Unit 8: Recursion

Dave Abel

April 8th, 2016

Outline For Today

- Quick Recap and Factorial Warmup
- Recursion, Theory
- Prefix notation
- Recursive Searching!
- Recursive Sorting!

Recursion: Recap

- **Definition**: a process, program, or object is said to be *recursive* if it involves repeated self-reference.

Recursion: Recap

- **Definition**: a process, program, or object is said to be *recursive* if it involves repeated self-reference.

- In general, *recursive entities can be described as:*
  - **A simple step**
  - **A recursive step**

- Recursion can be *infinite*

- For recursion to be *finite*, we need a **base case.**

Recursive Algorithms

- Factorial
- Word Length
- Is a word a palindrome?
- Fibonacci

Recursion: Factorial

factorial(4)

Recursion: Factorial

factorial(4) = 4 * factorial(3)

Recursion: Factorial

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

Recursion: Factorial

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

factorial(2) = 2 * factorial(1)

Recursion: Factorial

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

factorial(2) = 2 * factorial(1)

factorial(1) = 1

Recursion: Factorial

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

factorial(2) = 2 * factorial(1)

factorial(1) = 1

Recursion: Factorial

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

factorial(2) = 2 * 1

Recursion: Factorial

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * 2

Recursion: Factorial

factorial(4) = 4 * 6

Recursion: Factorial

factorial(4) = 24

Recursion: Factorial

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

factorial(2) = 2 * factorial(1)

factorial(1) = 1

Recursion!

So *some* algorithms can be recursive….

Recursion!

computing length of word

So *some* algorithms can be recursive….

computing factorial

Recursion!

computing length of word

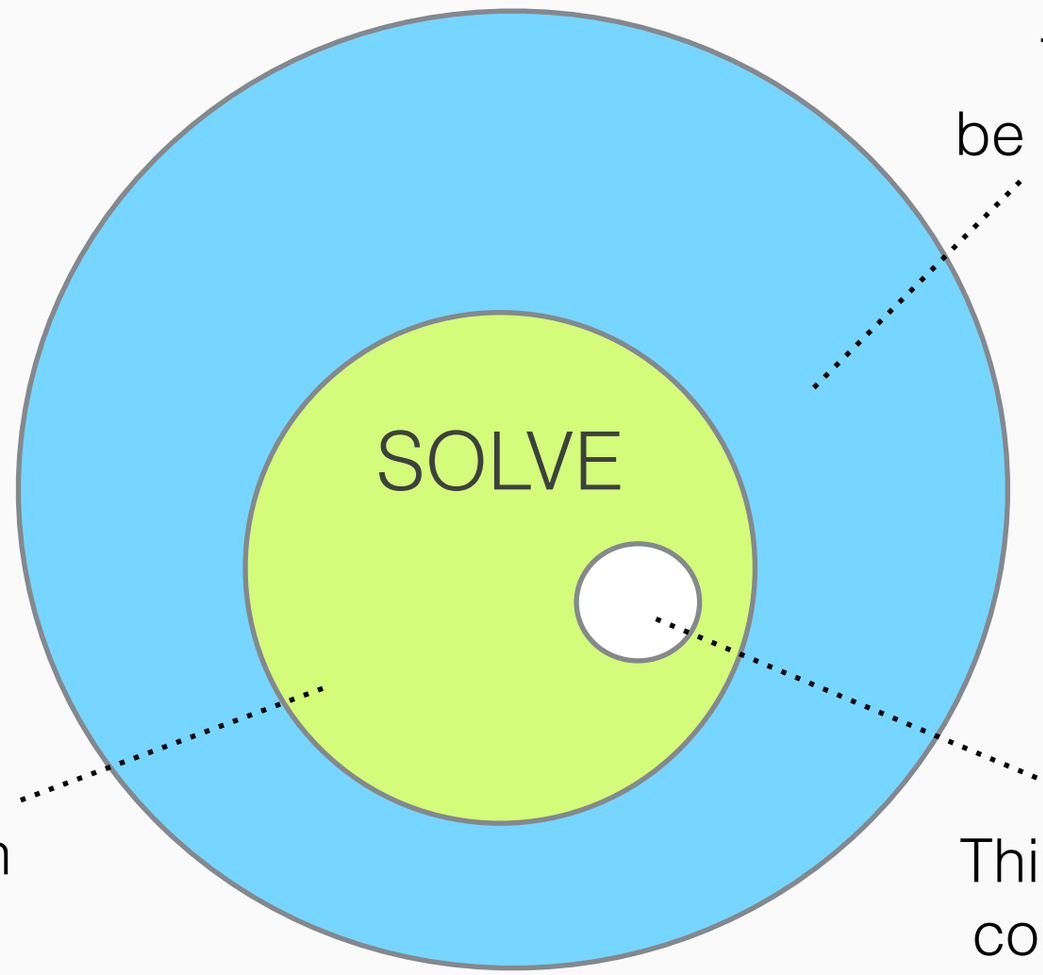
is word a palindrome?

So *some* algorithms can be recursive….

