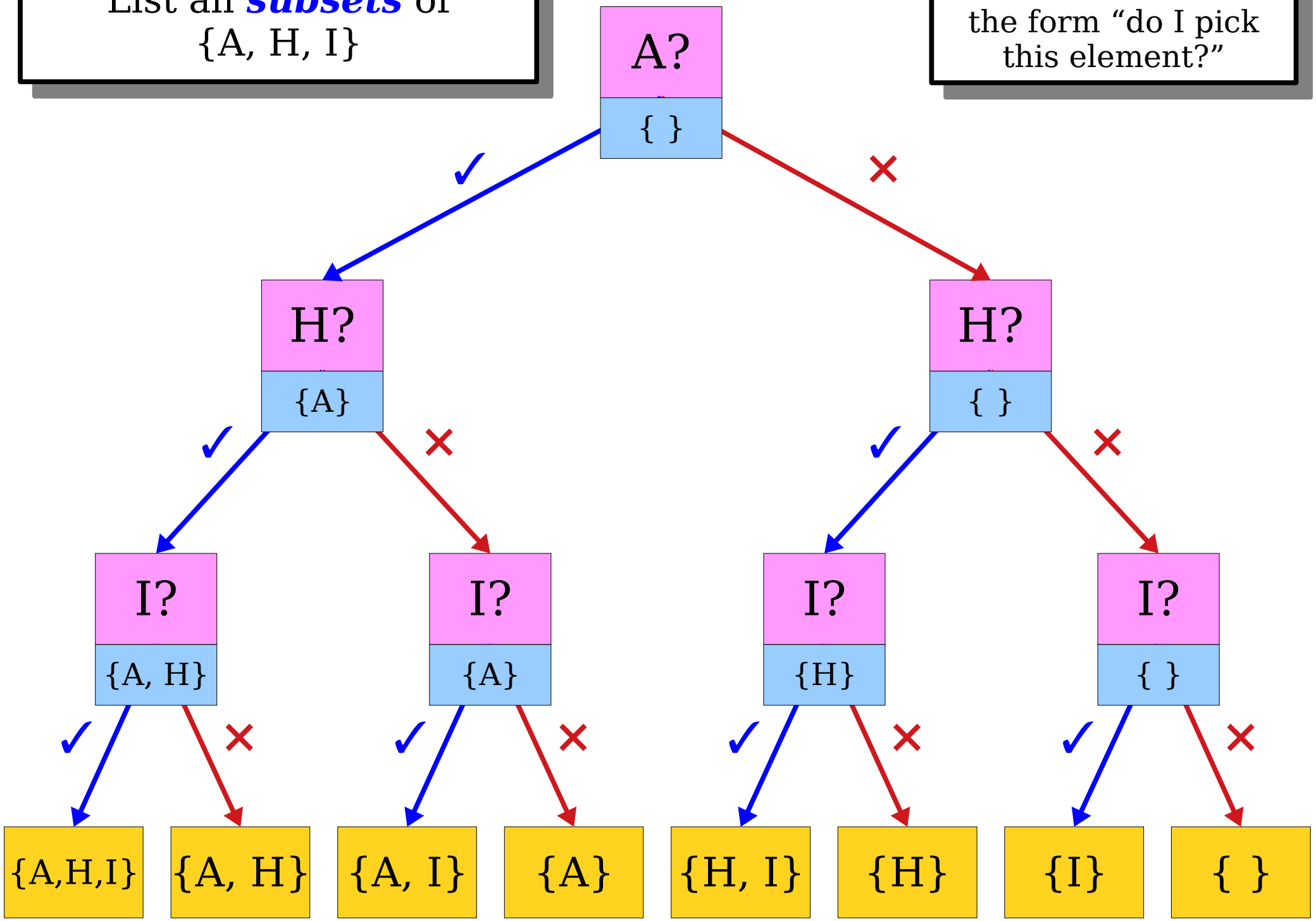
Enumerating Permutations

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

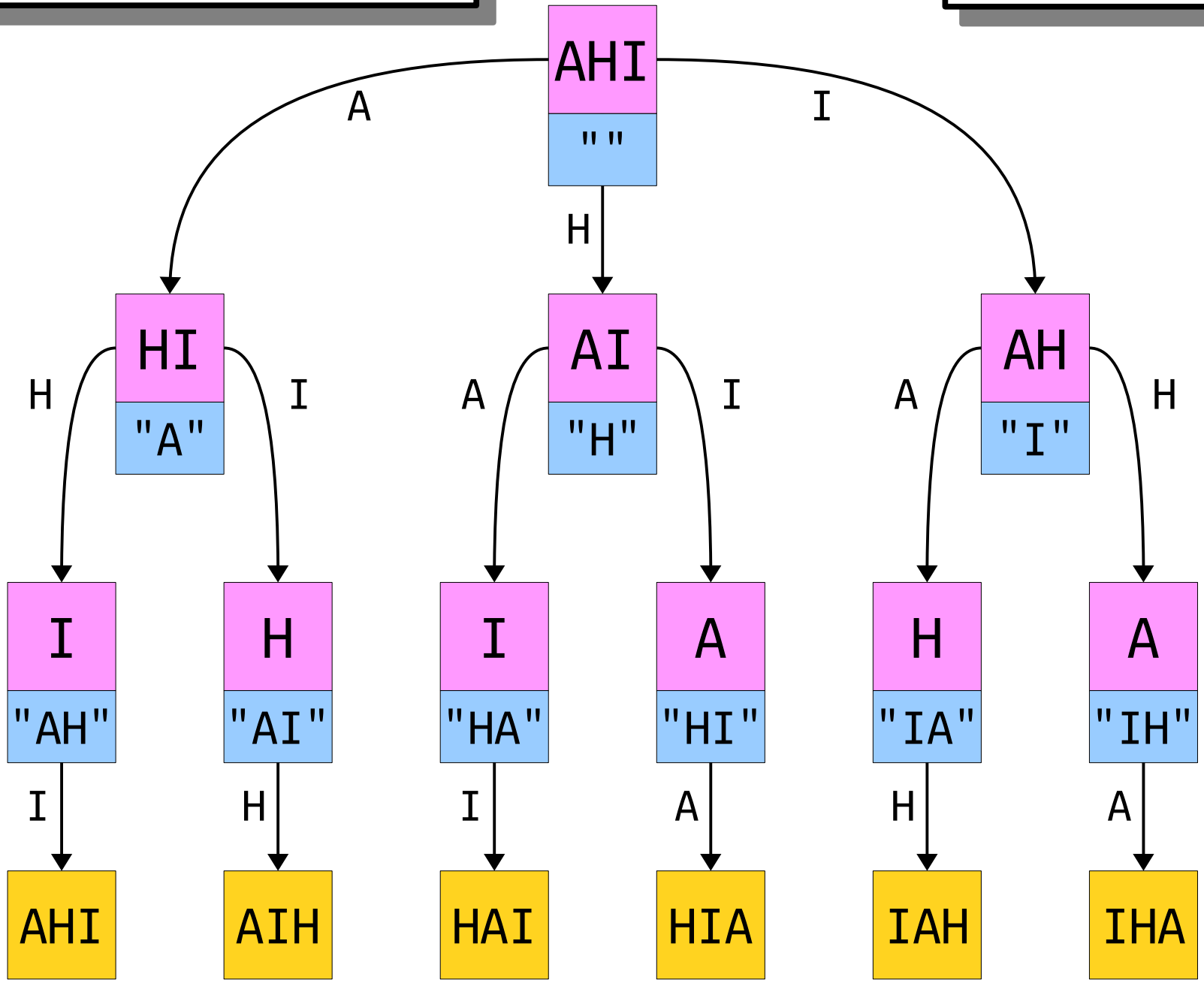
List all ***subsets*** of {A, H, I}

Each decision is of the form “do I pick this element?”



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

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



Base Case:
No decisions remain.

Decisions yet to be made

```
void listPermutationsRec(const string& str,
                         const string& chosen) {

  if (str == "") {
    cout << chosen << endl;
  } else {
    /* Try all options of what's next. */
    for (int i = 0; i < str.size(); i++) {
      char ch = str[i];
      string remaining = str.substr(0, i) +
                         str.substr(i + 1);
      listPermutationsRec(remaining, chosen + ch);
  }
}
```

Decisions already made

Recursive Case:
Try all options for the next decision.

Base Case:
No decisions remain.

Decisions yet to be made

```
void listSubsetsRec(const HashSet<int>& elems,
                    const HashSet<int>& chosen) {

  if (elems.isEmpty()) {
    cout << chosen << endl;
  } else {
    int elem = elems.first();
    HashSet<int> remaining = elems - elem;

    /* Option 1: Include this element. */
    listSubsetsRec(remaining, chosen + elem);

    /* Option 2: Exclude this element. */
    listSubsetsRec(remaining, chosen);
  }
}
```

Decisions already made

Recursive Case:
Try all options for the next decision.

Base Case: No decisions remain.

Decisions yet to be made

```
void exploreRec(decisions remaining,
                decisions already made) {

  if (no decisions remain) {
    process decisions made;
  } else {
    for (each possible next choice) {
      exploreRec(all remaining decisions,
                 decisions made + that choice);
    }
  }
}
```

Decisions already made

Recursive Case: Try all options for the next decision.

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

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***
  - If you're following our recommended timetable, you should have completed the Sierpinski Triangle and Human Pyramids and have started on What Are YOU Doing?
  - Aim to complete What Are YOU Doing? and to start Shift Scheduling by next time.

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.