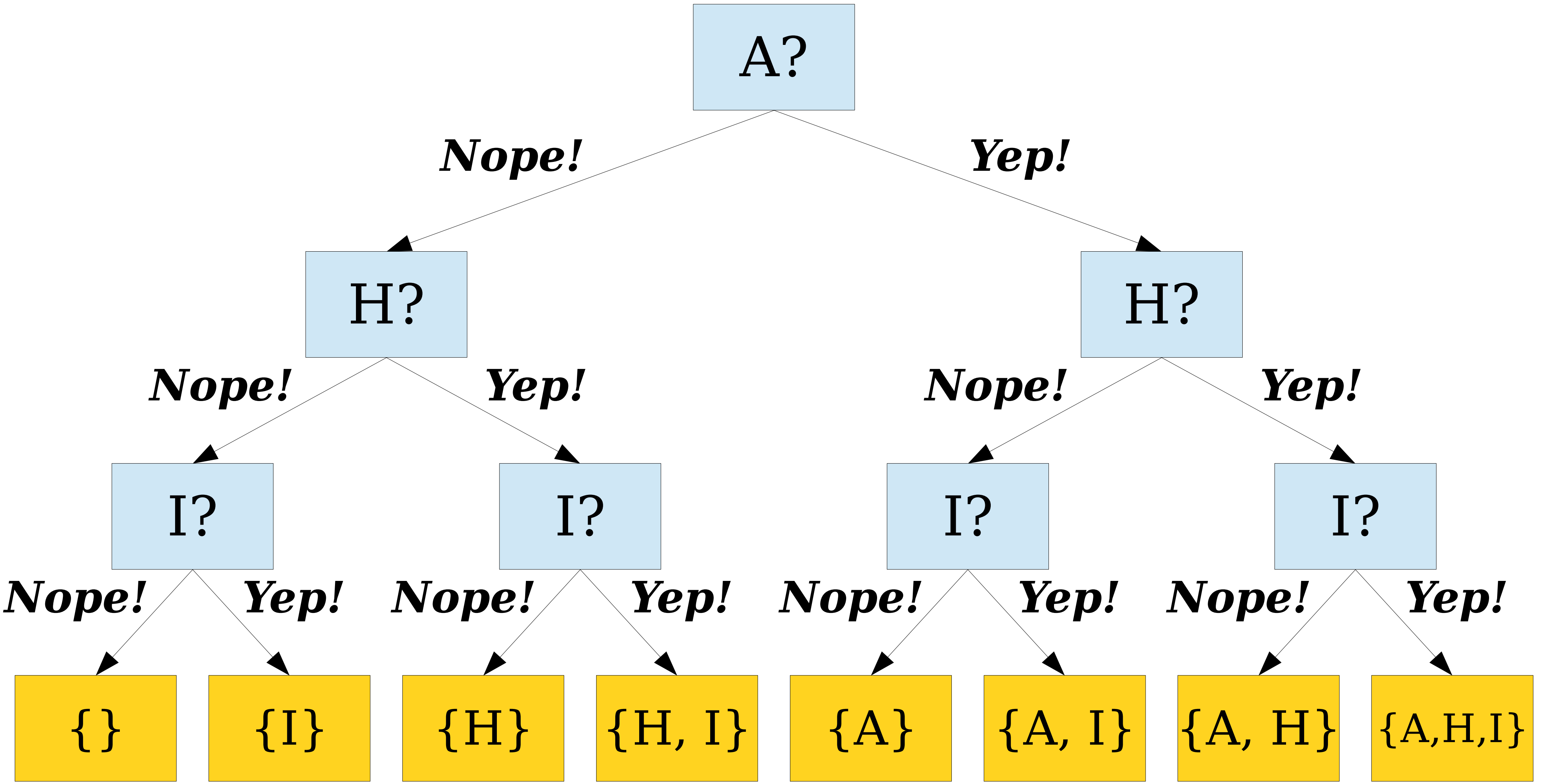
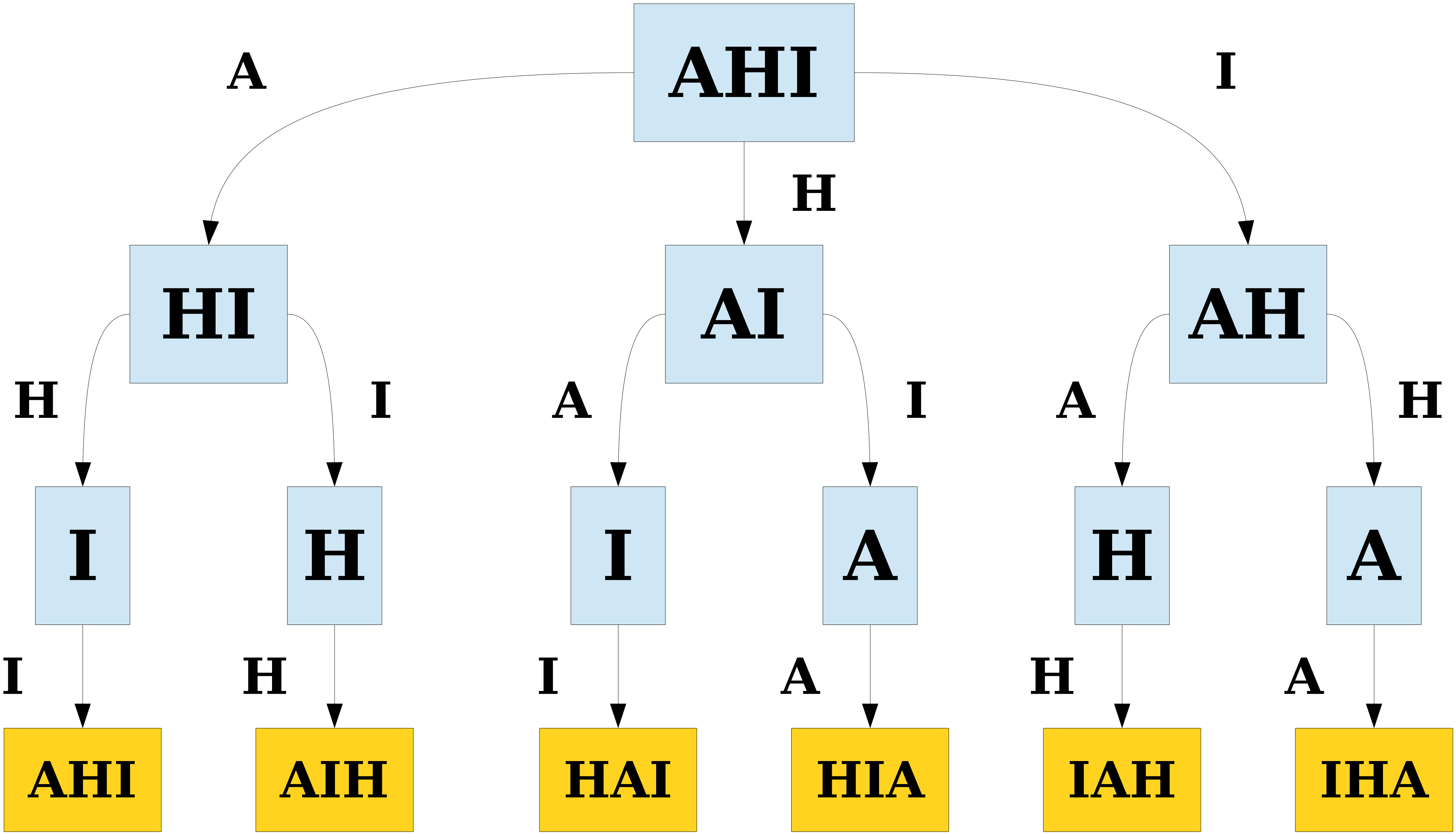
Thinking Recursively

Part IV

A Decision Tree



A Decision Tree



The Template

The Present

The Past

```
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, {});
}
```

The Future!

You need to pick 11 people to serve as starters on your soccer (football) team.

You have a good way of evaluating, roughly speaking, how any given team of 11 players will get along.

How do you decide which 11 players to pick?

Generating Combinations

- Suppose that we want to find every way to choose exactly ***one*** element from a set.
- We could do something like this:

```
for (int x: mySet) {
    cout << x << endl;
}
```

Generating Combinations

- Suppose that we want to find every way to choose exactly ***two*** elements from a set.
- We could do something like this:

```
for (int x: mySet) {
  for (int y: mySet) {
    if (x != y) {
      cout << x << ", " << y << endl;
    }
  }
}
```

Generating Combinations

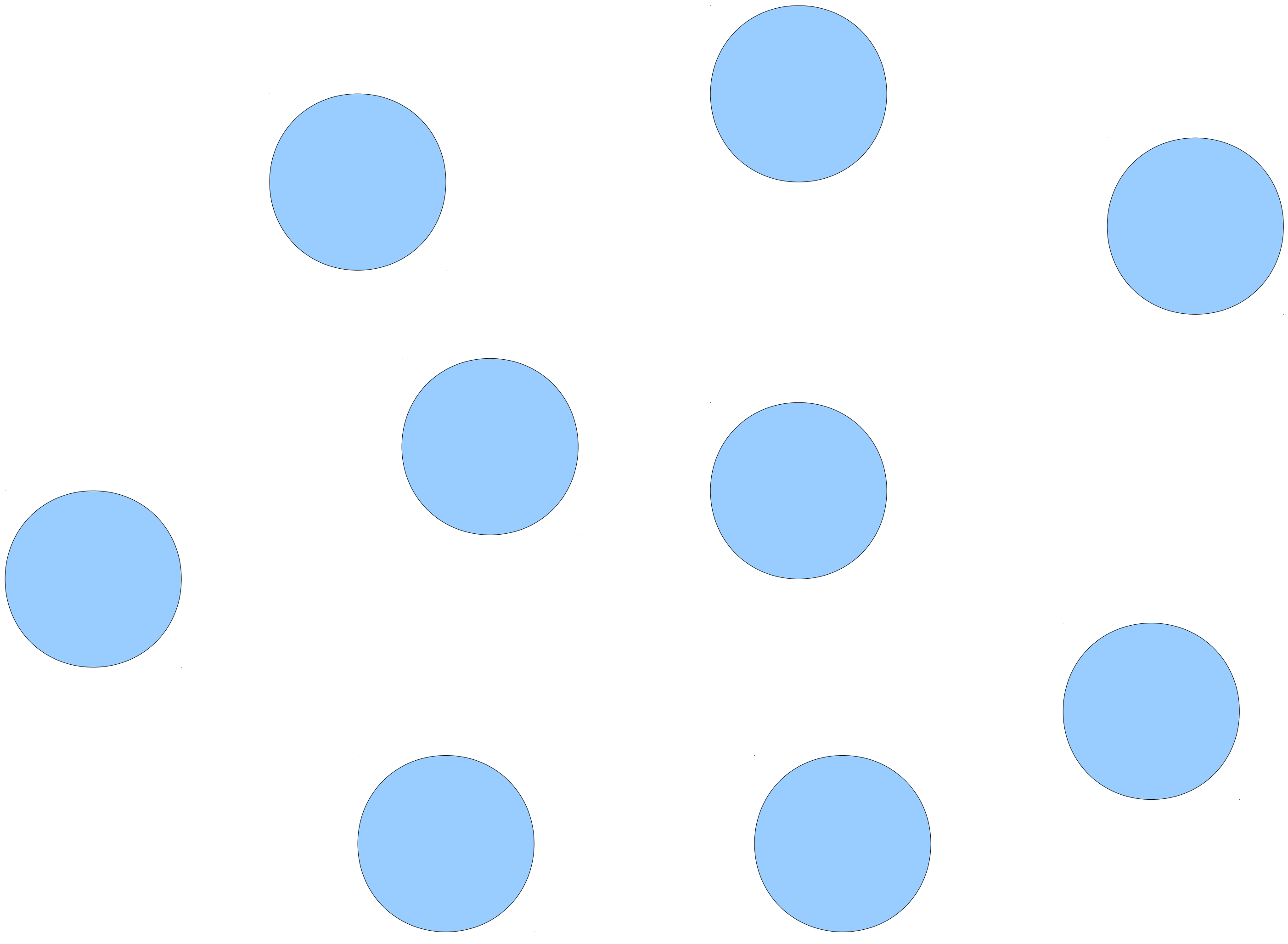
- Suppose that we want to find every way to choose exactly ***three*** elements from a set.
- We could do something like this:

```
for (int x: mySet) {
  for (int y: mySet) {
    for (int z: mySet) {
      if (x != y && x != z && y != z) {
        cout << x << ", " << y << ", " << z << endl;
      }
    }
  }
}
```