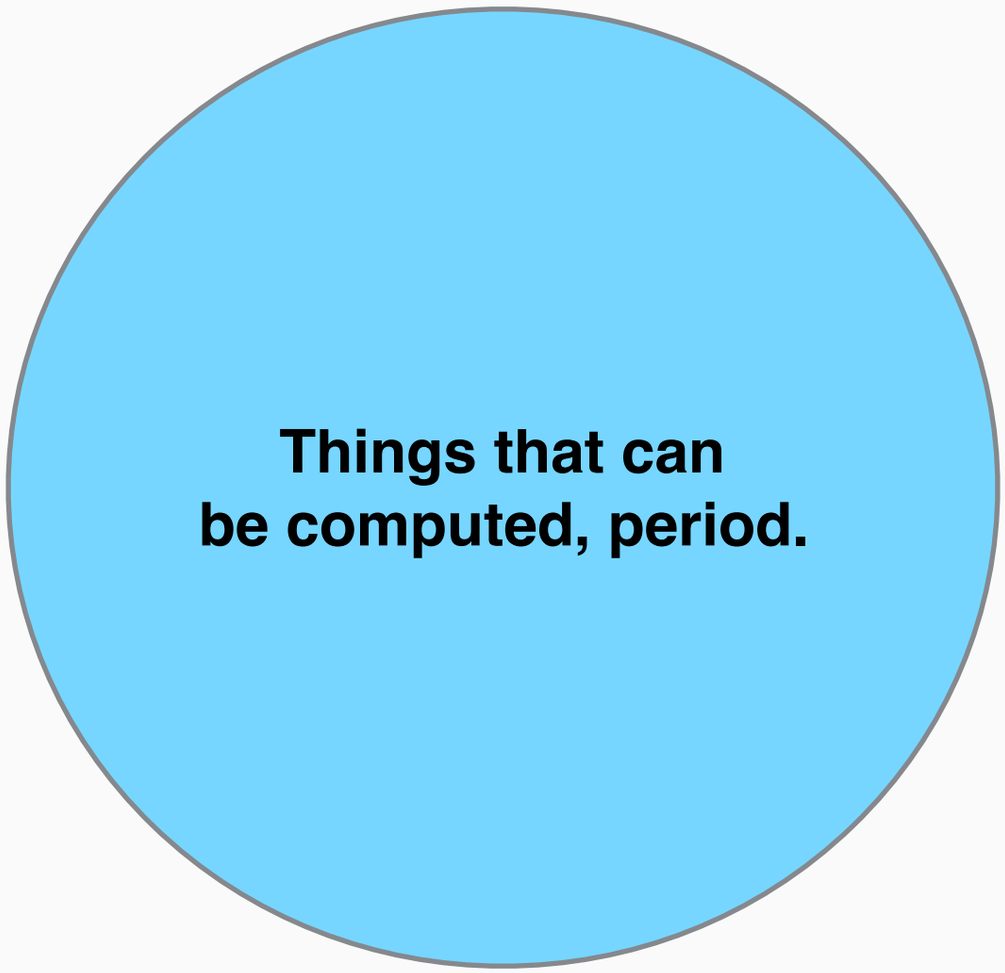
linear search

computing factorial

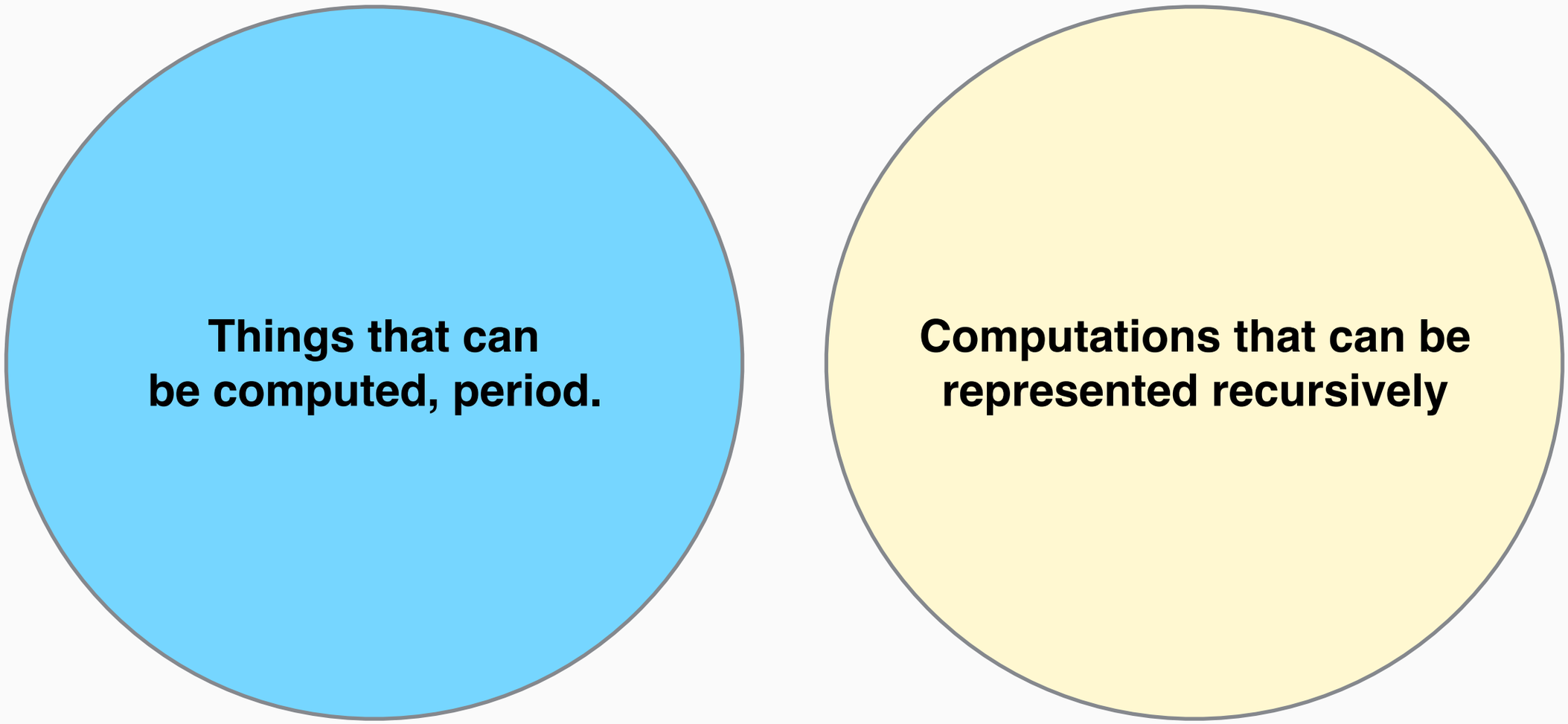
Cool Connection to Theory



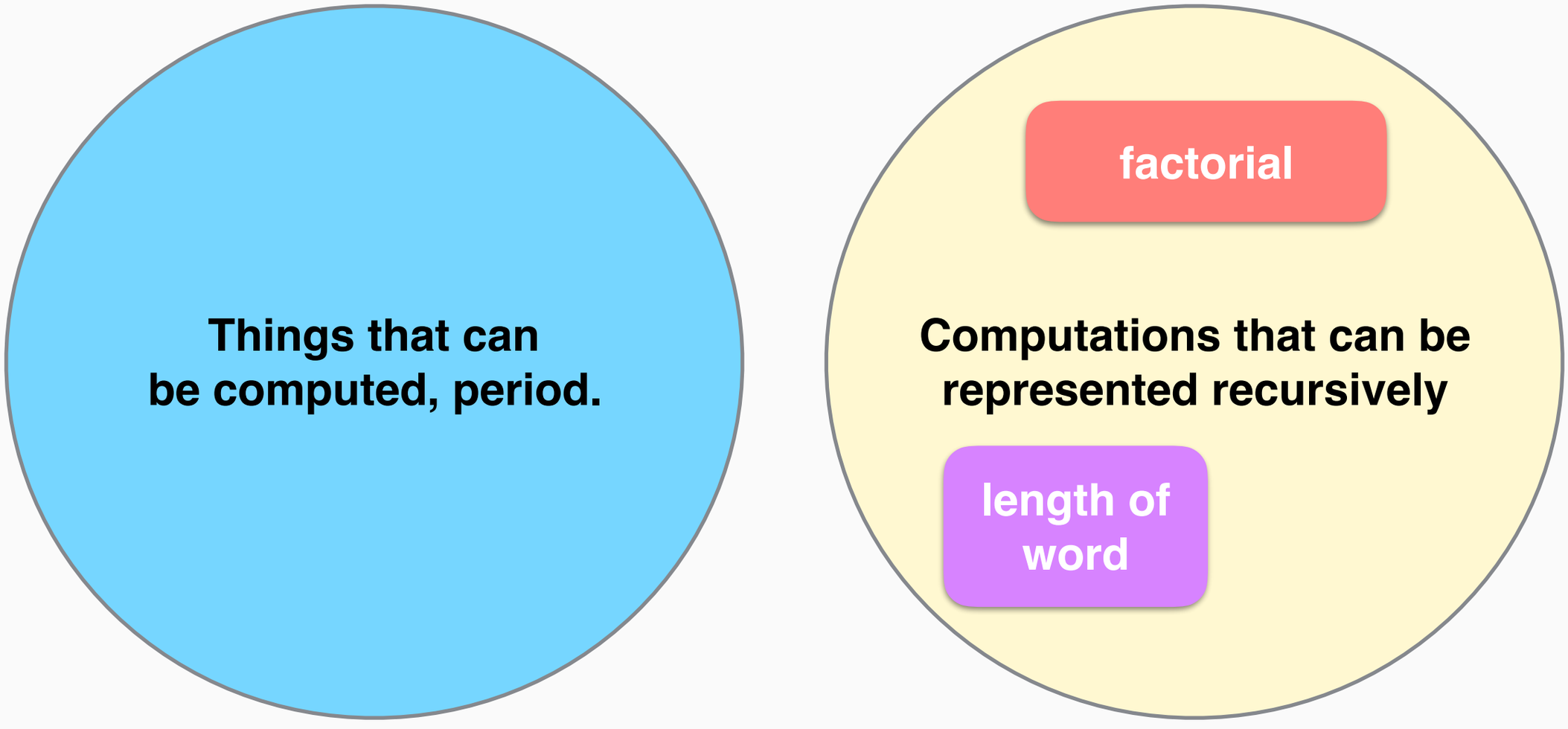
Cool Connection to Theory



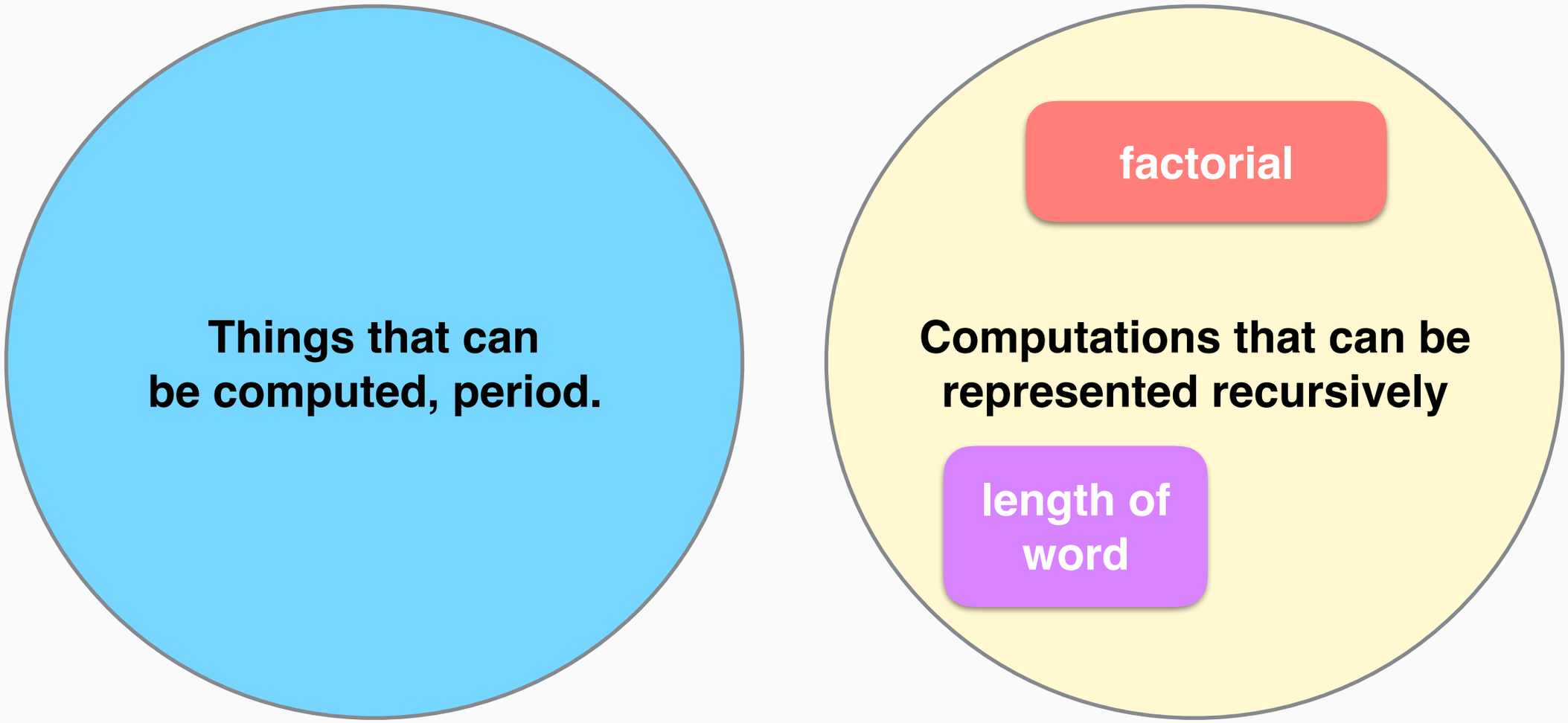
