Thinking Recursively

Part Three

Friday Four Square!
4:15PM, Outside Gates

Permutations

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

Permutations

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

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

Listing all Permutations

- Like subsets, permutations are an important structure in programming.
- Listing all permutations is useful for answering questions like these:
  - What is the best order in which to perform a series of tasks?
  - What possible DNA strands can be made by assembling smaller fragments together?

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generating Permutations

x_1	x_2	x_3	x_4

x_1	x_2	x_3	x_4
x_1	x_2	x_4	x_3
x_1	x_3	x_2	x_4
x_1	x_3	x_4	x_2
x_1	x_4	x_2	x_3
x_1	x_4	x_3	x_2

x_2	x_1	x_3	x_4
x_2	x_1	x_4	x_3
x_2	x_3	x_1	x_4
x_2	x_3	x_4	x_1
x_2	x_4	x_1	x_3
x_2	x_4	x_3	x_1

x_3	x_1	x_2	x_4
x_3	x_1	x_4	x_2
x_3	x_2	x_1	x_4
x_3	x_2	x_4	x_1
x_3	x_4	x_1	x_2
x_3	x_4	x_2	x_1

x_4	x_1	x_2	x_3
x_4	x_1	x_3	x_2
x_4	x_2	x_1	x_3
x_4	x_2	x_3	x_1
x_4	x_3	x_1	x_2
x_4	x_3	x_2	x_1

Generating Permutations

- How to generate all permutations of a string?
- **Base Case**:
  - If the string is empty, there is just one permutation – that string itself.
- **Recursive Step:**
  - For each character in the string:
    - Remove that character.
    - Permute the rest of the string.
    - Add that character back in.

Let's Code it Up!

Generating Combinations

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

```
foreach (int x in 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:

```
foreach (int x in mySet) {
  foreach (int y in 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:

```
foreach (int x in mySet) {
  foreach (int y in mySet) {
    foreach (int z in 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?

Pascal's Triangle Revisited

					1					
				1		1				
			1		2		1			
		1		3		3		1		
	1		4		6		4		1	
1		5		10		10		5		1

Pascal's Triangle Revisited

					1					
				1		1				
			1		2		1			
		1		3		3		1		
	1		4		6		4		1	
1		5		10		10		5		1

Pascal's Triangle Revisited

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

Pascal's Triangle Revisited

1

1 1

1 2 1

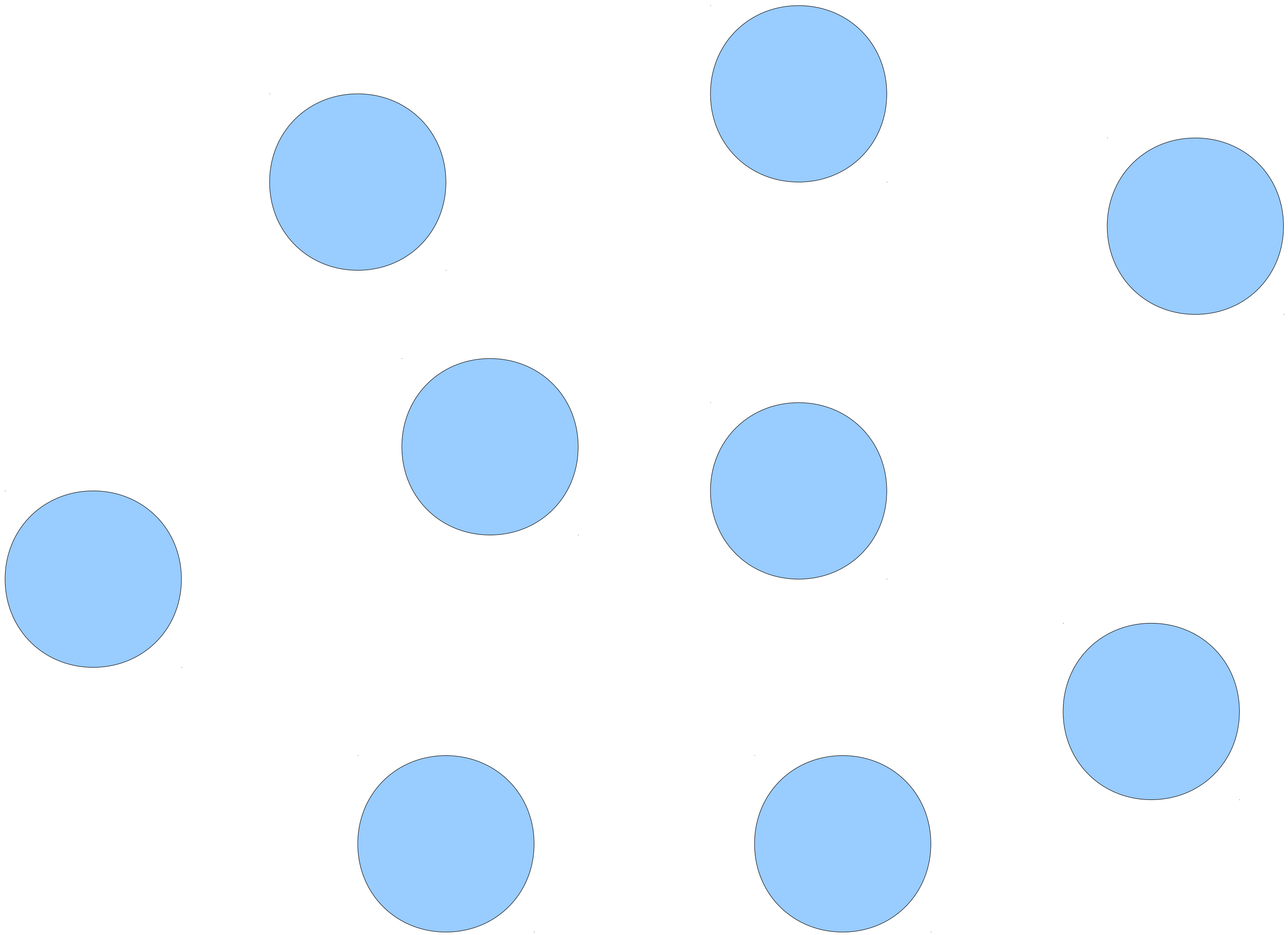
1 3 3 1

1 4 6 4 1