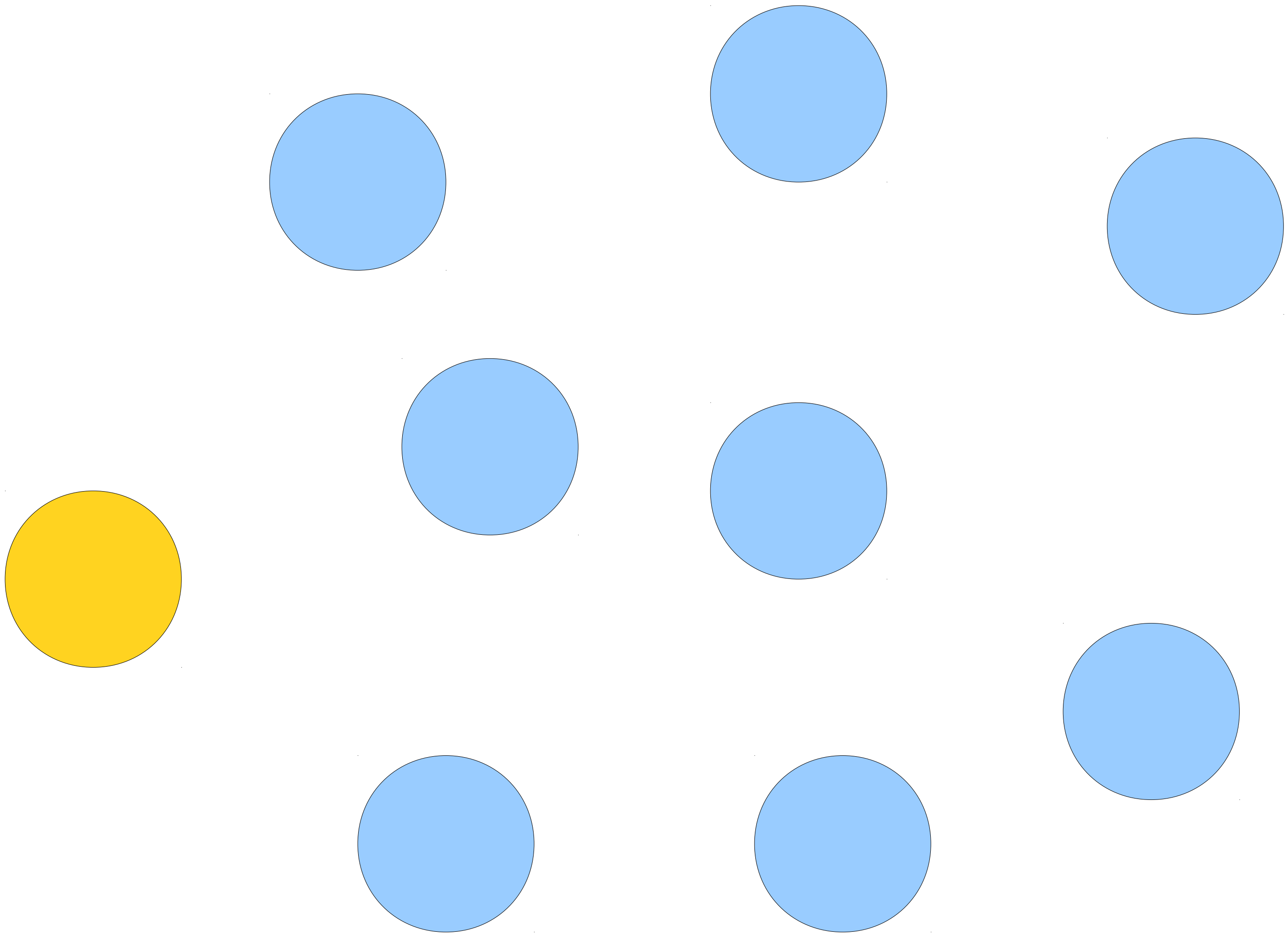
Generating Combinations

- If we know how many elements we want in advance, we can always just nest a whole bunch of loops.
- But what if we don't know in advance?
- Or we *do* know in advance, but it’s a large number and we don’t want to type until our fingers bleed?