Cool Connection to Theory



Cool Connection to Theory

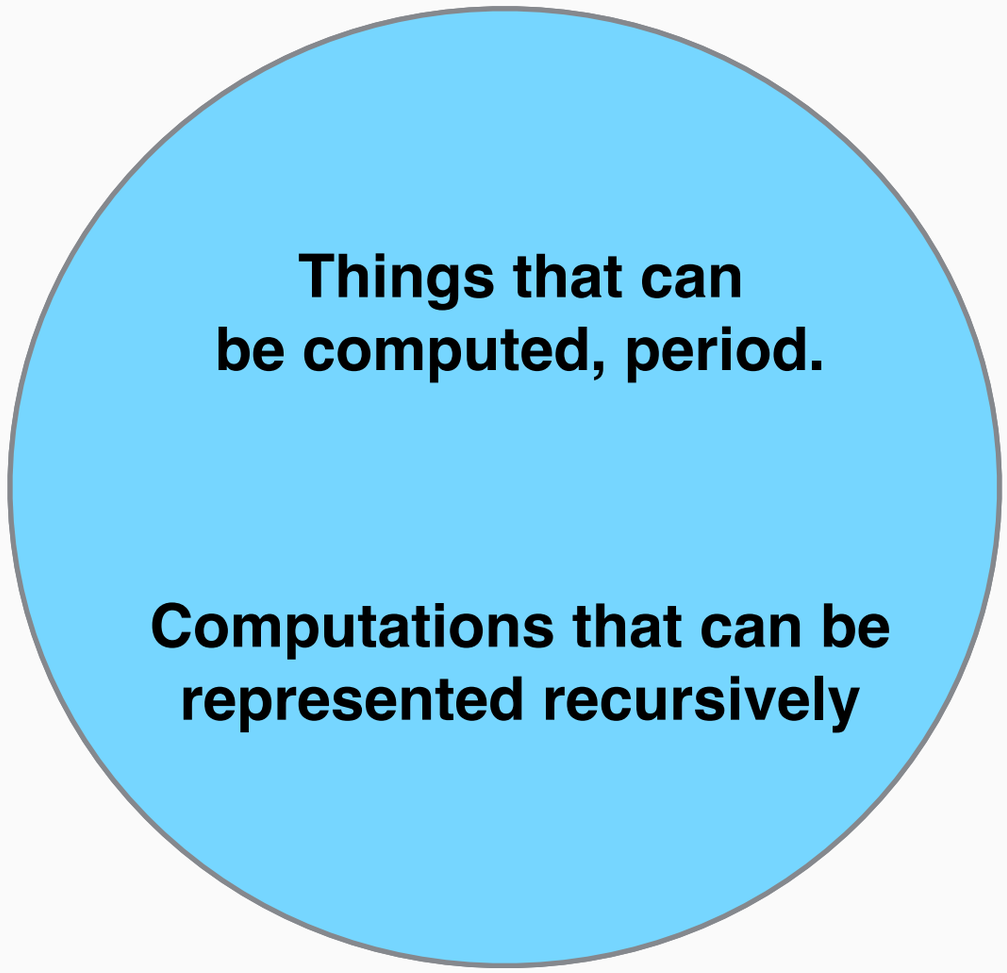


Cool Connection to Theory



Q: How do these two bubbles relate?

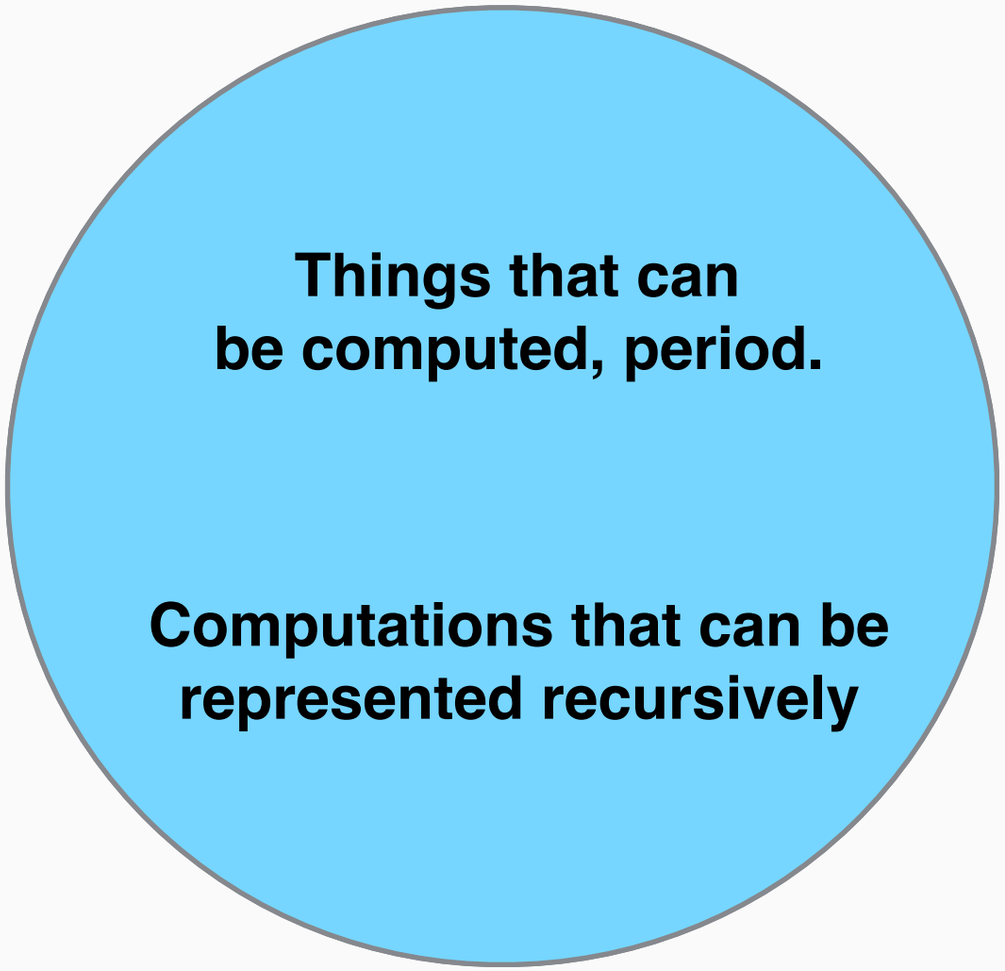
Cool Connection to Theory



Q: How do these two bubbles relate?