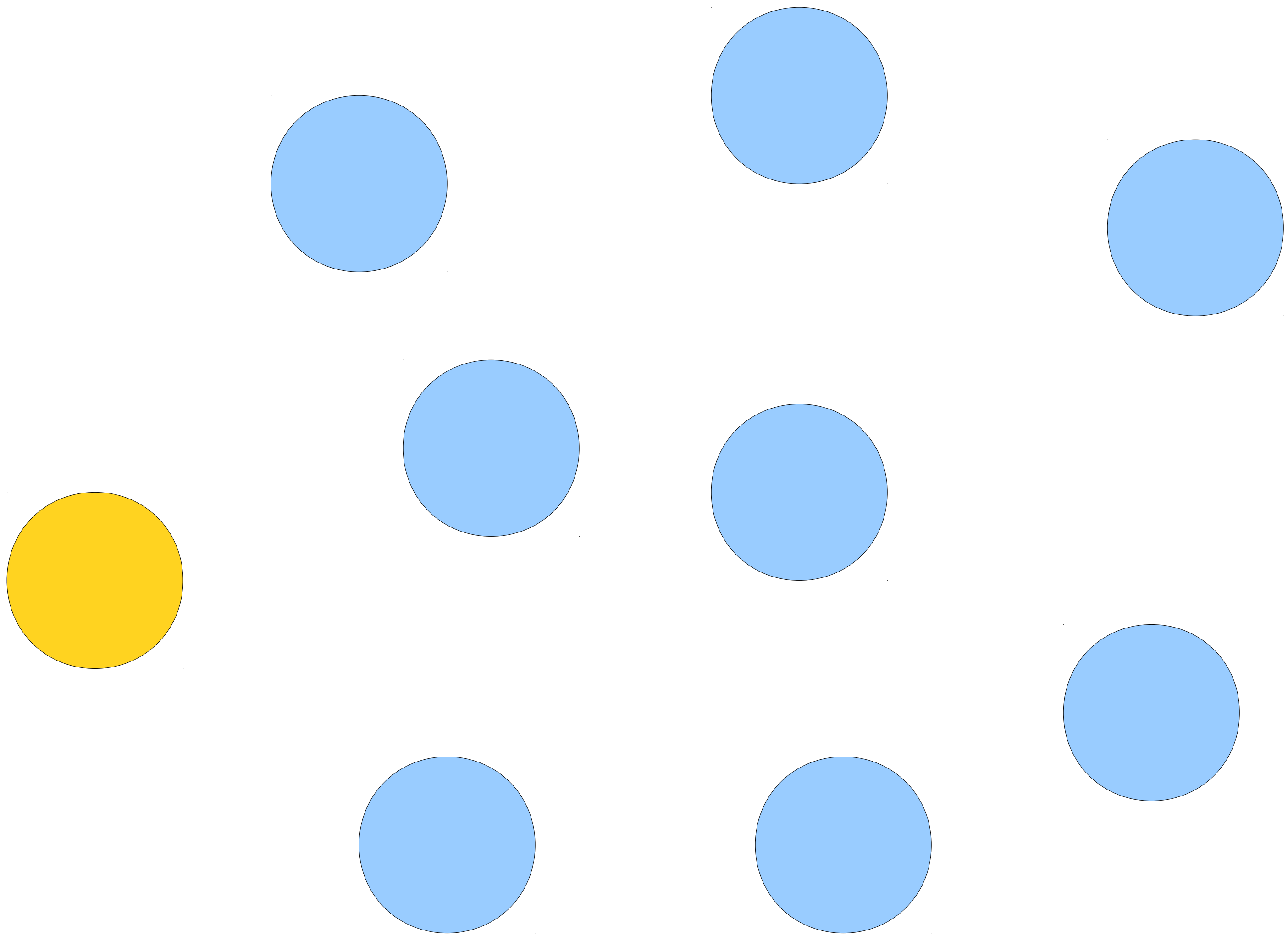
1 5 10 10 5 1

What's up with that?

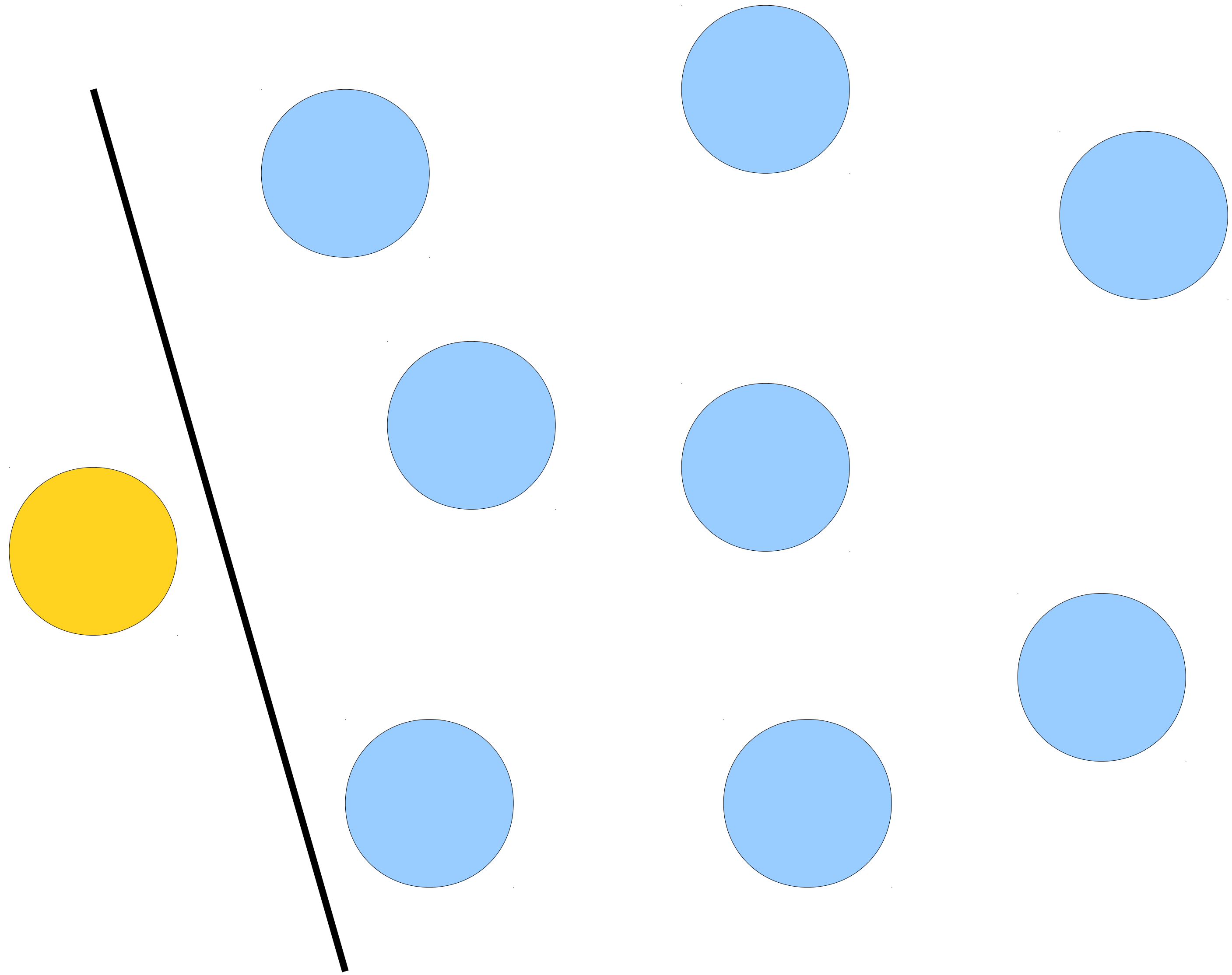
Generating Combinations



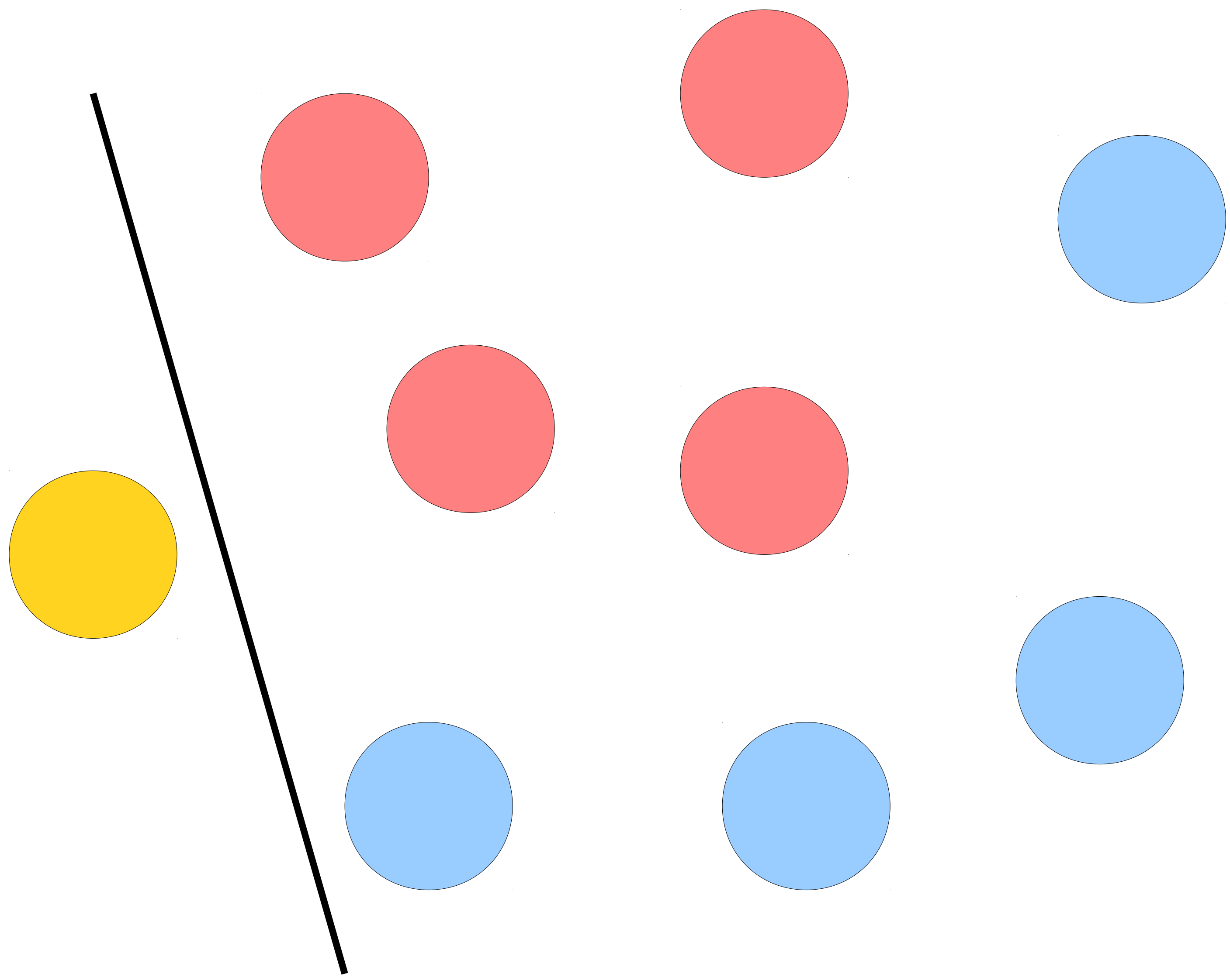
Generating Combinations



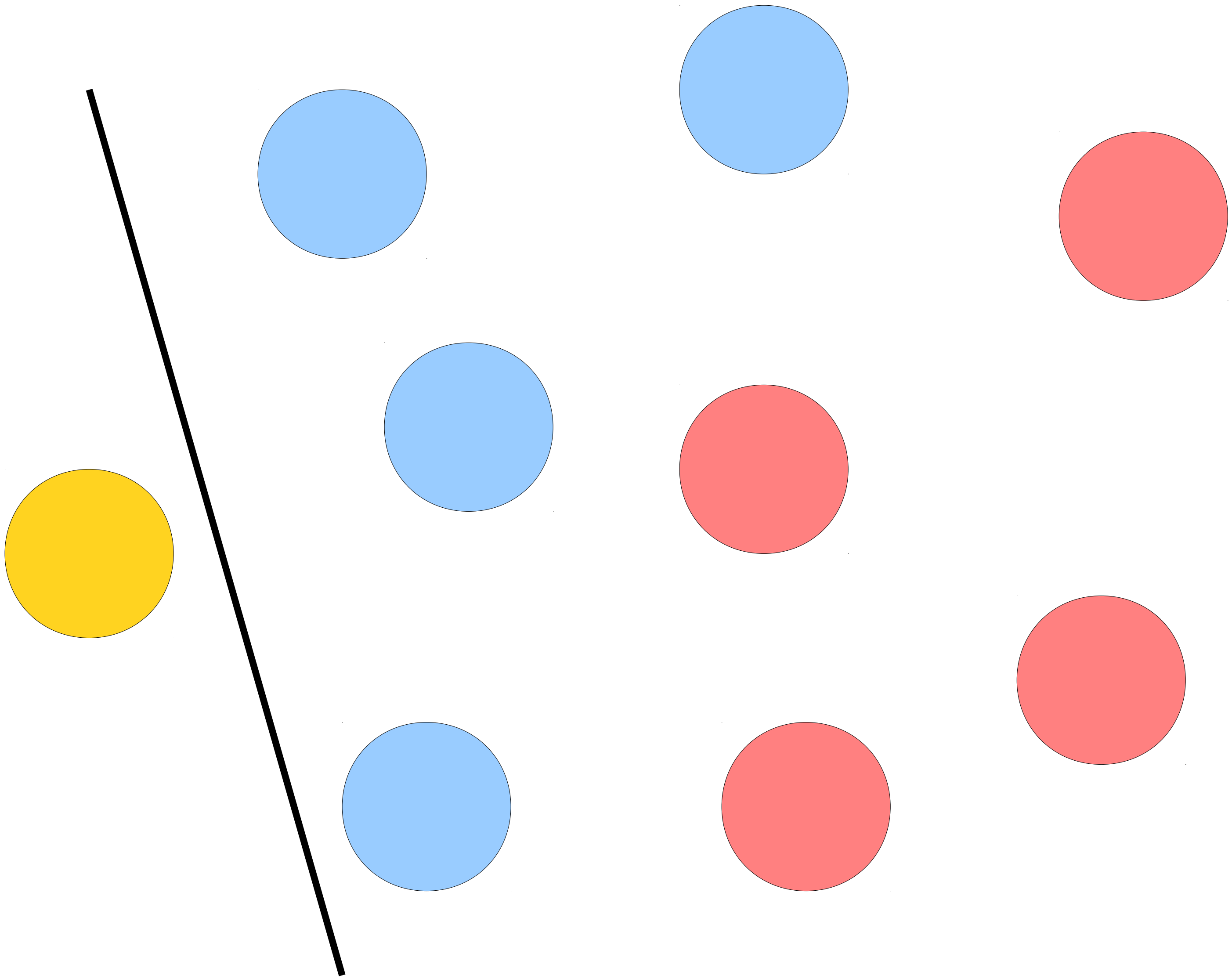
Generating Combinations



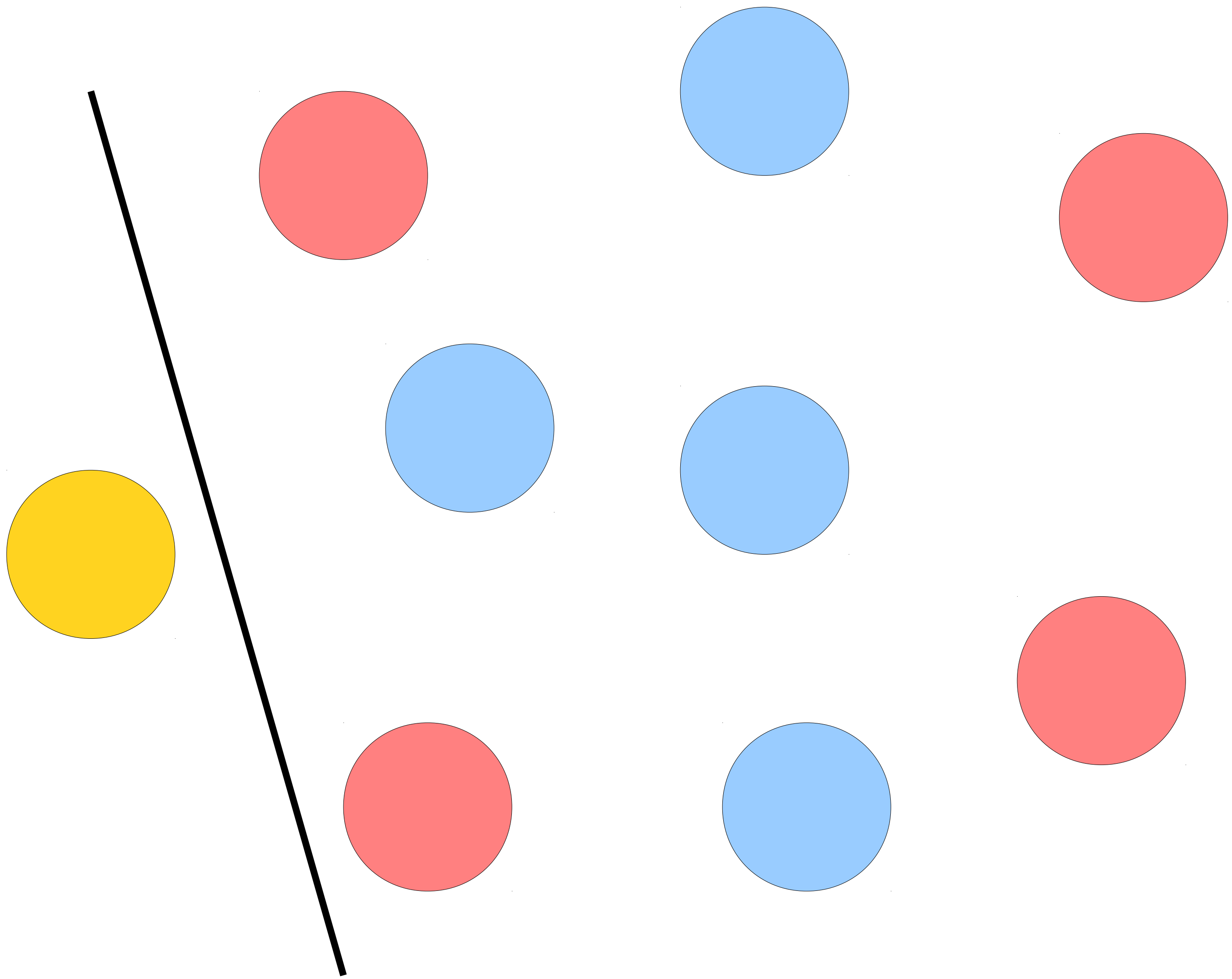
Generating Combinations



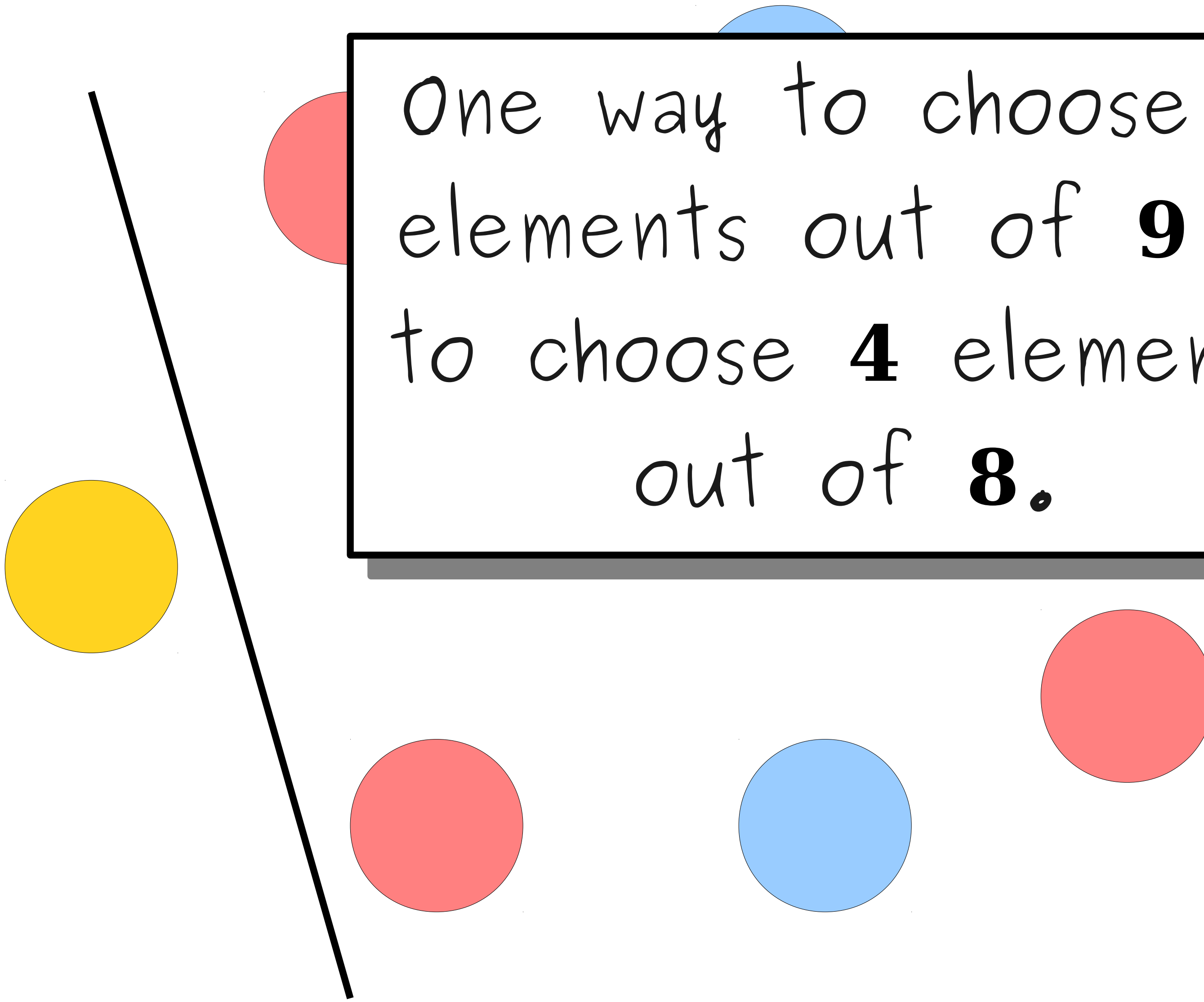
Generating Combinations



Generating Combinations

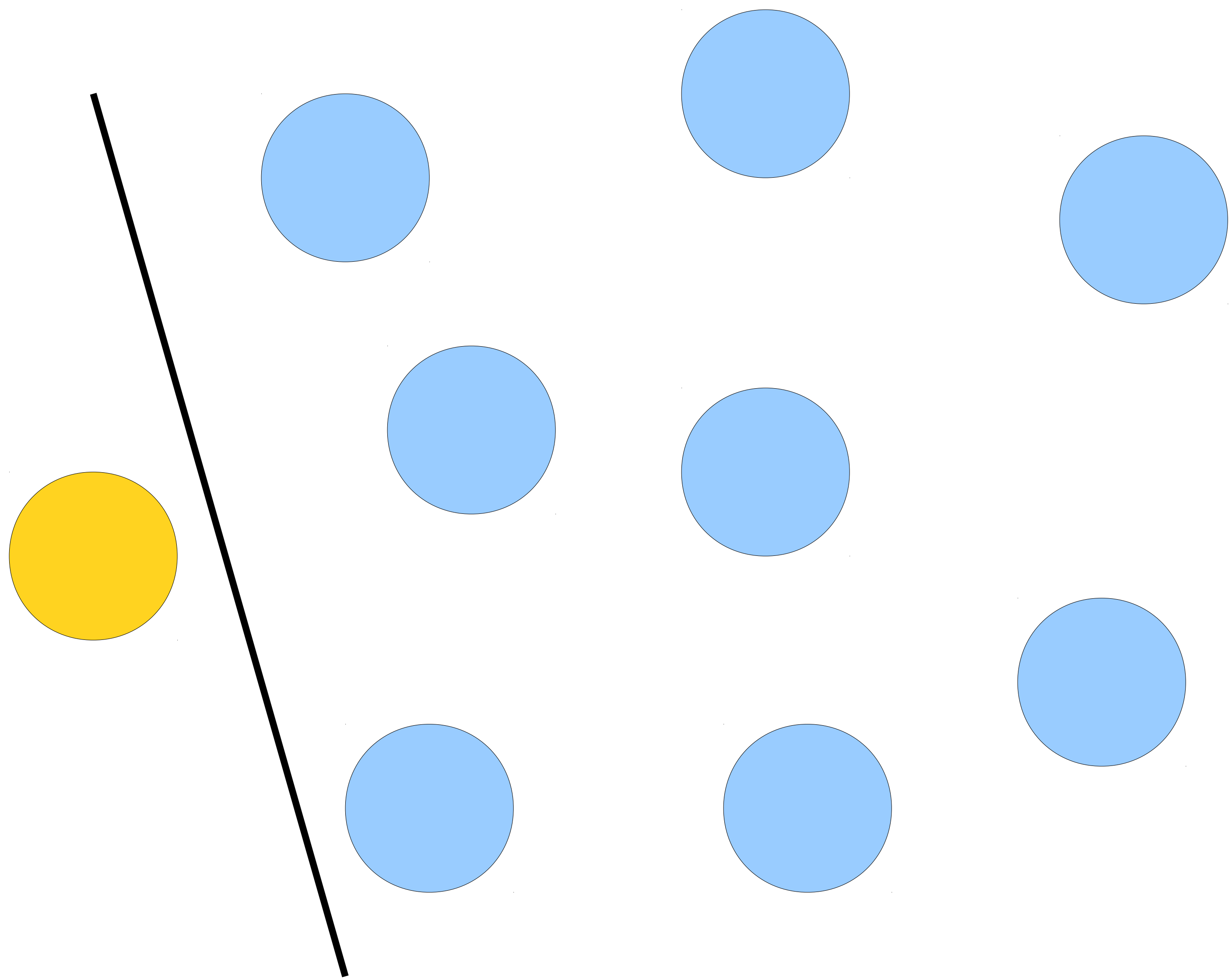


Generating Combinations

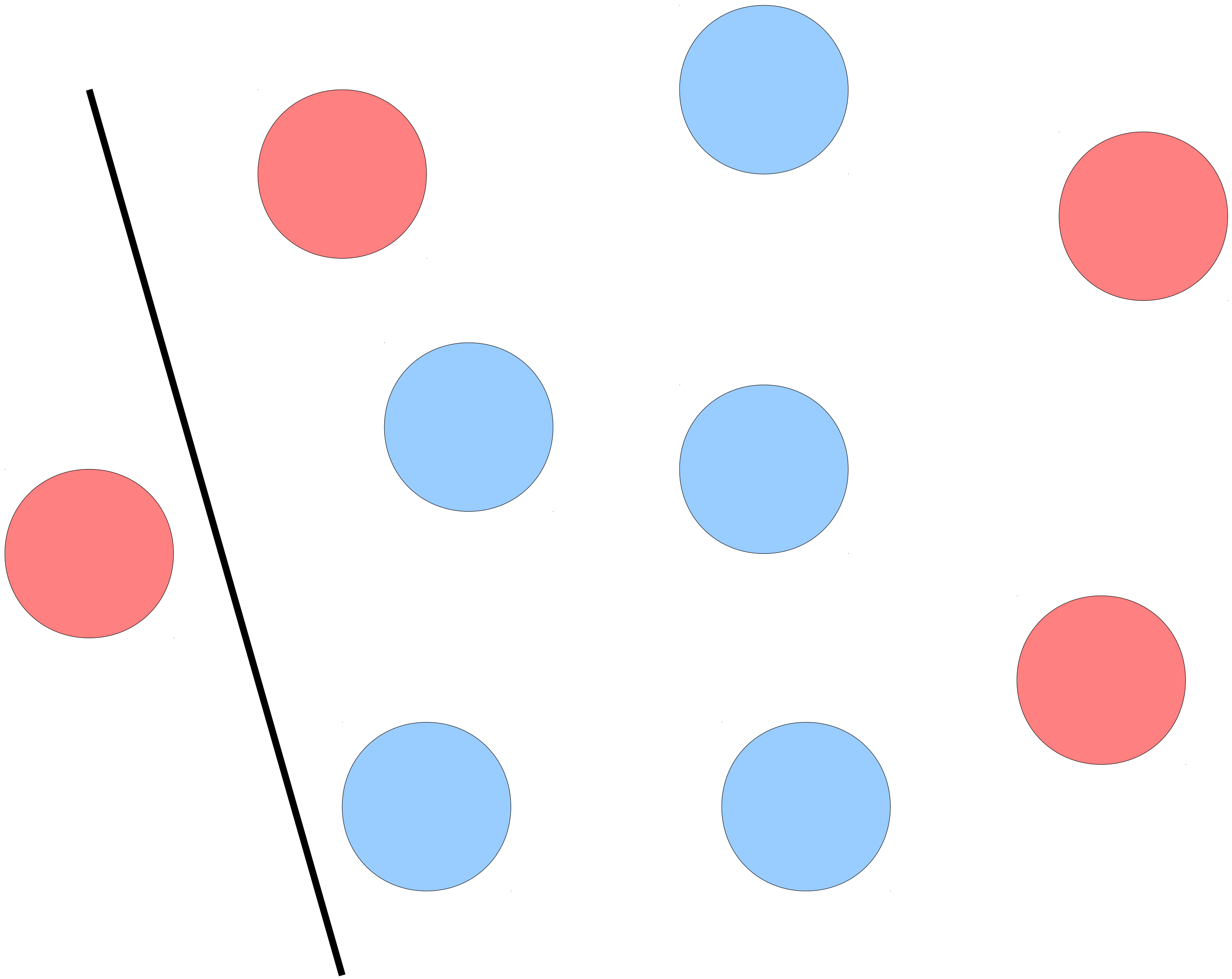


One way to choose **4** elements out of **9** is to choose **4** elements out of **8**.

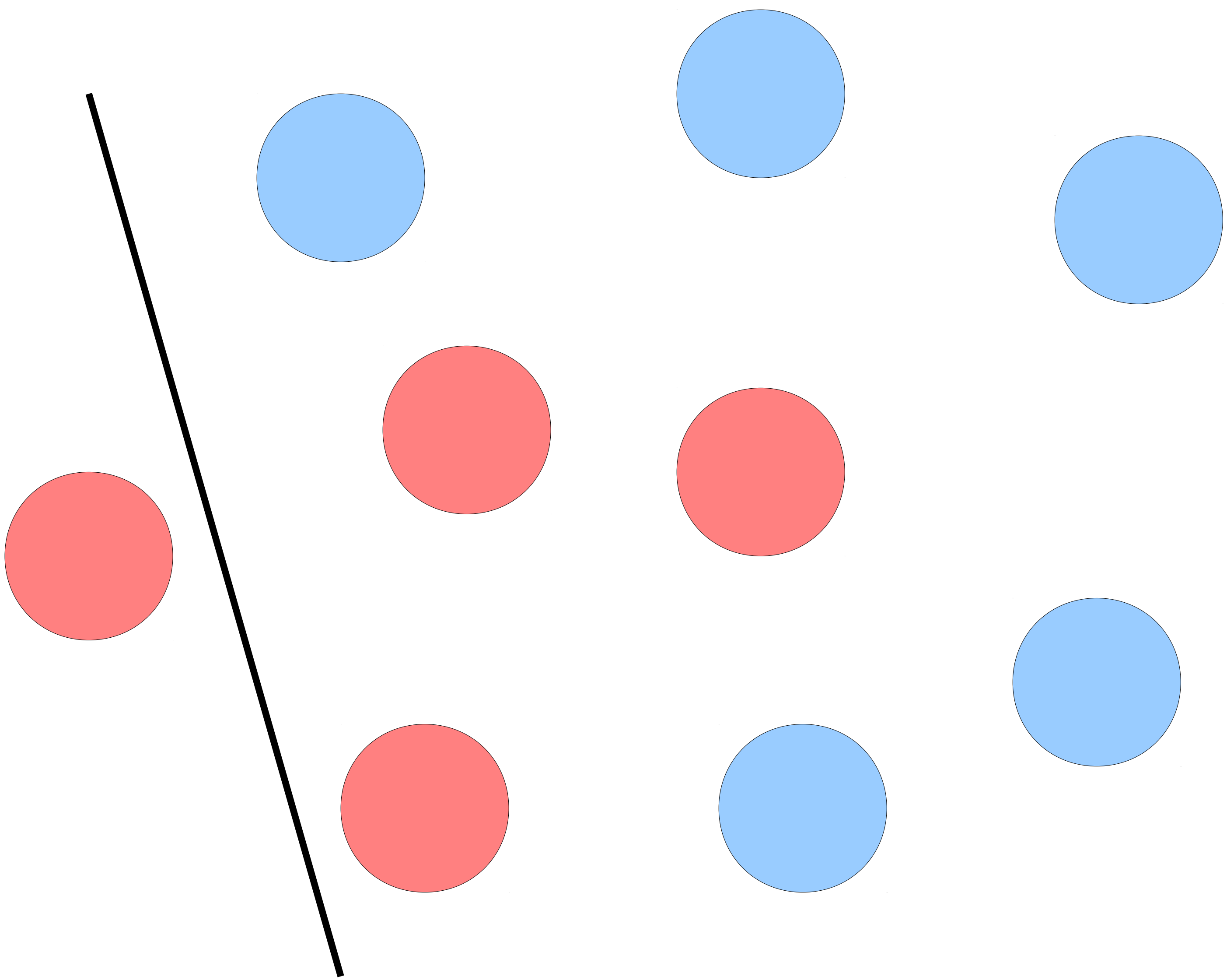