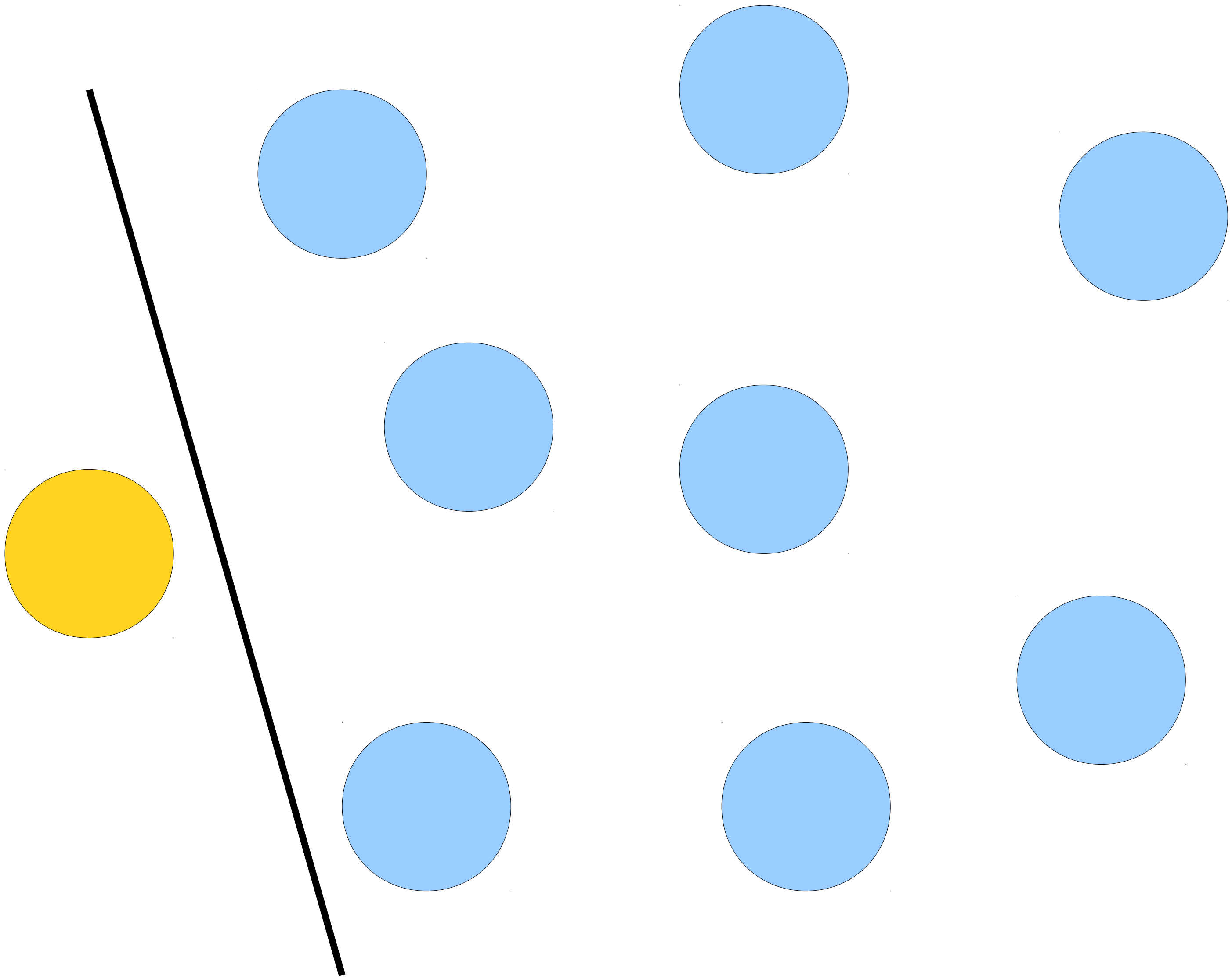
Generating Combinations



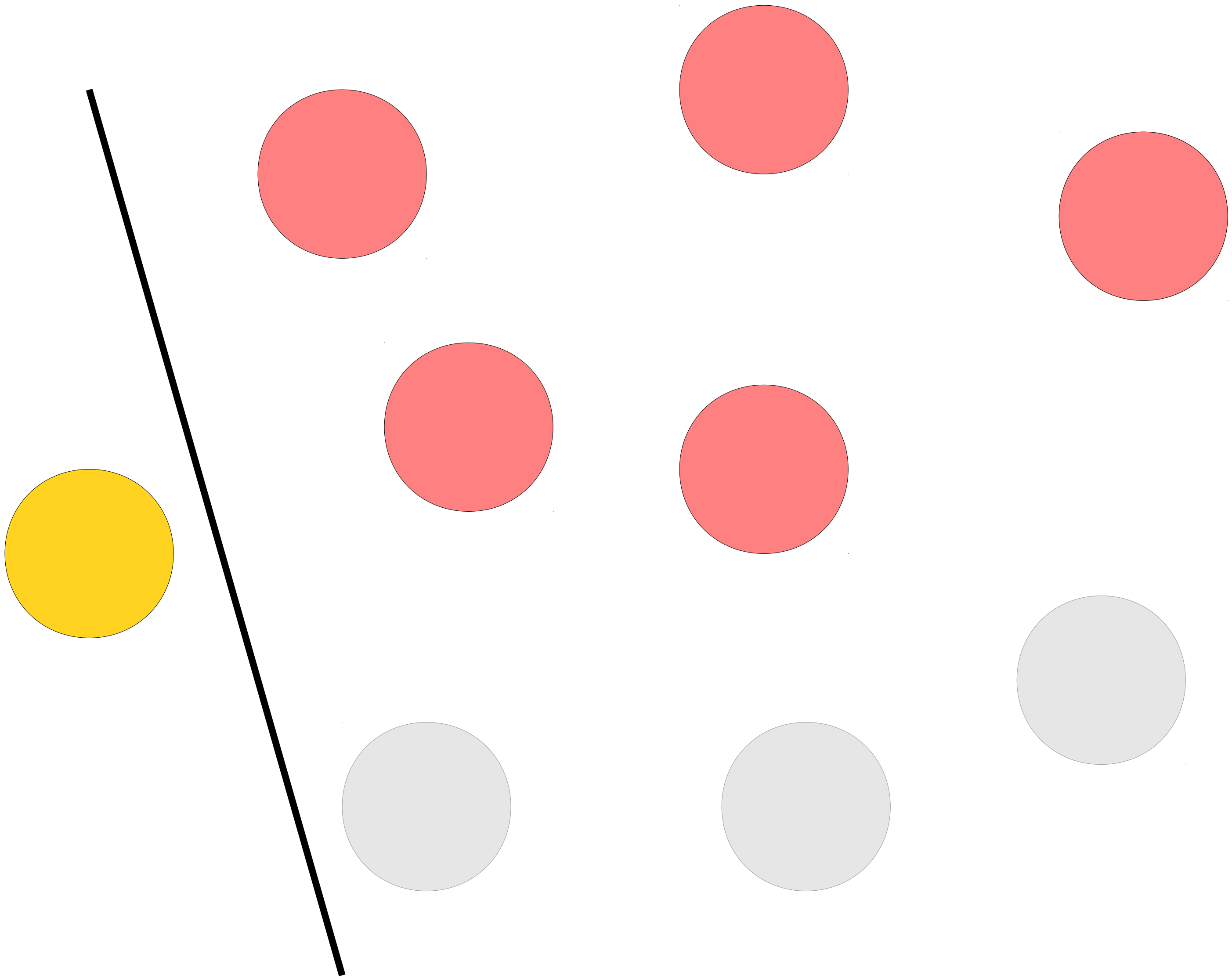
Generating Combinations



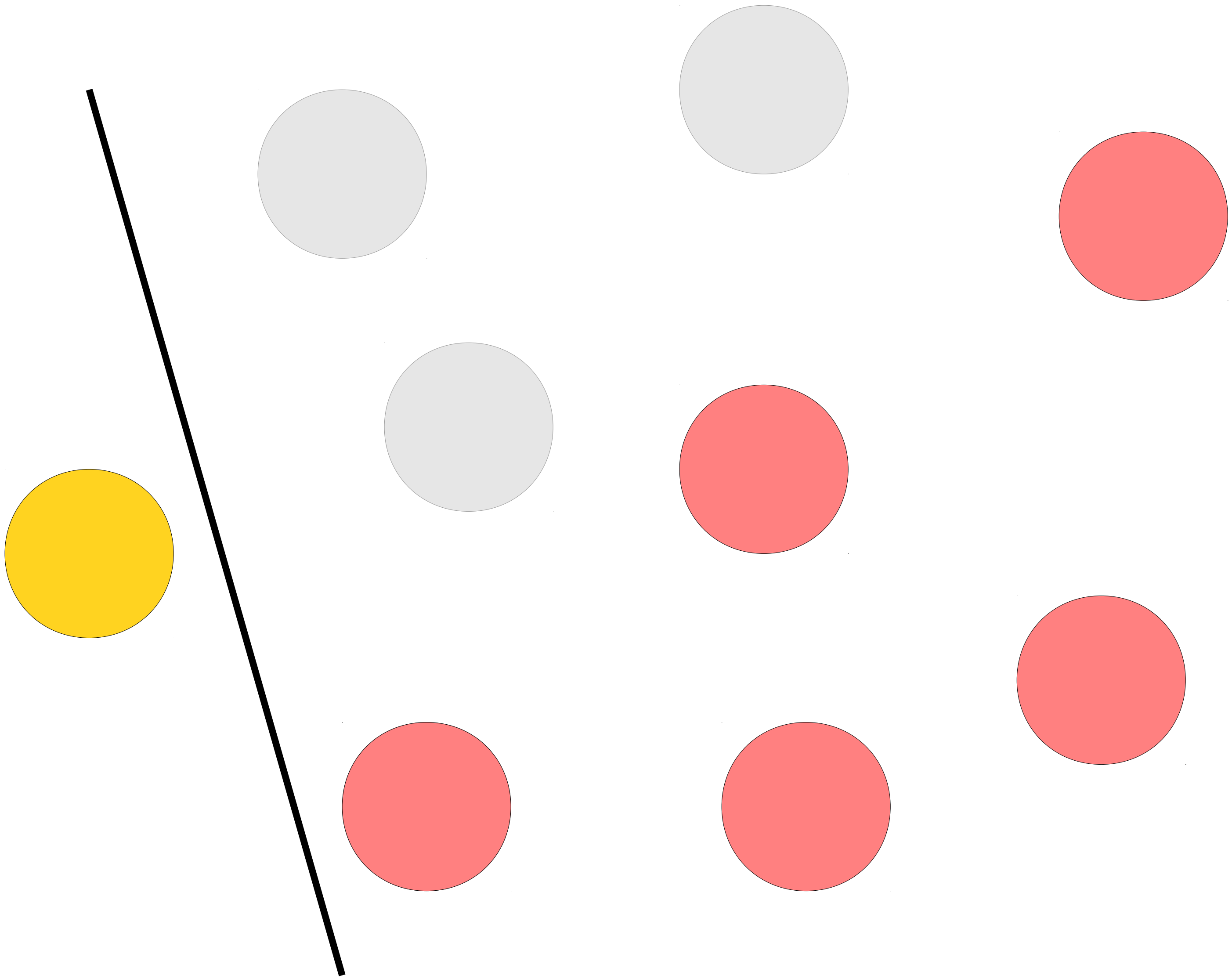
Generating Combinations



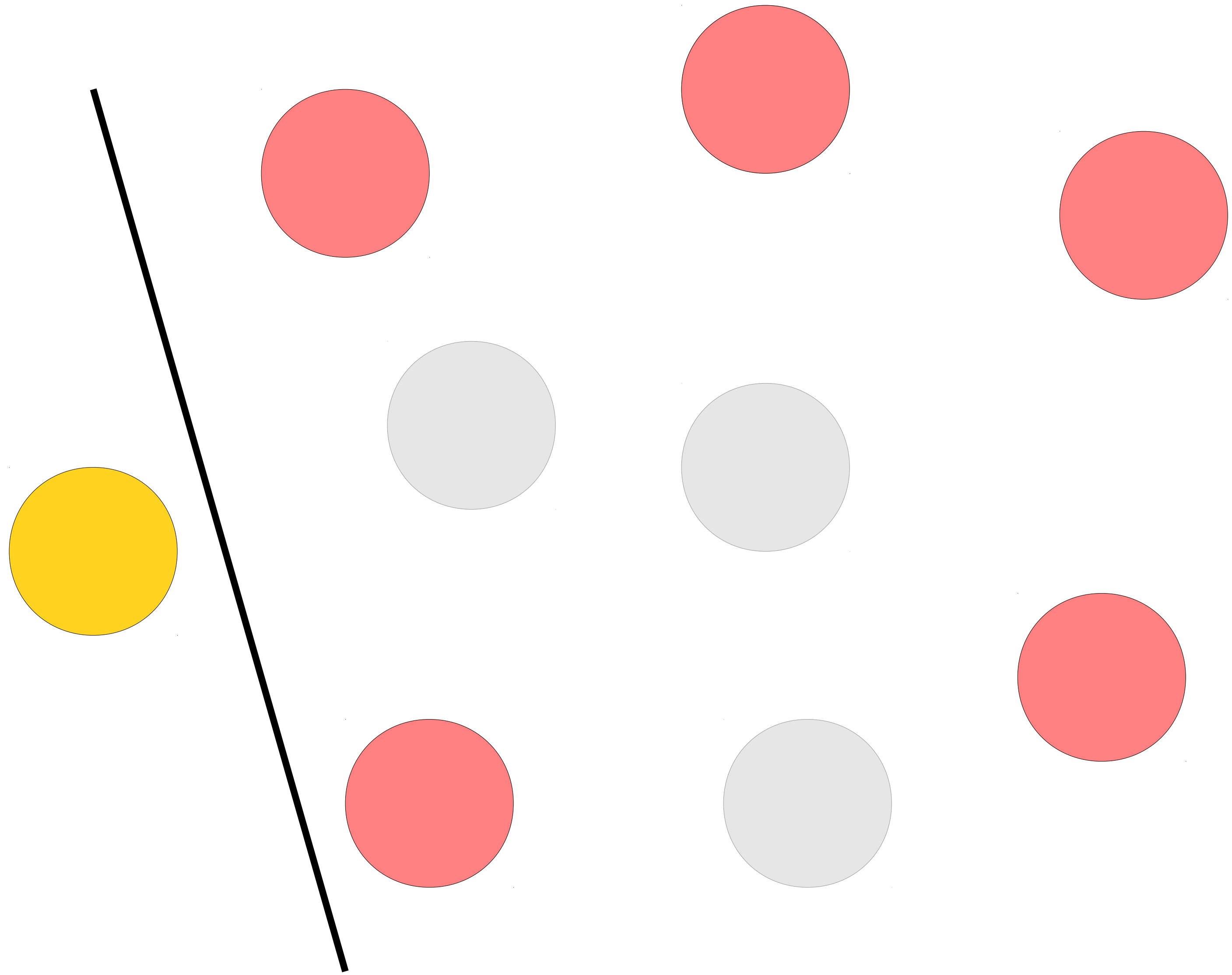
Generating Combinations



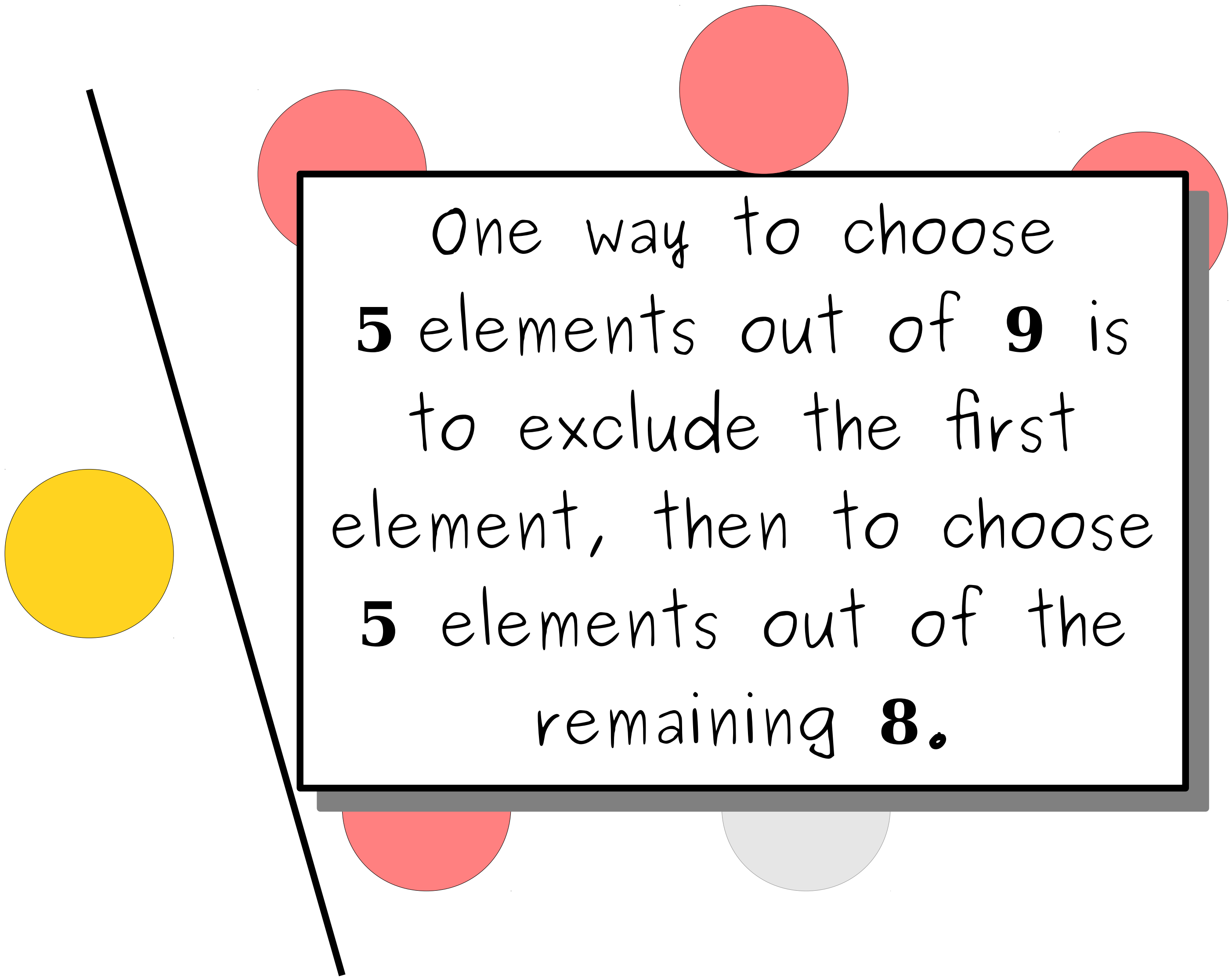