A: They're *identical…*

Cool Connection to Theory



If something *can't* be computed, it also *can't* be represented recursively

Q: How do these two bubbles relate?

A: They're *identical*

Prefix Notation

- **Idea:** another way of writing arithmetic that fits naturally into recursive solutions

- **Put the operator at the beginning:**
  - 5 + 7 becomes + 5 7
  - (5 + 7) * (3 + 2) becomes * + 5 7 + 3 2
  - * + 34 2 10 becomes (34 + 2) * 10

Clicker Question!

Q: What is the result of: * + 3 7 + 4 2?

Clicker Question!

Q: What is the regular notation of: * + 3 7 + * 2 4 2?

[A] (3*7) + (2*4) * 2

[B] (3+7) * (2+4) * 2

[C] (3+7) * (2+4) + 2

[D] (3+7) * ((2*4) + 2)

[E] I'm confused.

Our First Problem: Search

Problem Specification:

- *Input:*
  - a collection of objects, call it “Basket”
  - a specific object, call it “Snozzberry”

- *Output:*
  - True if “Snozzberry” is in “Basket”.
  - False if “Snozzberry” is *not* in “Basket”

Recursive Solution!

- Q: Can we do **linear** search recursively?
- Sure!
- Recursive Linear Search:
  - Is our list empty? If so, return false!
  - Is the first item in the list our item?
  - If not, run Recursive Linear Search on the rest of the list!

Recursive Solution!

- Q: Can we do **binary** search recursively?
- Sure! This one is perhaps *more* naturally recursive
- Recursive Binary Search:
  - Is our list empty? If so, return false!
  - Check the middle item, is it our item? If so, return True.
  - If not, run Binary Search on the correct half of the list.

Our Second Problem: Sorting

Problem Specification:

- *Input:*
  - a collection of *orderable* objects, call it “Basket”
- *Output:*
  - “Basket”, where each item is in order.

Recursive Solution!

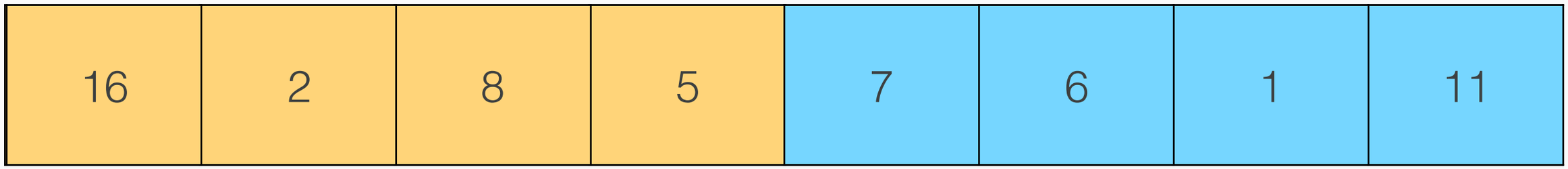
- Q: Can we do **sort** recursively?
- Sure! There are many ways. Lets talk about one.
- Merge Sort:
  - Split the list in half, merge sort each half.
  - Merge the two together.

Merge Sort

16	2	8	5	7	6	1	11

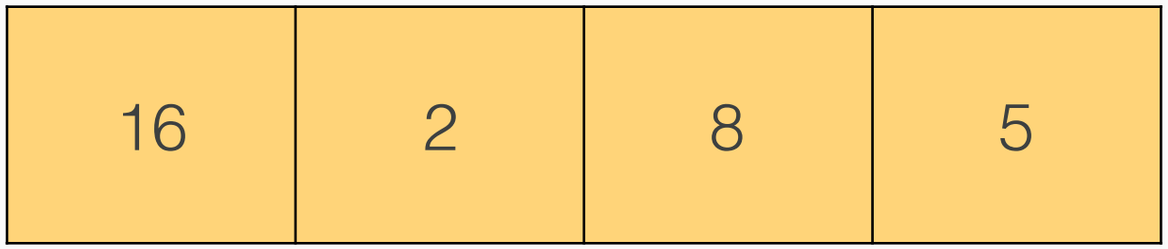
- Split the list in half, merge sort each half.
- Merge the two together.

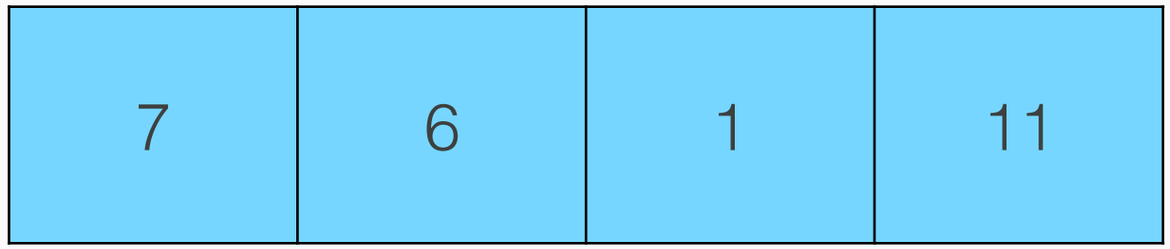
Merge Sort



- **Split the list in half,** merge sort each half.
- Merge the two together.

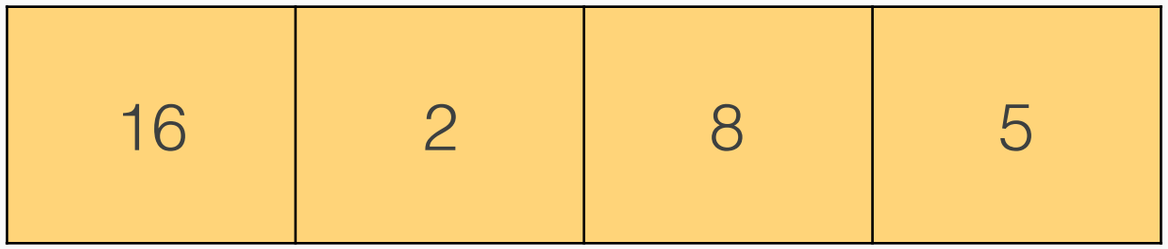
Merge Sort

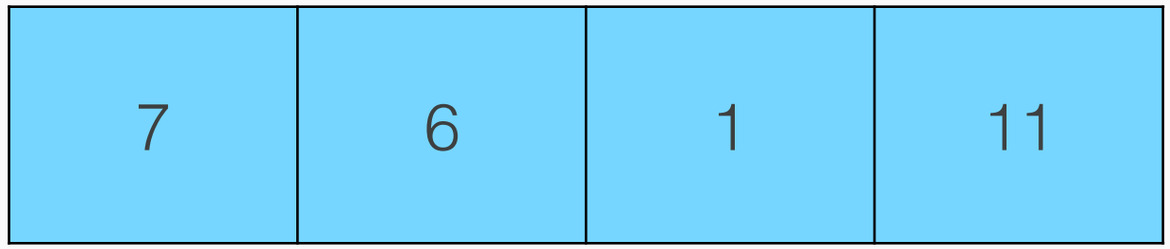




- Split the list in half, merge sort each half.
- Merge the two together.

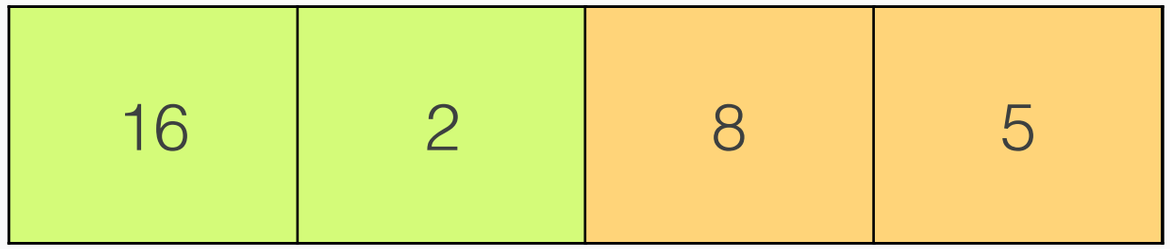
Merge Sort

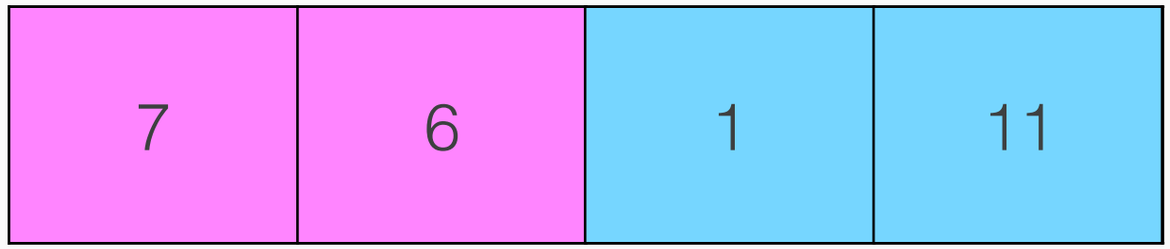




- Split the list in half, **merge sort** each half.
- Merge the two together.