Generating Combinations



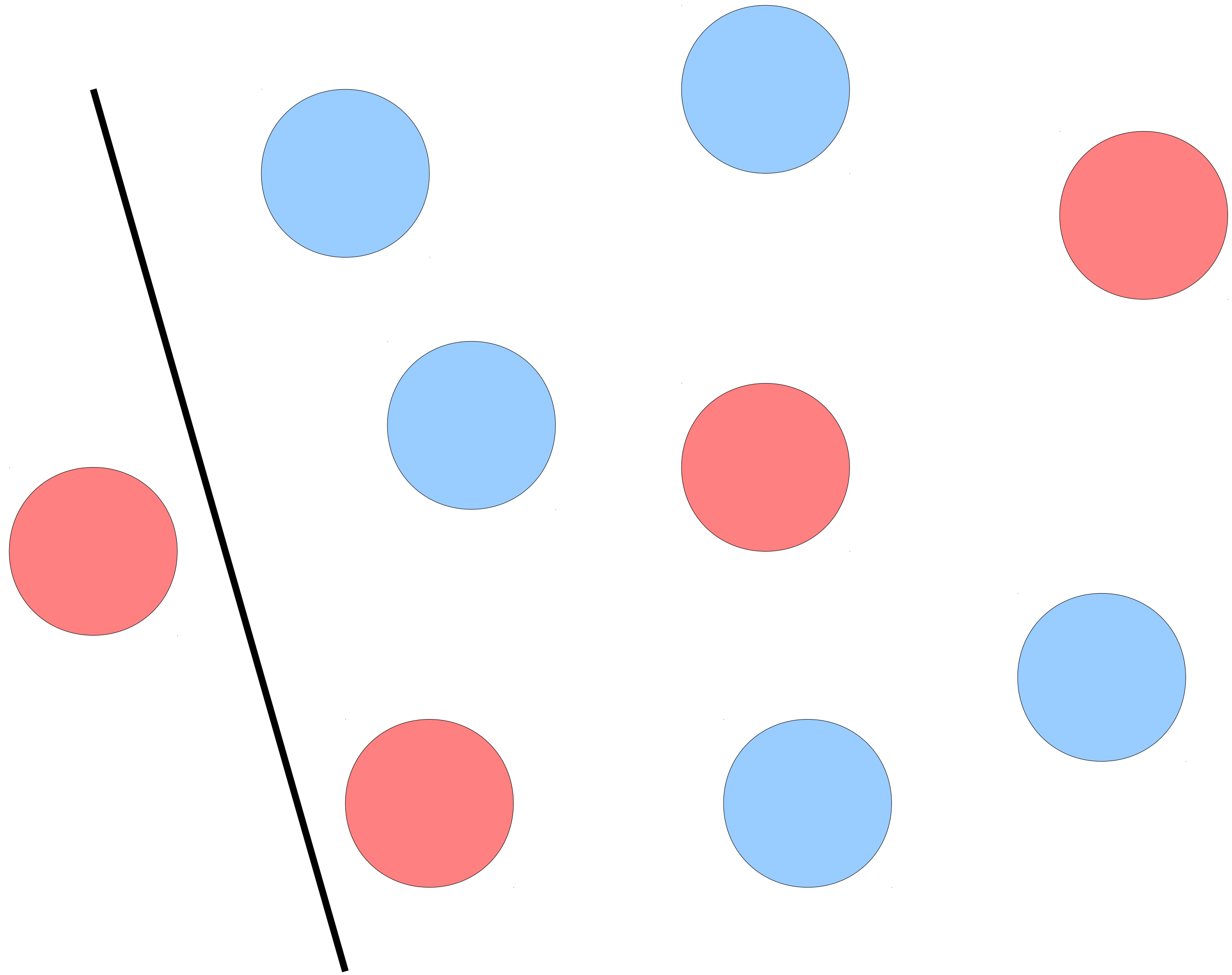
Generating Combinations



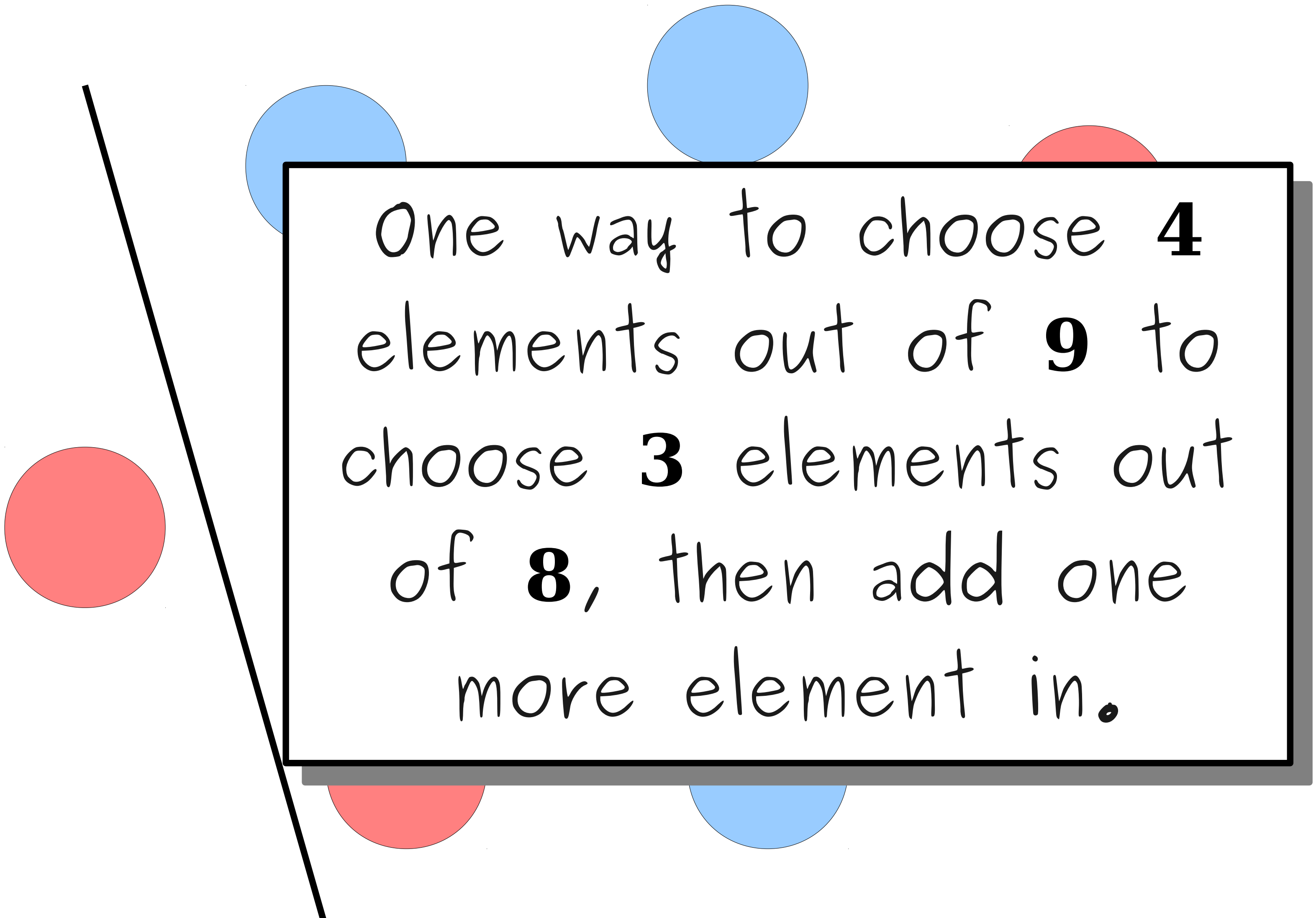
Generating Combinations



Generating Combinations



Generating Combinations



One way to choose **4** elements out of **9** to choose **3** elements out of **8**, then add one more element in.

Pascal's Triangle Revisited

					1					
				1		1				
			1		2		1			
		1		3		3		1		
	1		4		6		4		1	
1		5		10		10		5		1

Pascal's Triangle Revisited

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

Pascal's Triangle Revisited

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

Pascal's Triangle Revisited

(0, 0)

(0, 1) (1, 1)