Generating Combinations



Generating Combinations

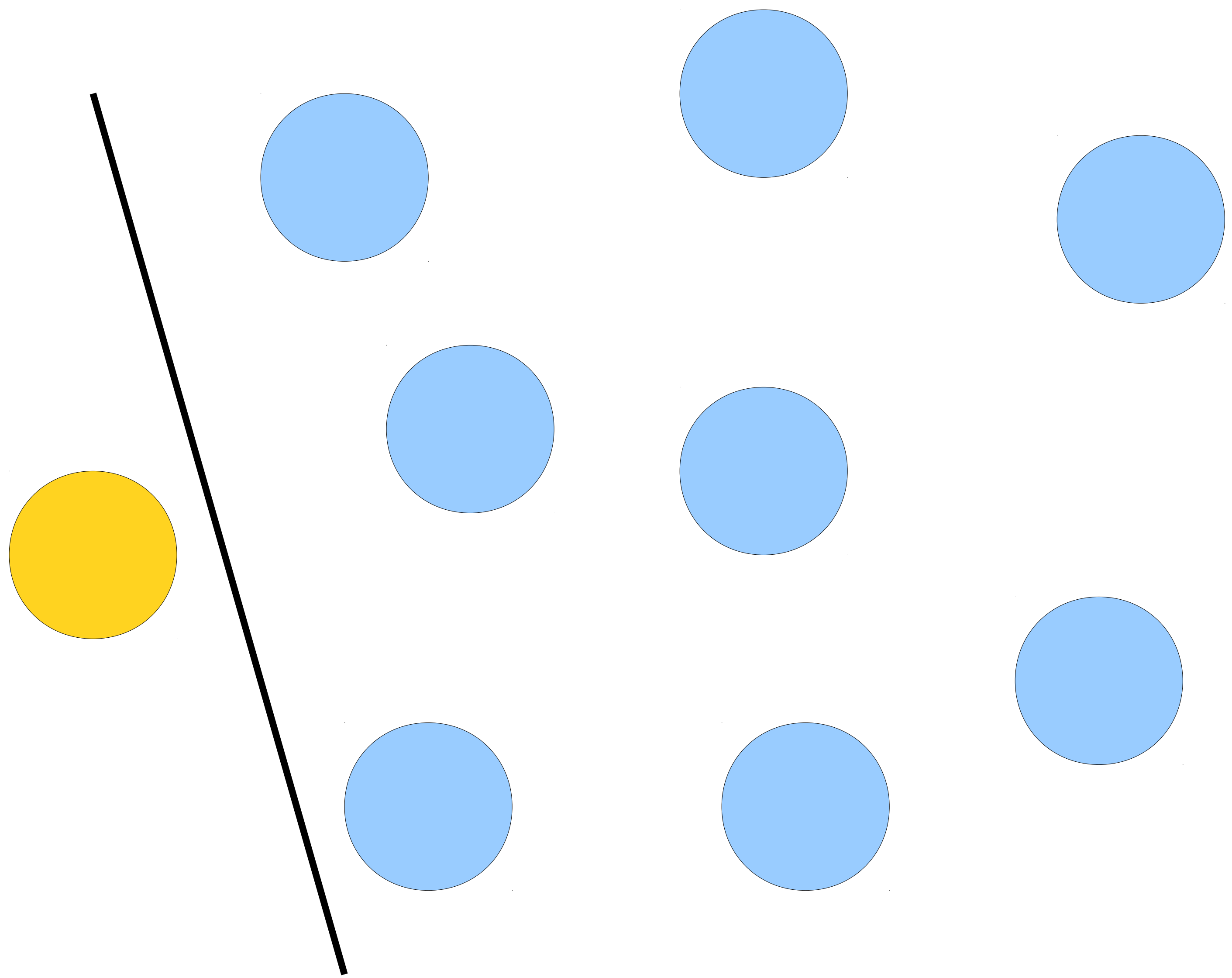


Generating Combinations

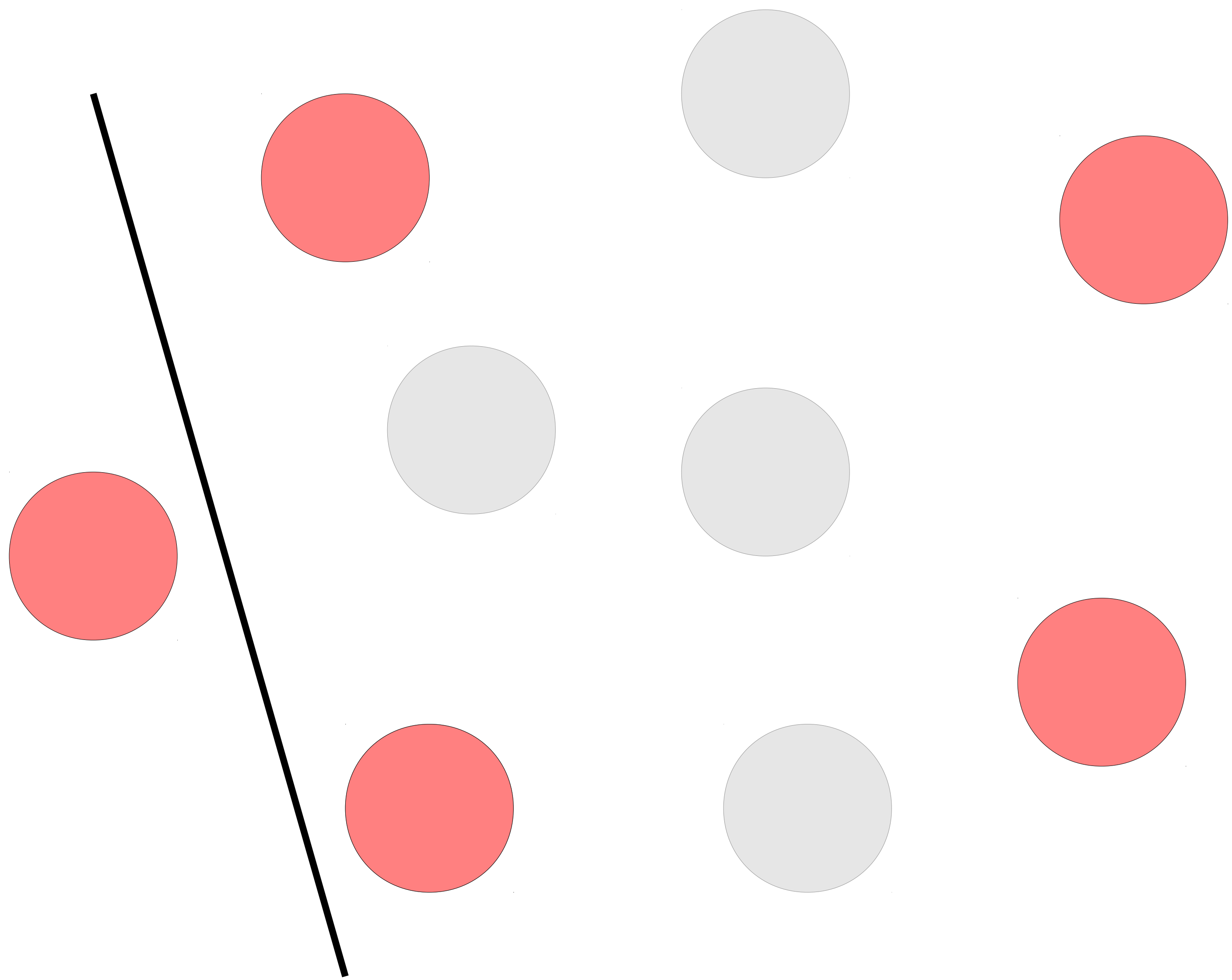


One way to choose **5** elements out of **9** is to exclude the first element, then to choose **5** elements out of the remaining **8**.