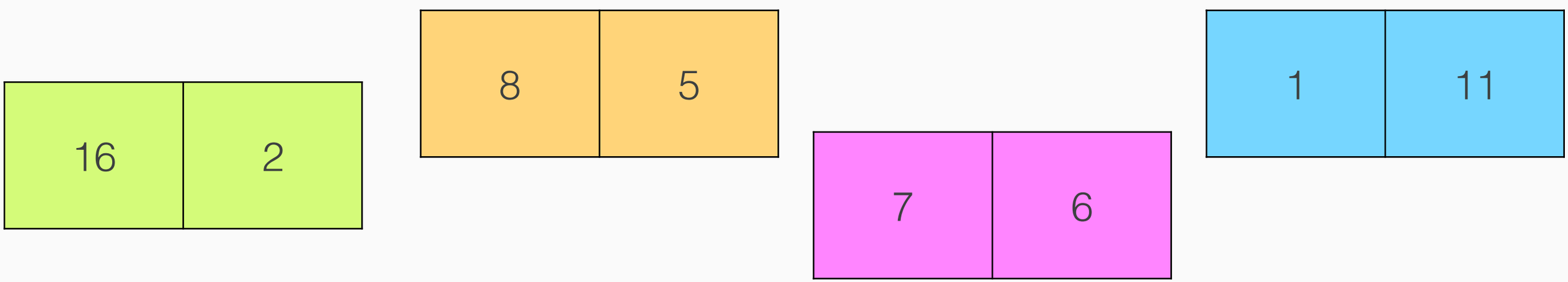
Merge Sort





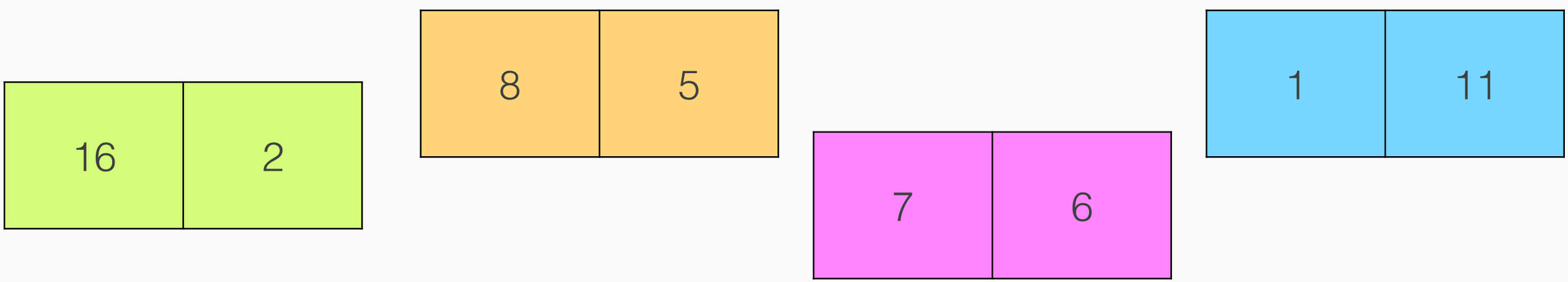
- **Split the list in half,** merge sort each half.
- Merge the two together.

Merge Sort



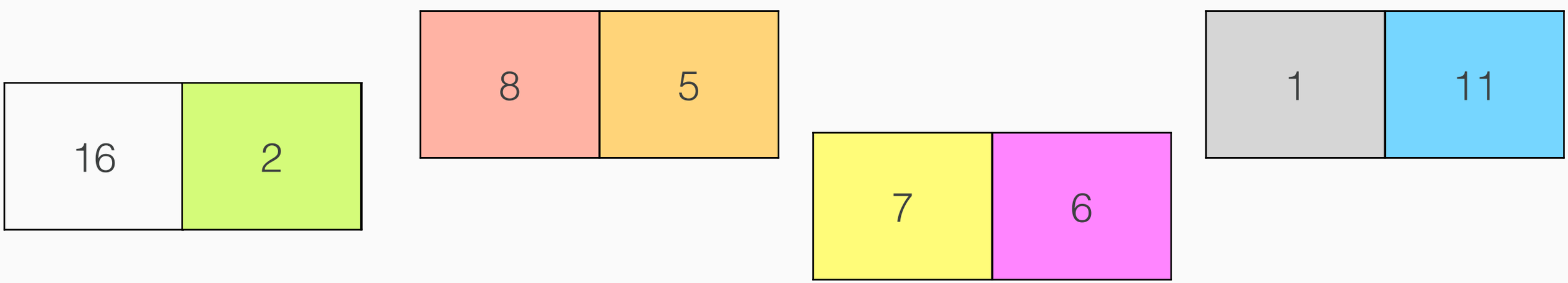
- **Split the list in half,** merge sort each half.
- Merge the two together.

Merge Sort



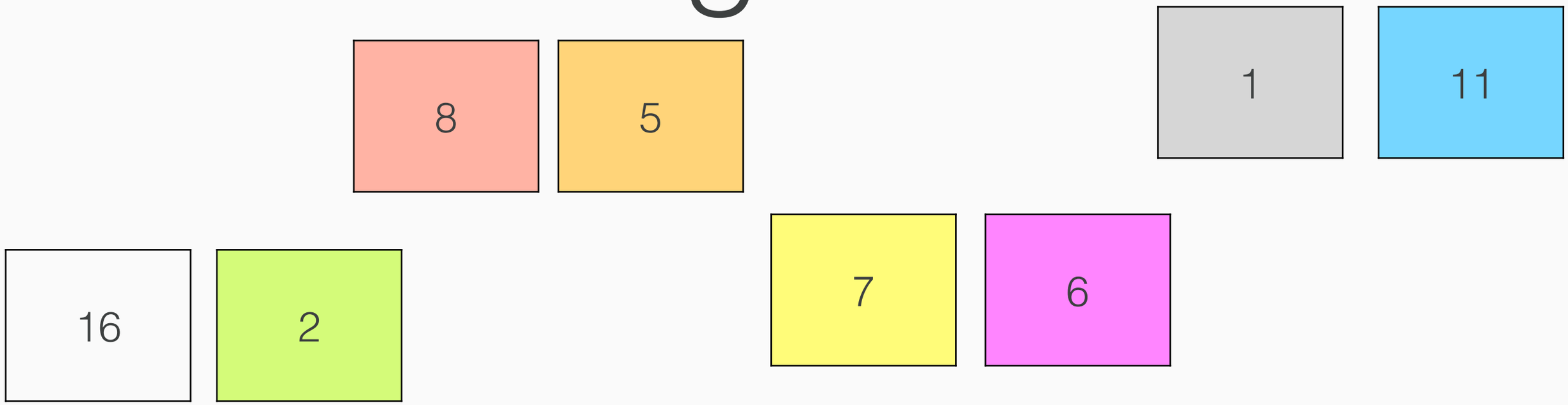
- Split the list in half, **merge sort** each half.
- Merge the two together.

Merge Sort



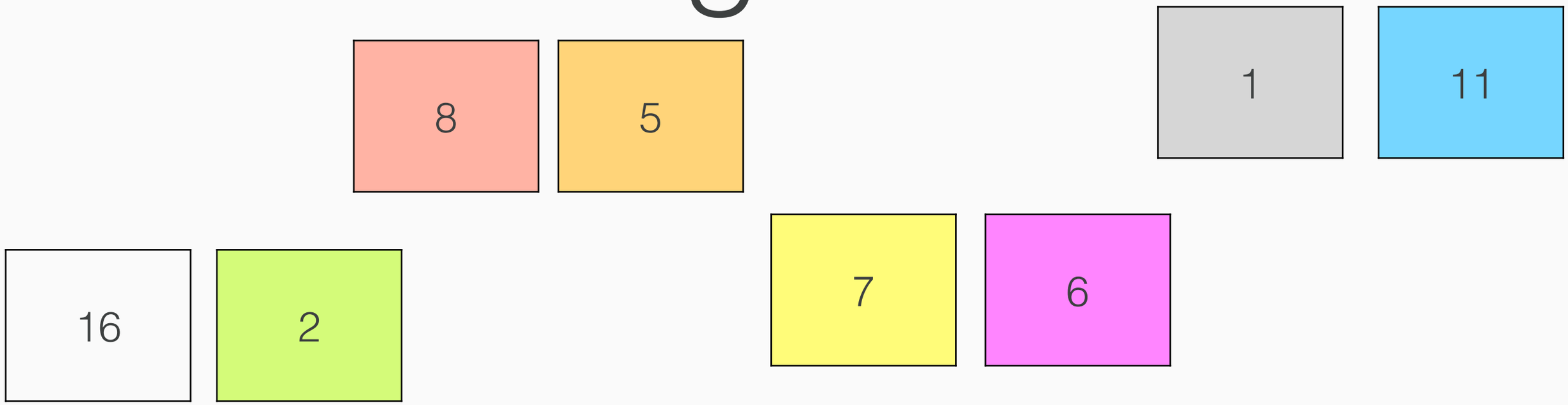
- **Split the list in half,** merge sort each half.
- Merge the two together.