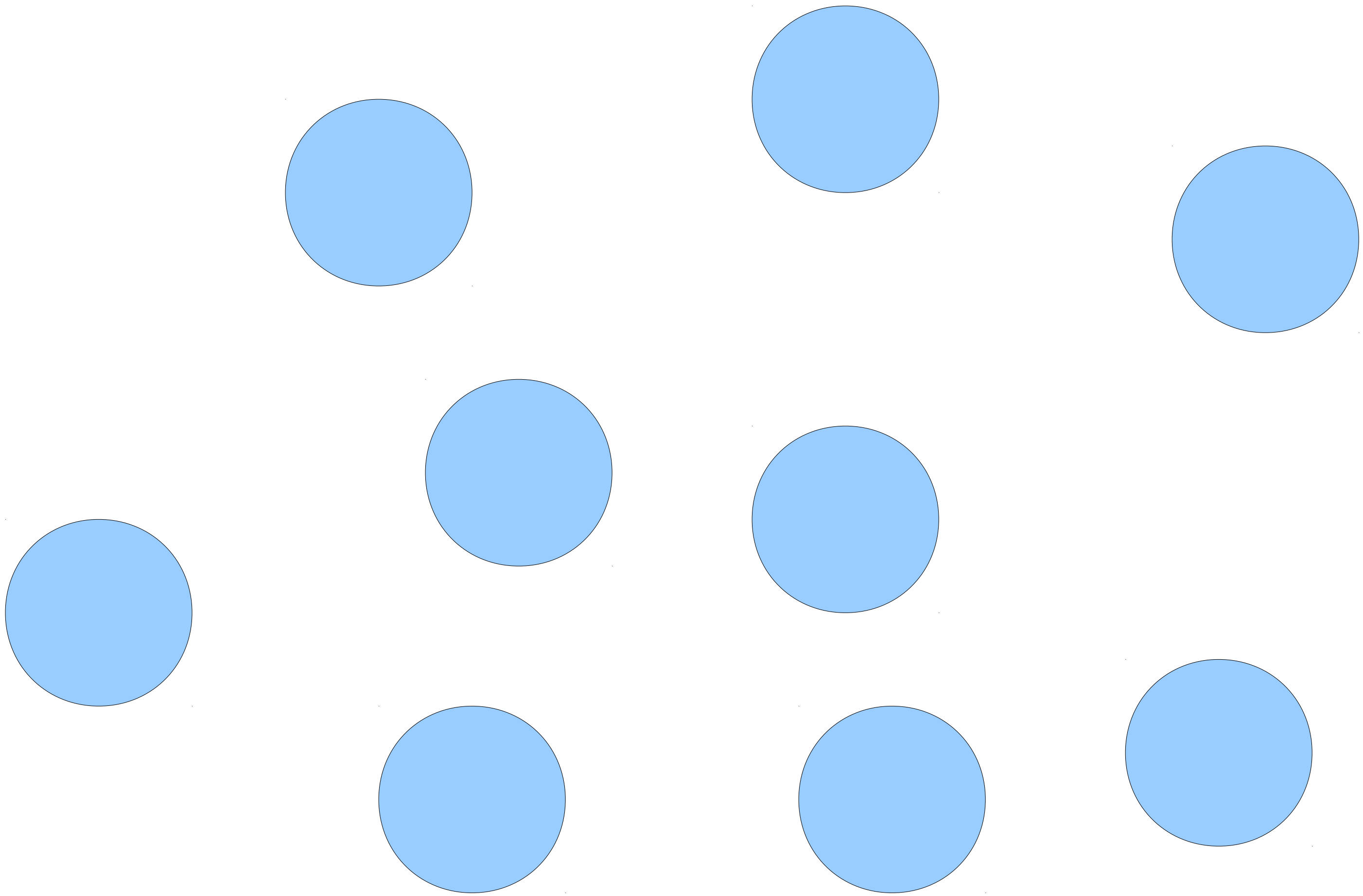
(0, 2) (1, 2) (2, 2)

(0, 3) (1, 3) (2, 3) (3, 3)

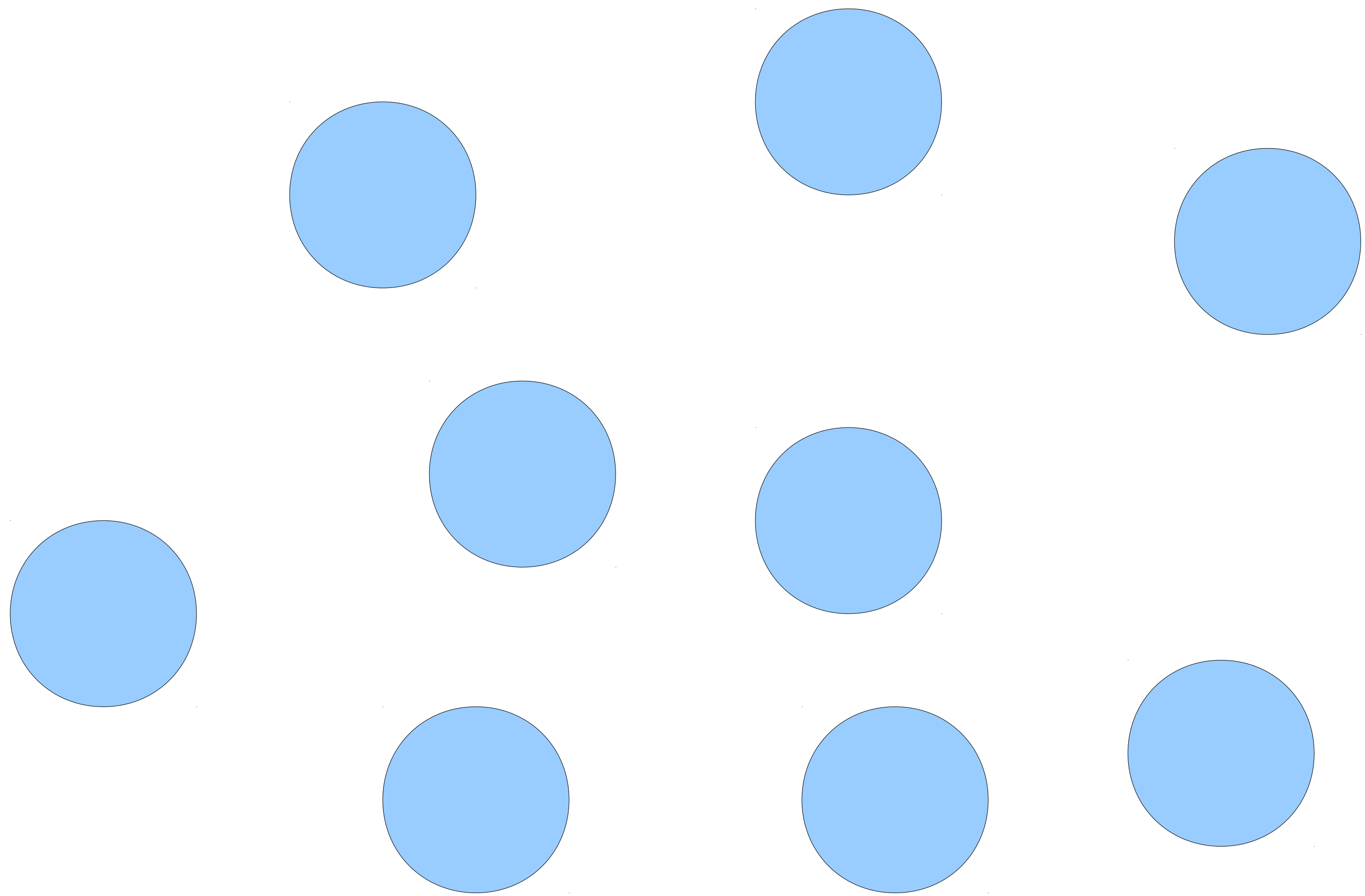
(0, 4) (1, 4) (2, 4) (3, 4) (4, 4)

(0, 5) (1, 5) (2, 5) (3, 5) (4, 5) (5, 5)

Generating Combinations



Generating Combinations



How many ways are there to pick 0 things from this set?

Generating Combinations

Generating Combinations

How many ways are there to pick 10 things from this set?

Generating Combinations

How many ways are there to pick 0 things from this set?

Combinations, Recursively

- How to pick k elements from a set?
- **Base Cases:**
  - If $k = 0$, there's exactly one set we can pick – namely, the empty set.
  - Otherwise, if the set is empty, there are no subsets.