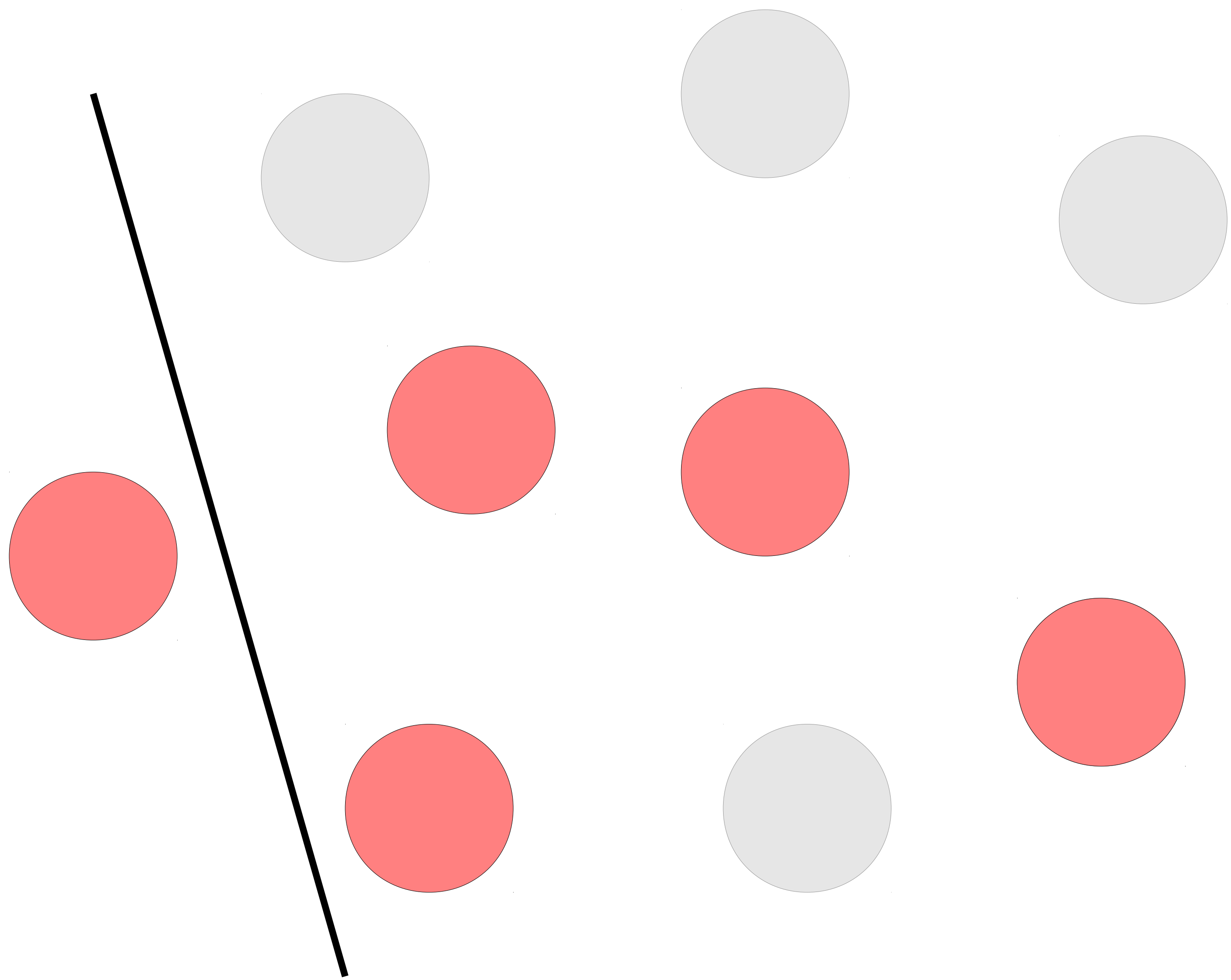
Generating Combinations



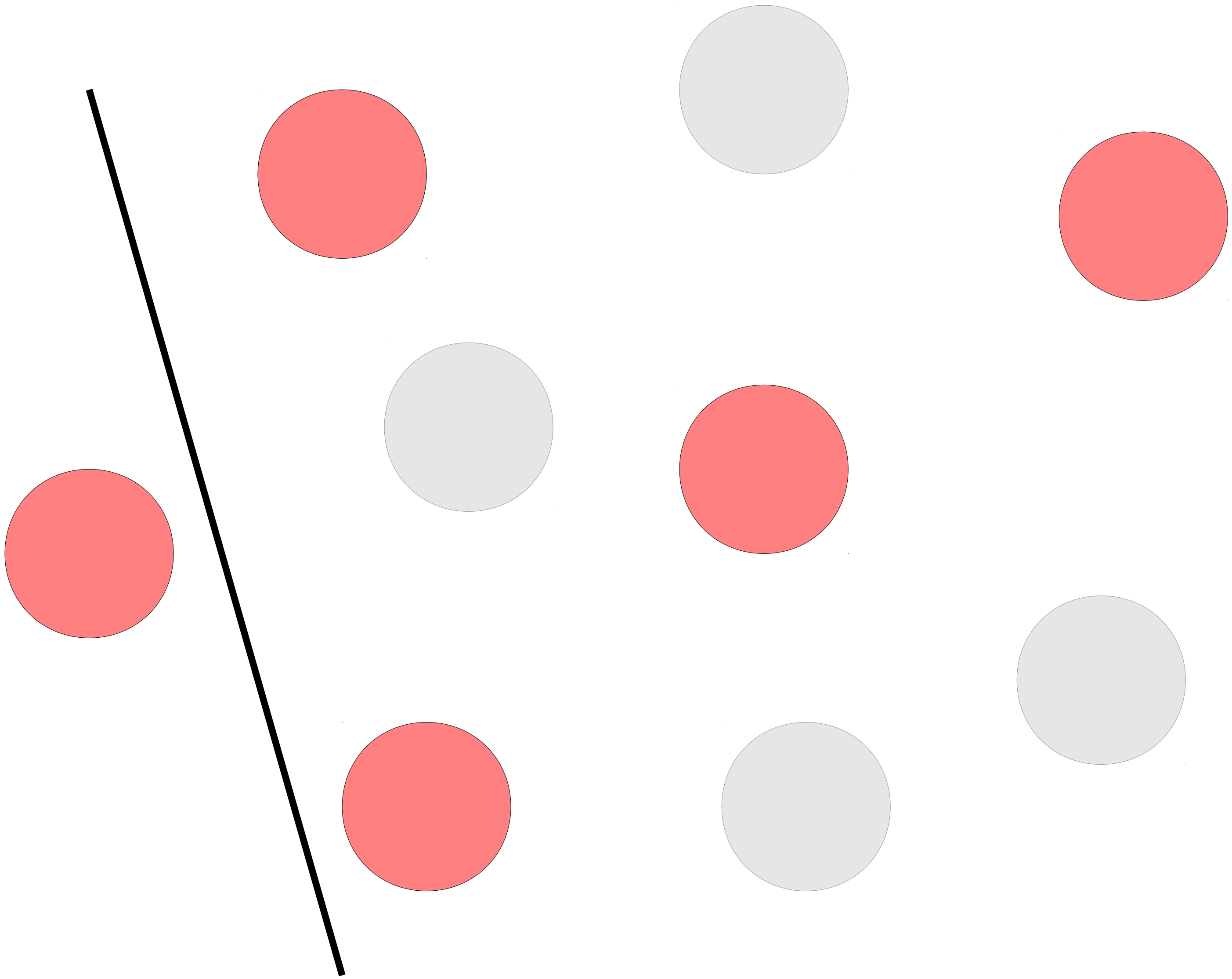
Generating Combinations



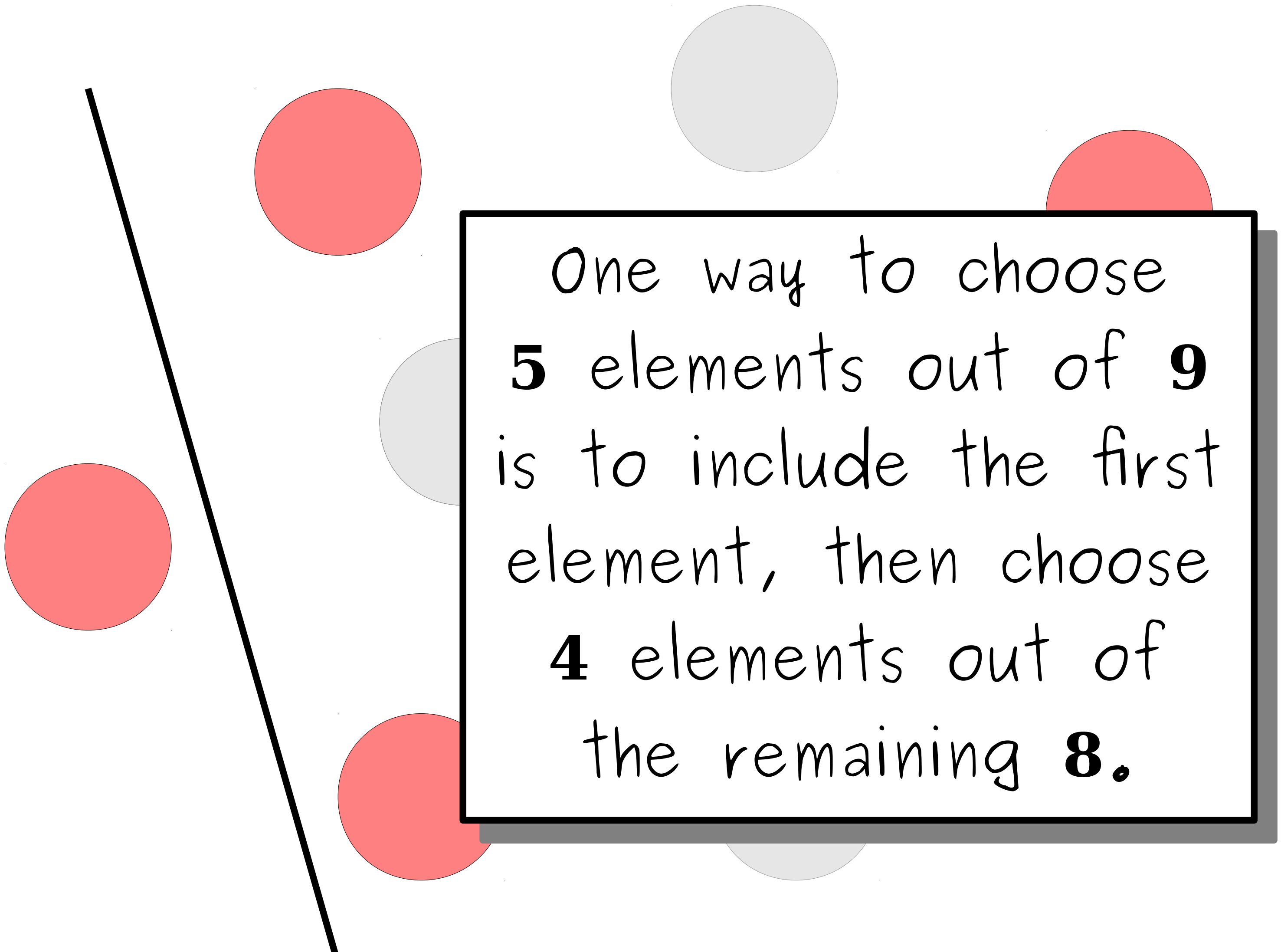
Generating Combinations



Generating Combinations

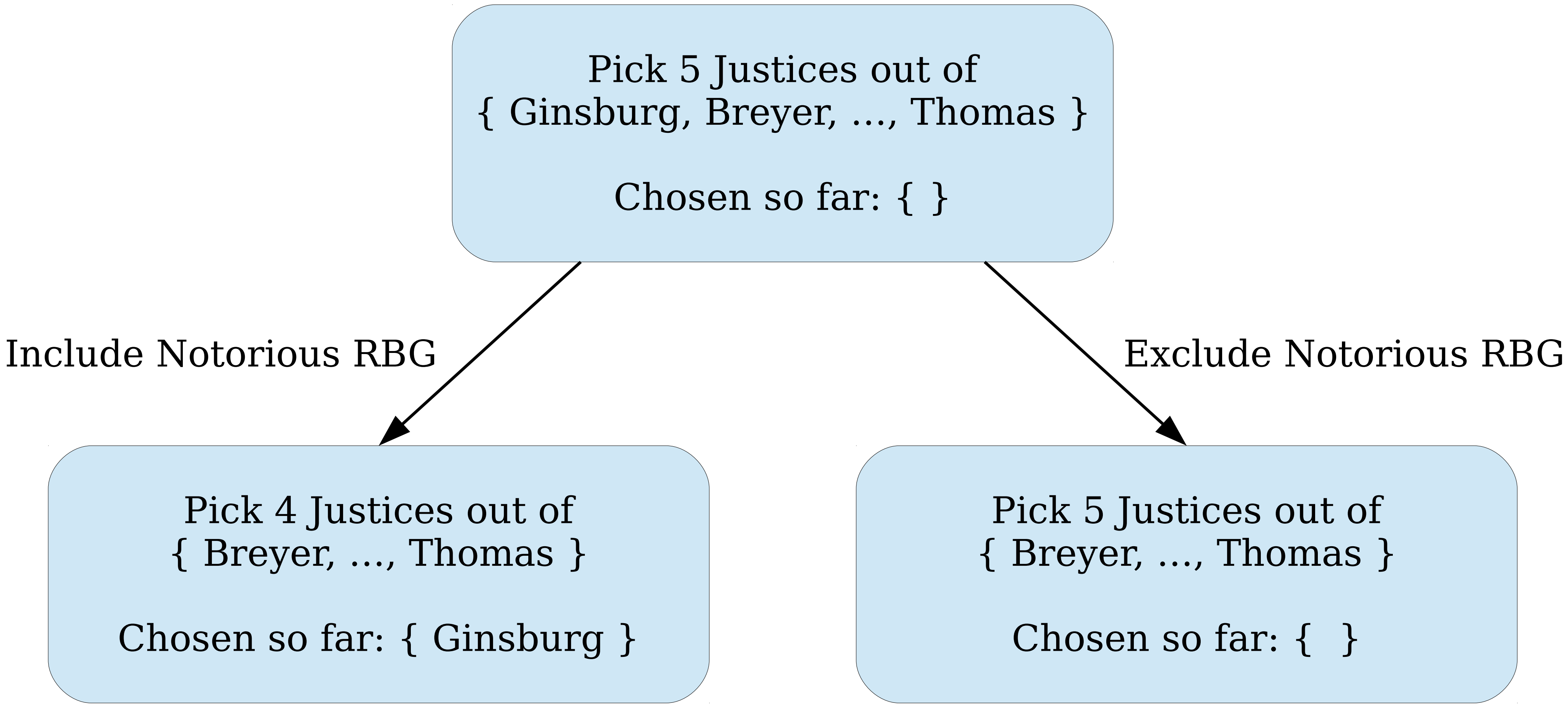


Generating Combinations



One way to choose **5** elements out of **9** is to include the first element, then choose **4** elements out of the remaining **8**.

Judicial Decisions



Combinations, Recursively

- ***Base Cases:***
  - If $k = 0$, then we've already picked all our elements and should output what we have.
  - If $k \neq 0$ but the remaining set of choices, there's nothing we can do to get up to k elements.
- ***Recursive Step:***
  - Pick some element x from the set.
  - Find all ways of picking k elements of what remains, excluding x from what you find.
  - Find all ways of picking $k - 1$ elements of what remains, including x in what you find.

A Little Word Puzzle

“What nine-letter word can be reduced to a single-letter word one letter at a time by removing letters, leaving it a legal word at each step?”