Merge Sort



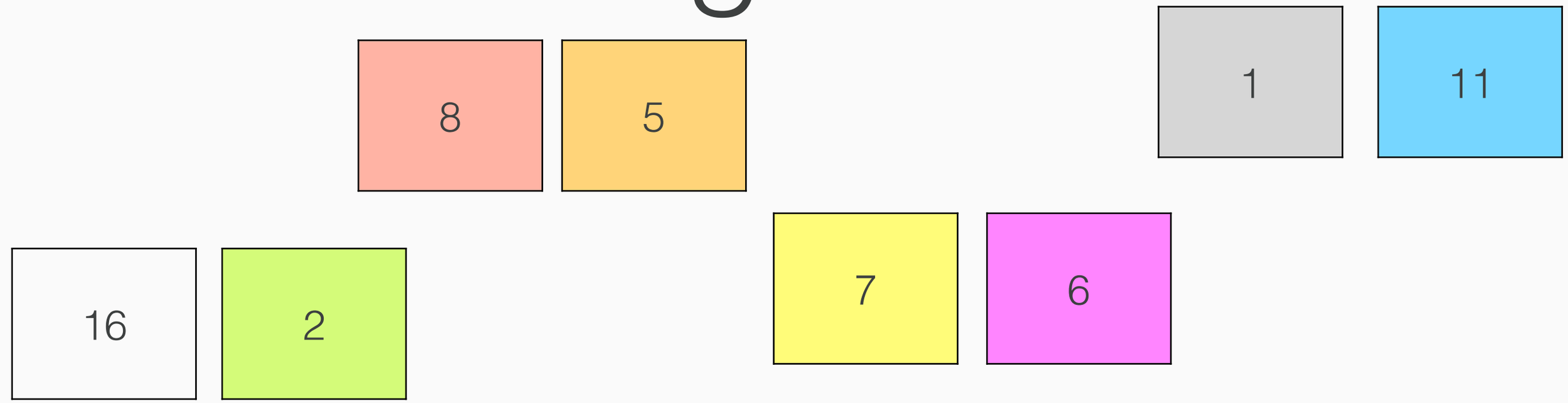
- **Split the list in half,** merge sort each half.
- Merge the two together.

Merge Sort



- Split the list in half, **merge sort** each half.
- Merge the two together.

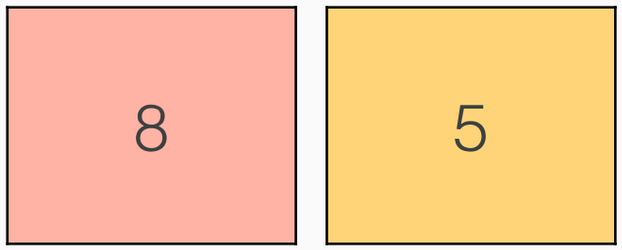
Merge Sort

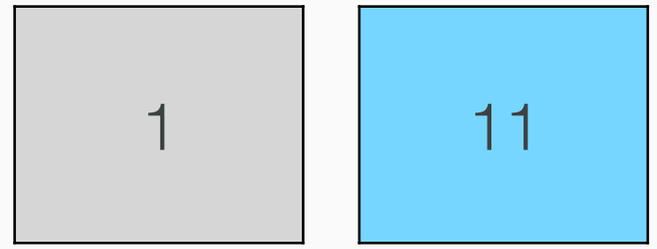


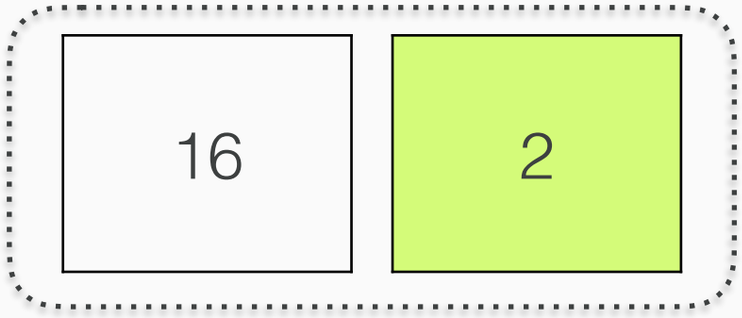
Sorting a length 1 list is our trivial case!

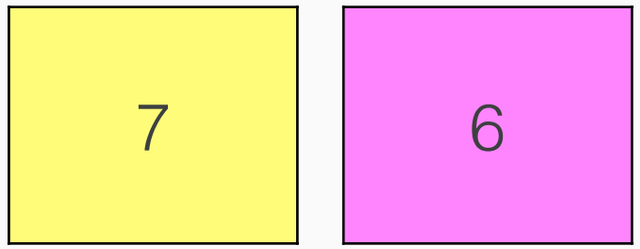
- Split the list in half, merge sort each half.
- Merge the two together.

Merge Sort



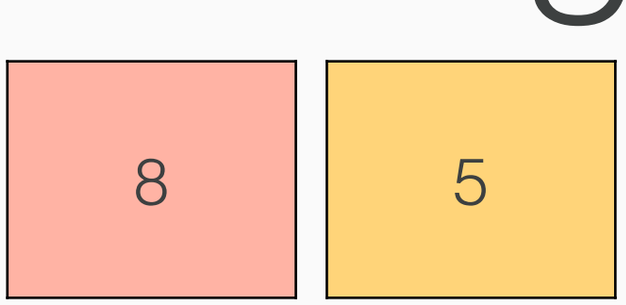


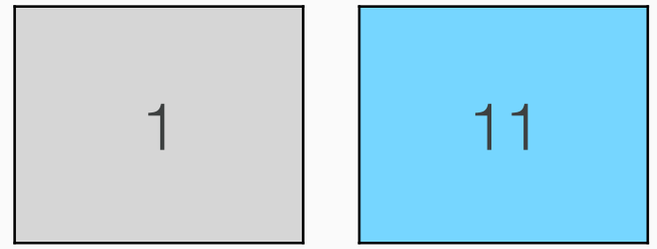


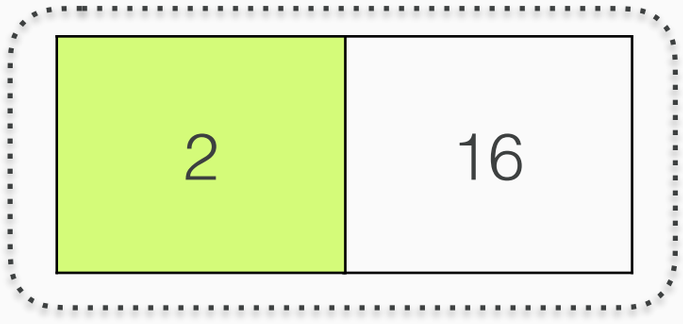


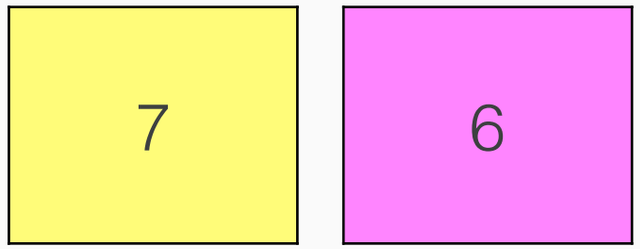
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