- **Recursive Step:**
  - Pick some element x from the set.
  - Find all ways of picking k elements of what remains.
  - Find all ways of picking $k – 1$ elements of what remains, then add x back in.

Quick… to the codemobile!

A Pattern

- When generating subsets, permutations, and combinations, our recursive decomposition was
  - Remove some element.
  - Recursively process the rest.
  - Add that element back in.
- Many recursive functions are written this way.

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?

Shrinkable Words

- Let's call a word with this property a **shrinkable word**.
- Anything that isn't a word isn't a shrinkable word.
- Any single-letter word is shrinkable
  - A, I, O
- Any multi-letter word is shrinkable if you can remove a letter to form a word, and that word itself is shrinkable.
- So how many shrinkable words are there?

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.

Recursive Backtracking

```
if (problem is sufficiently simple) {
    return whether or not the problem is solvable
} else {
    for (each choice) {
        try out that choice.
        if it succeeds, return success.
    }
    return failure
}
```

Failure in Backtracking

S	T	A	R	T	L	I	N	G

Failure in Backtracking

S	T	A	R	T	L	I	N	G

S	T	A	R	T	L	I	G

Failure in Backtracking

S	T	A	R	T	L	I	N	G

S	T	A	R	T	L	I	G

Failure in Backtracking

S	T	A	R	T	L	I	N	G

Failure in Backtracking

S	T	A	R	T	L	I	N	G

S	T	A	R	T	I	N	G

Failure in Backtracking

S T A R T L I N G

S T A R T I N G

S T R T I N G

Failure in Backtracking

S T A R T L I N G

S T A R T I N G

S T R T I N G

Failure in Backtracking

S	T	A	R	T	L	I	N	G
	S	T	A	R	T	I	N	G
		S	T	A	R	I	N	G

Failure in Backtracking

- Returning false in recursive backtracking does **not** mean that the entire problem is unsolvable!
- Instead, it just means that the current subproblem is unsolvable.
- Whoever made the call to this function can then try other options.
- Only when all options are exhausted can we know that the problem is unsolvable.