The Startling Truth

S	T	A	R	T	L	I	N	G

The Startling Truth

S	T	A	R	T	I	N	G

The Startling Truth

S	T	A	R	I	N	G

The Startling Truth

S	T	R	I	N	G

The Startling Truth

S	T	I	N	G

The Startling Truth

S	I	N	G

The Startling Truth

S	I	N

The Startling Truth

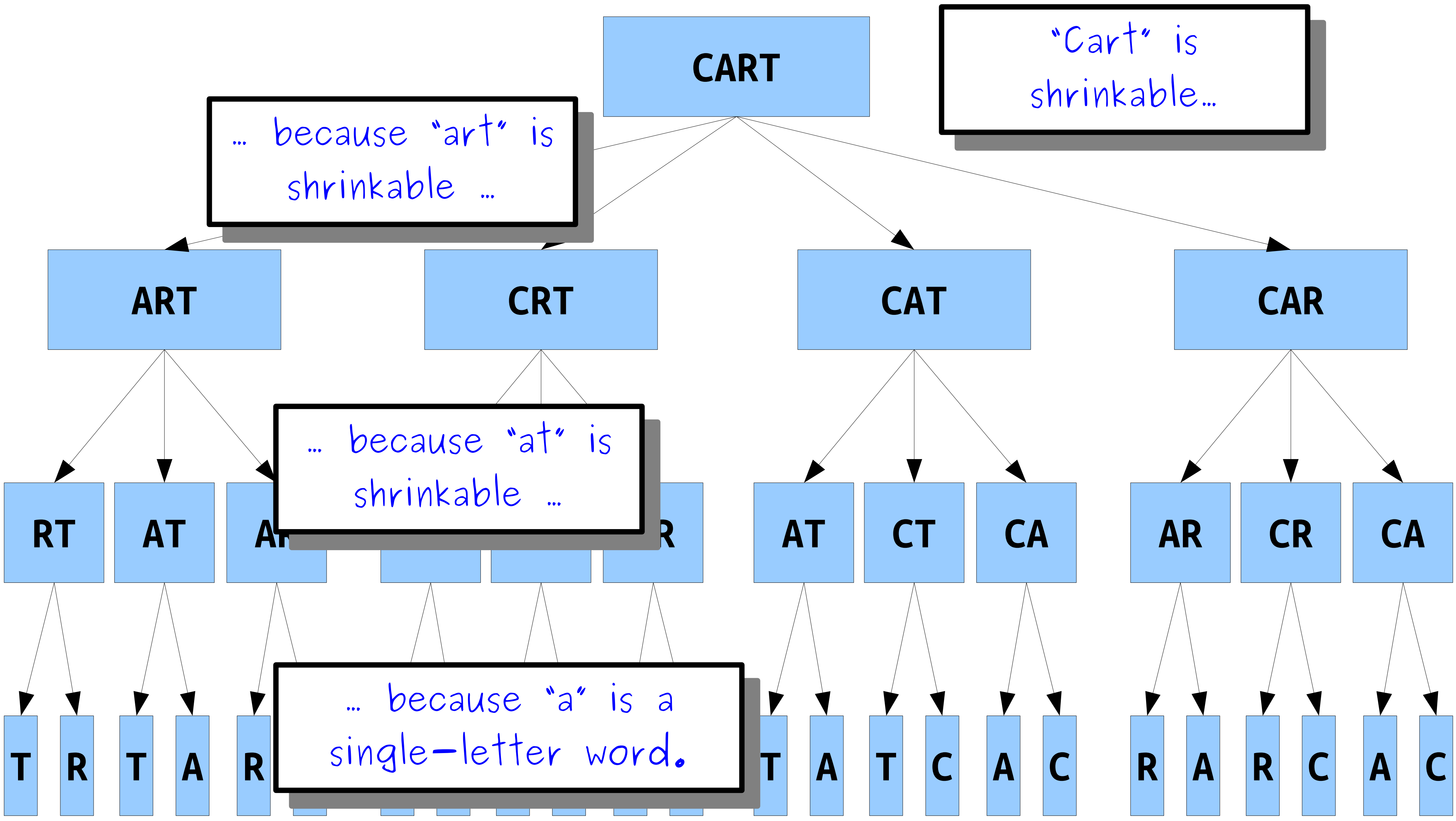
I	N

The Startling Truth

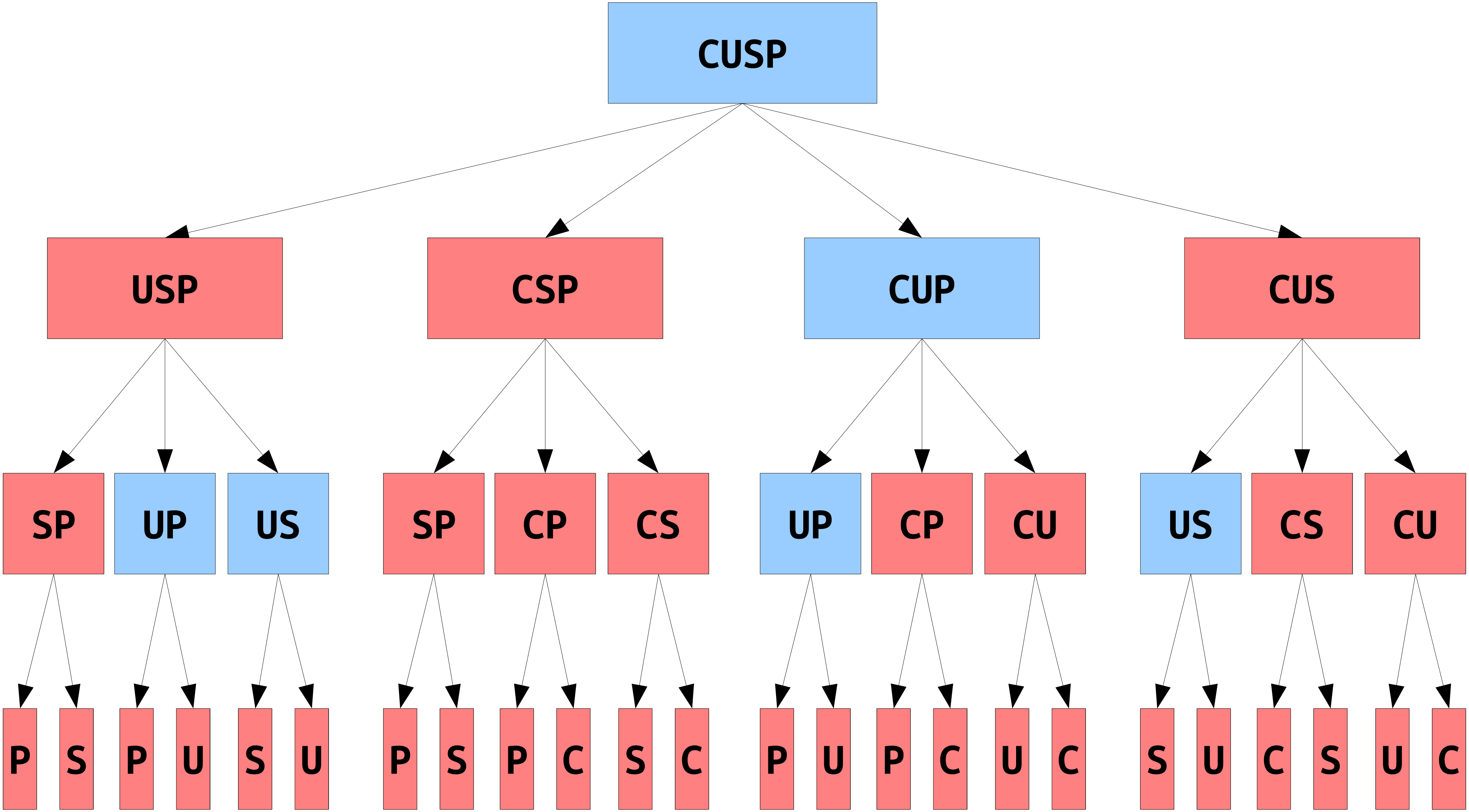
I

Is there ***really*** just one nine-letter word with this property?