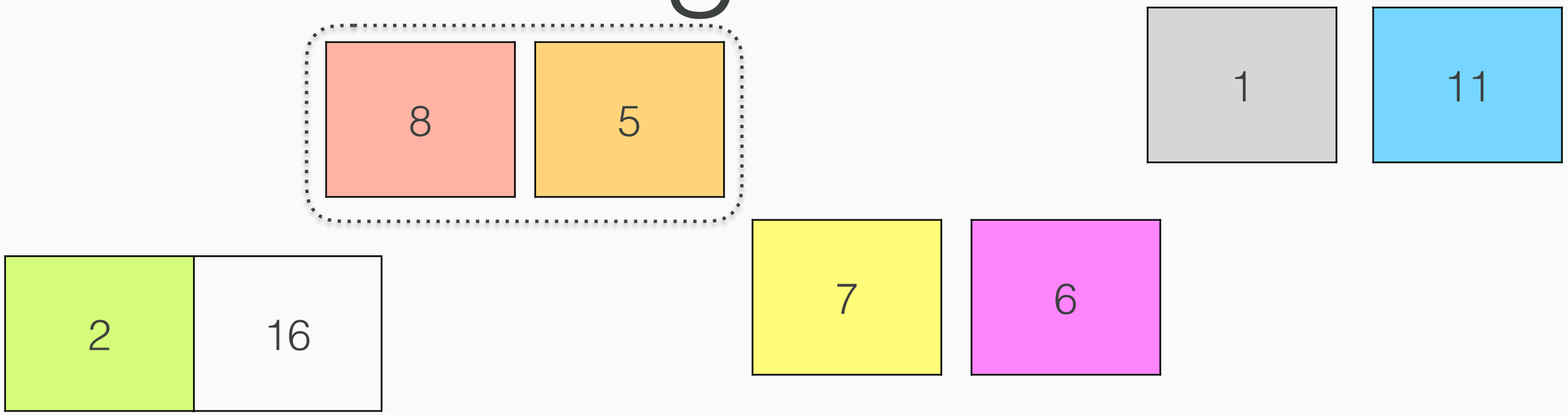






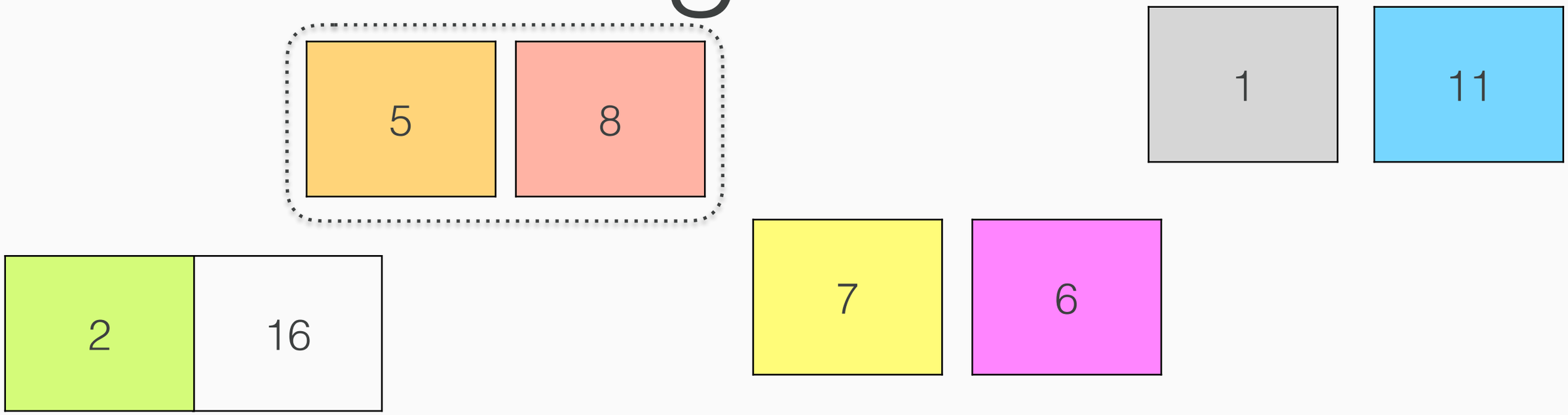
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



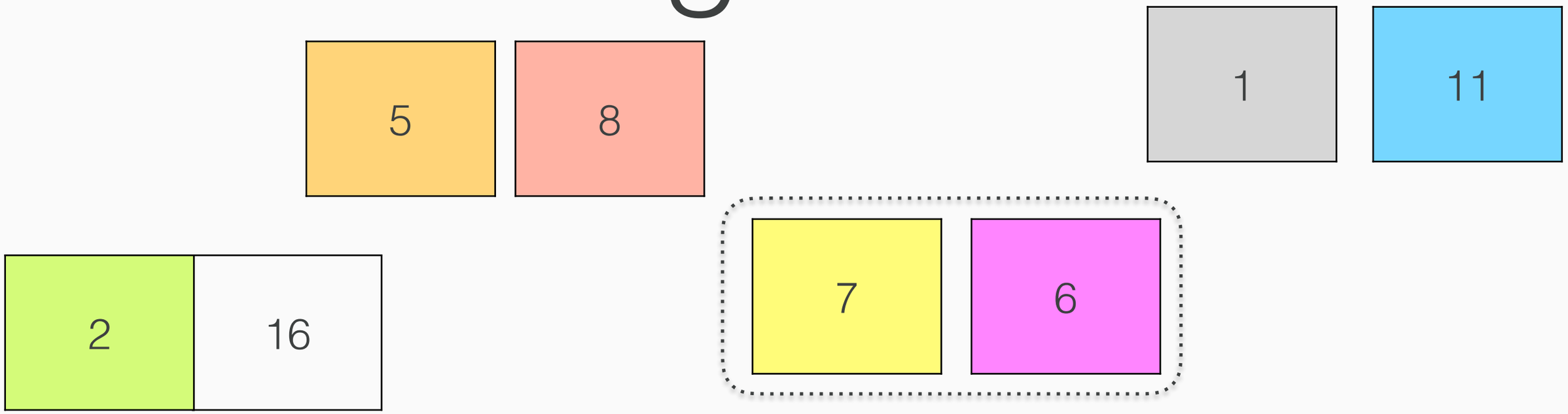
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



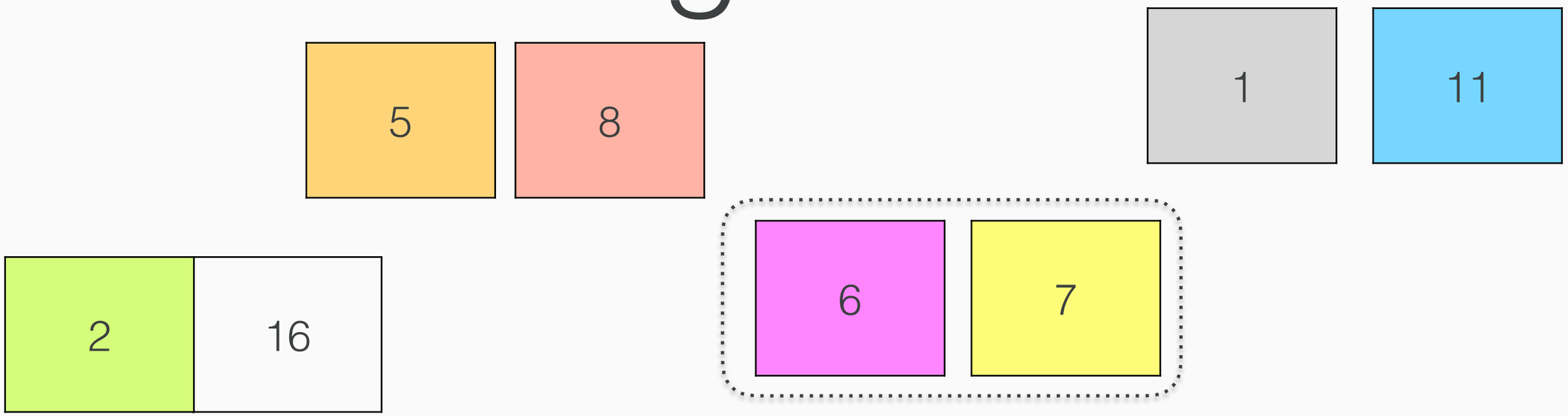
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



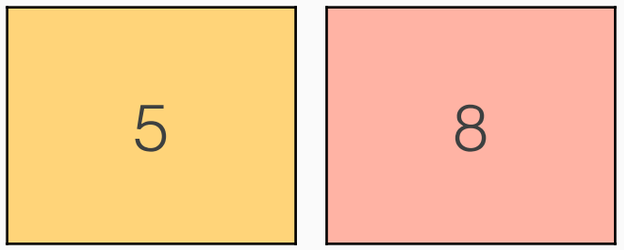
- Split the list in half, merge sort each half.
- **Merge the two together.**