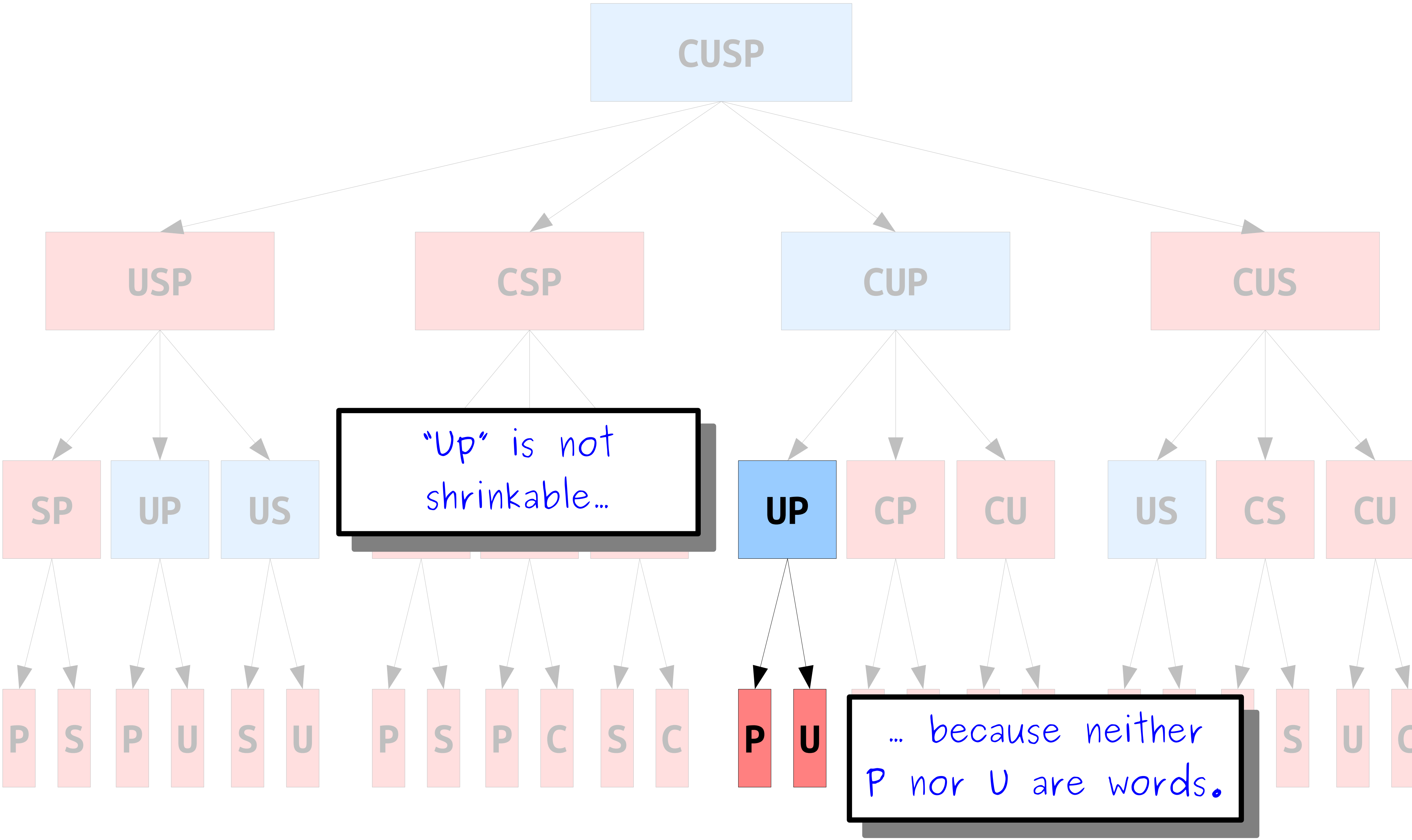
All Possible Paths



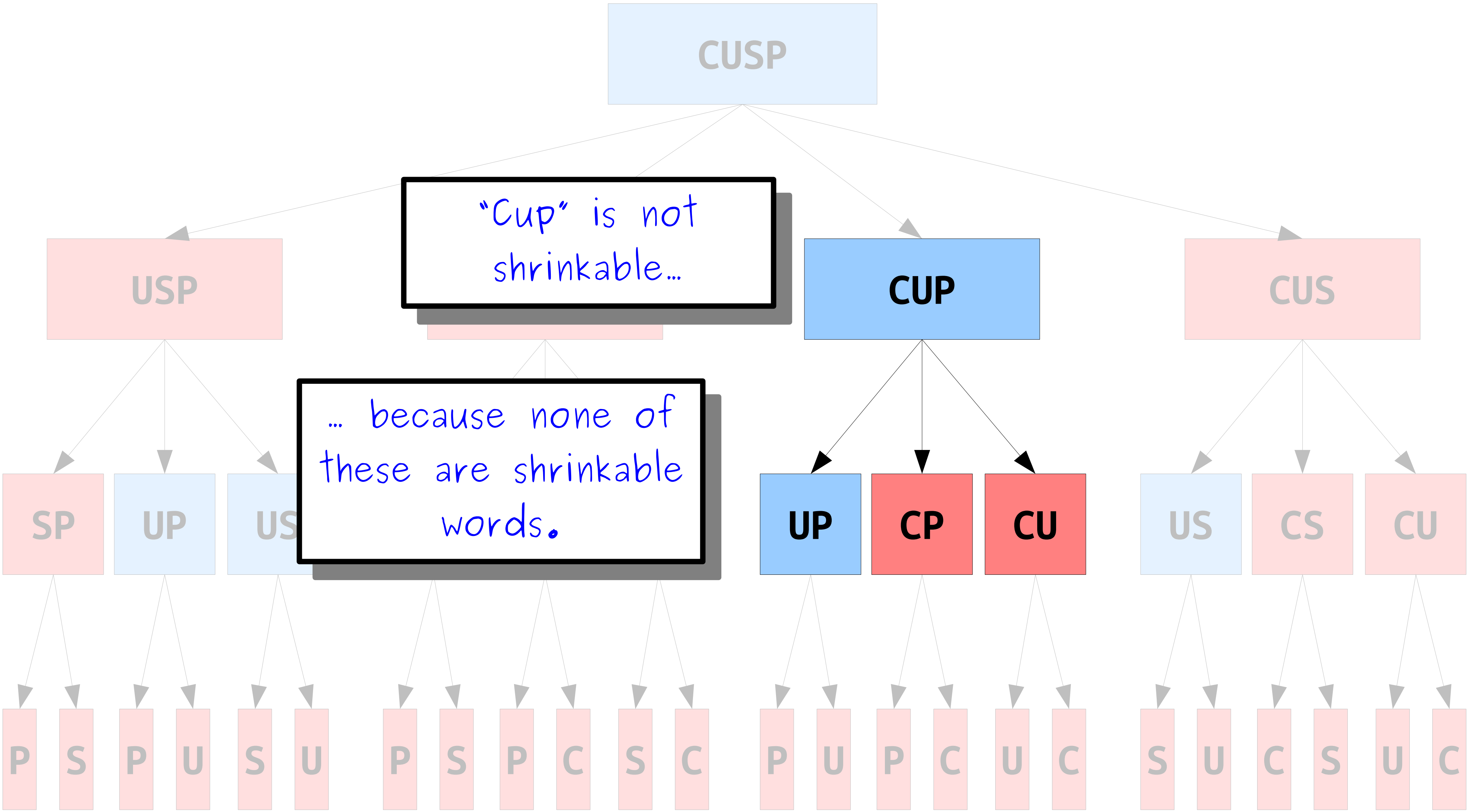
All Possible Paths



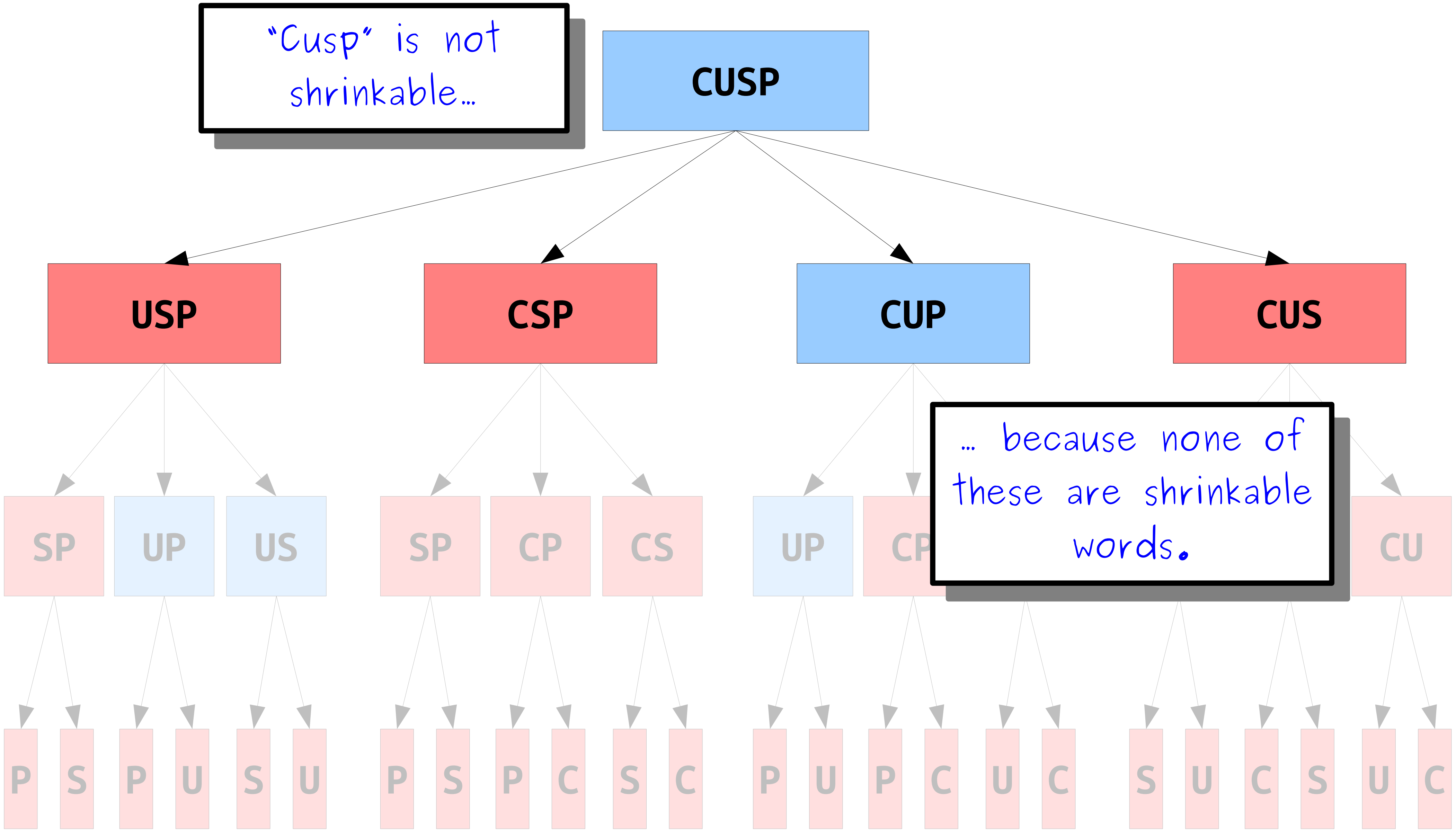
All Possible Paths



All Possible Paths



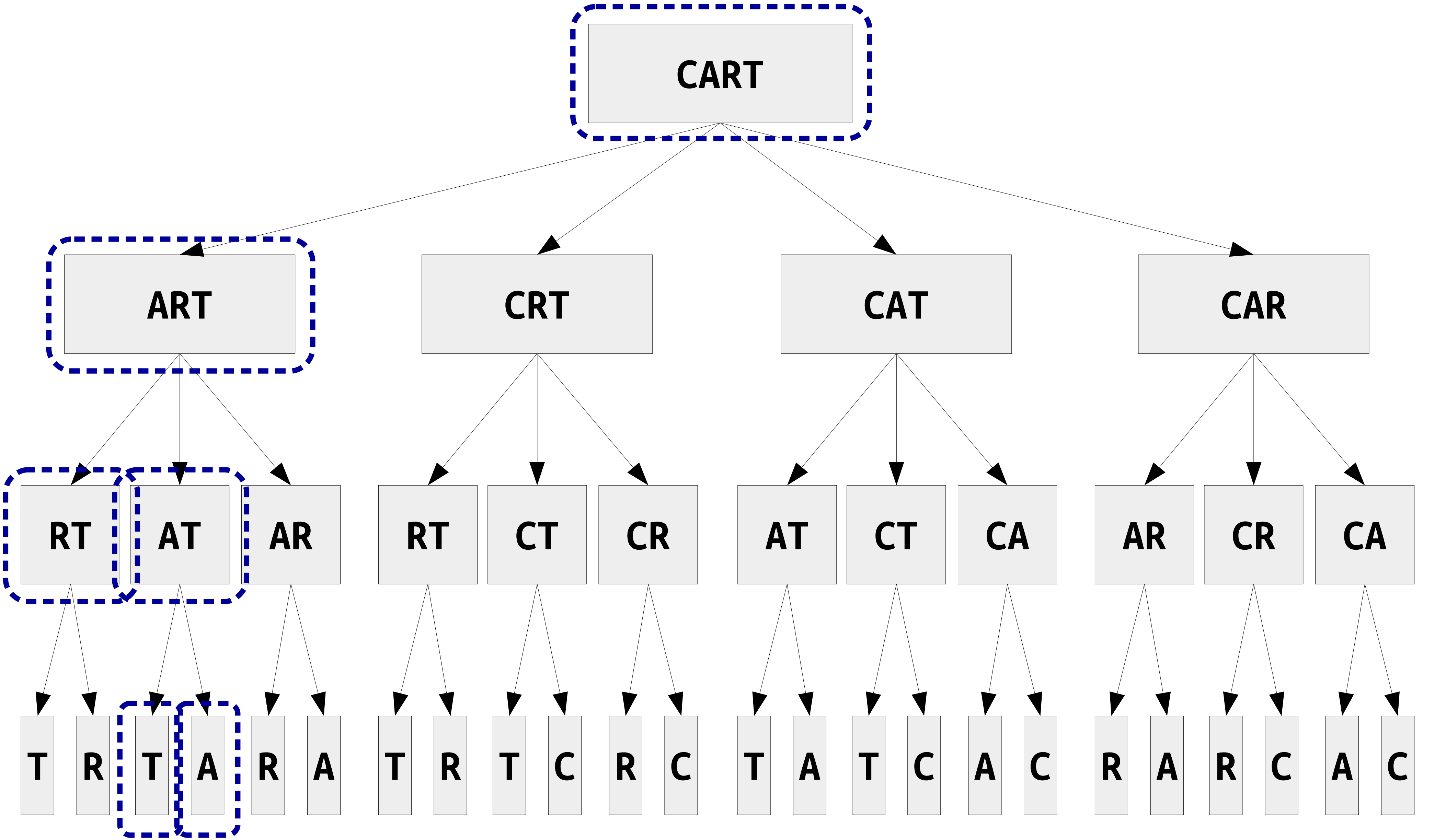
All Possible Paths



Shrinkable Words

- Let's define a ***shrinkable word*** as a word that can be reduced down to one letter by removing one character at a time, leaving a word at each step.
- ***Base Cases*:**
  - A string that is not a word is not a shrinkable word.
  - Any single-letter word is shrinkable (A, I, and O).
- ***Recursive Step*:**
  - A multi-letter word is shrinkable if you can remove a letter to form a shrinkable word.
  - A multi-letter word is not shrinkable if no matter what letter you remove, it's not shrinkable.

Finding a Good Shrink



Recursive Backtracking

- The function we have just written is an example of ***recursive backtracking***.
- At each step, we try one of many possible options.
- If *any* option succeeds, that's great! We're done.
- If *none* of the options succeed, then this particular problem can't be solved.

```
bool isShrinkable(const string& word, const Lexicon& english) {
    if (!english.contains(word)) return false;
    if (word.length() == 1) return true;

    for (int i = 0; i < word.length(); i++) {
        string shrunken = word.substr(0, i) + word.substr(i + 1);
        if (isShrinkable(shrunken, english)) {
            return true;
        }
    }
    return false;
}
```

```
bool isShrinkable(const string& word, const Lexicon& english) {
    if (!english.contains(word)) return false;
    if (word.length() == 1) return true;

    for (int i = 0; i < word.length(); i++) {
        string shrunken = word.substr(0, i) + word.substr(i + 1);
        if (isShrinkable(shrunken, english)) {
            return true;
        }
    }
    return false;
}
```

```
bool isShrinkable(const string& word, const Lexicon& english) {
    if (!english.contains(word)) return false;
    if (word.length() == 1) return true;

    for (int i = 0; i < word.length(); i++) {
        string shrunken = word.substr(0, i) + word.substr(i + 1);
        return isShrinkable(shrunken, english); // ⚠ Bad Idea ⚠
    }


    return false;
}
```

```
bool isShrinkable(const string& word, const Lexicon& english) {
    if (!english.contains(word)) return false;
    if (word.length() == 1) return true;

    for (int i = 0; i < word.length(); i++) {
        string shrunken = word.substr(0, i) + word.substr(i + 1);
        return isShrinkable(shrunken, english); // ⚠ Bad Idea ⚠
    }
    return false;
}
```

Your Action Items

- ***Read Chapter 9 of the textbook.***
  - There's tons of cool backtracking examples there, and it will help you prep for Friday.
- ***Keep working on Assignment 3.***
  - Aim to complete the first three parts by tonight if you can.
  - Try to complete all four parts by Friday evening so you have time to clean things up and ask questions.

Next Time

- ***More Backtracking***
  - Techniques in searching for feasibility.
- ***Closing Thoughts on Recursion***
  - It'll come back, but we're going to focus on other things for a while!