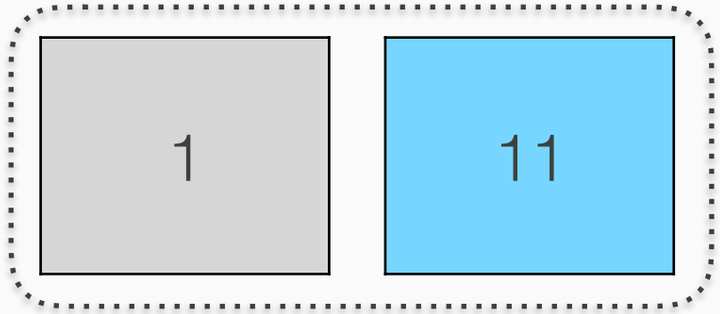
Merge Sort

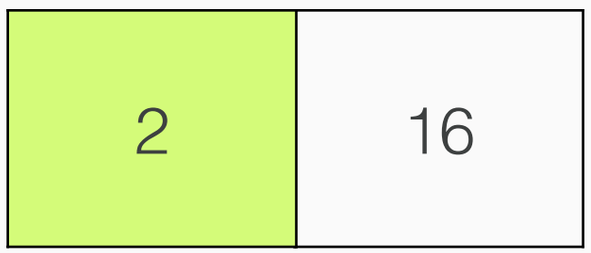


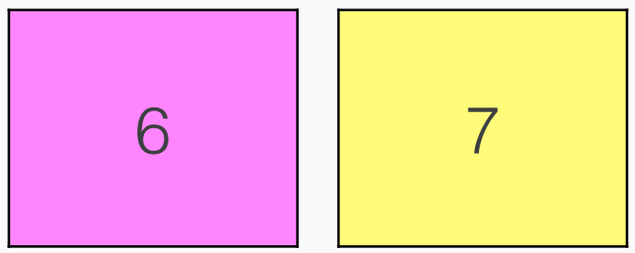
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



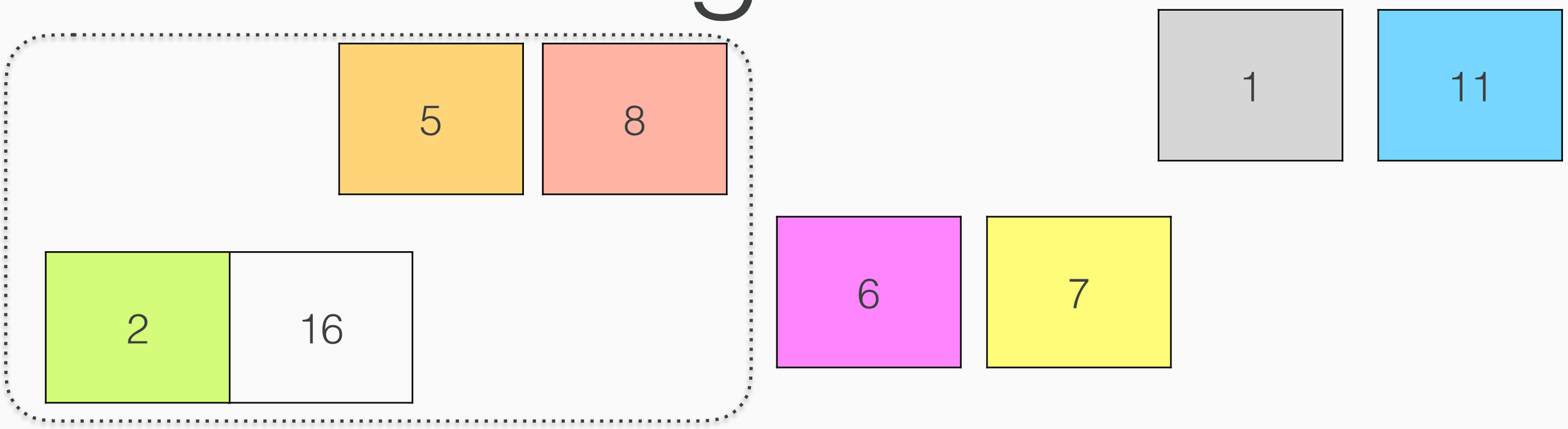






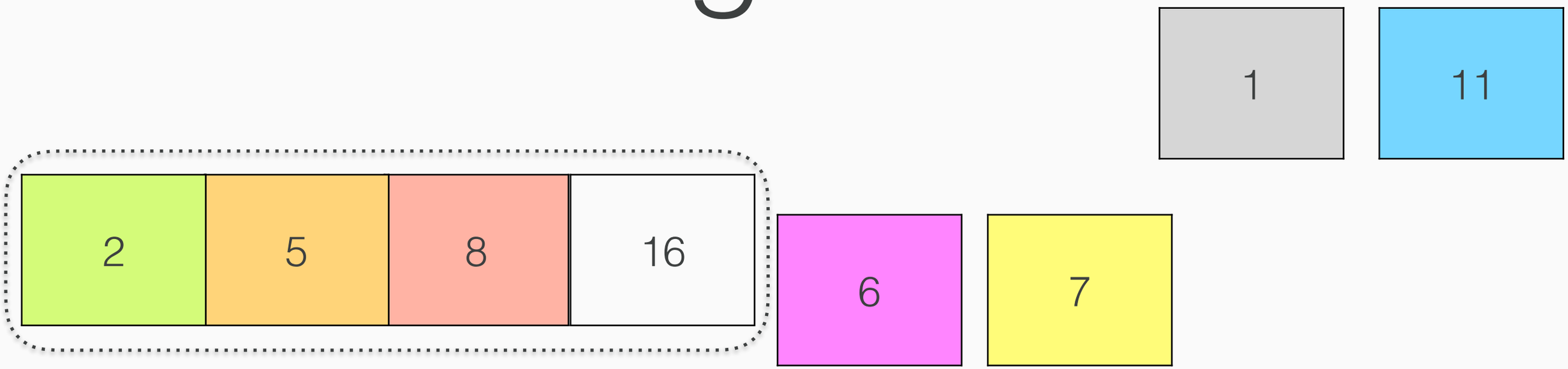
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



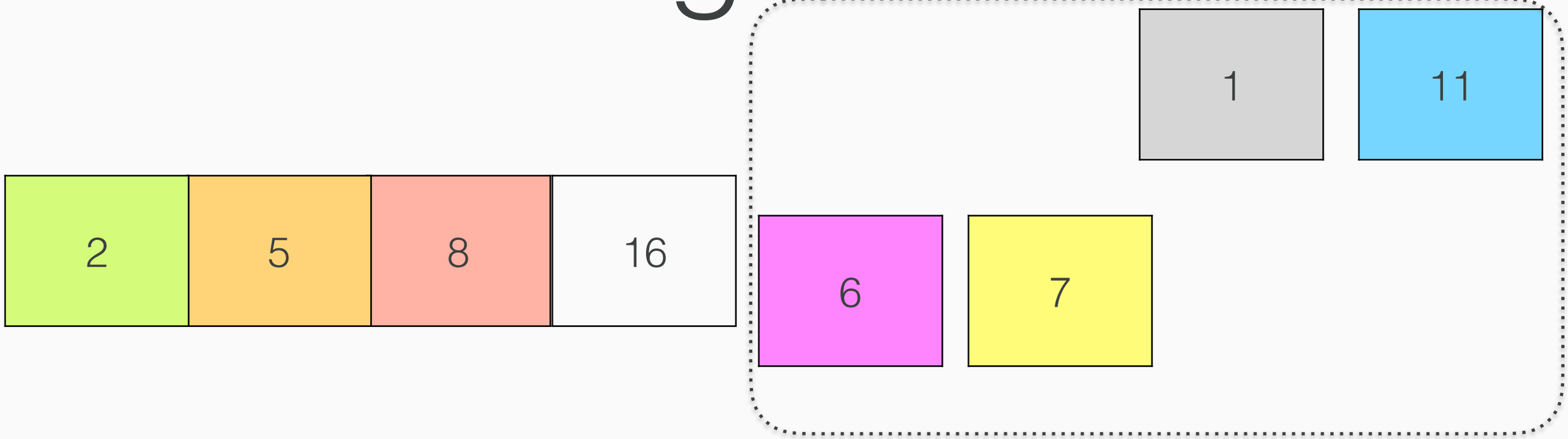
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



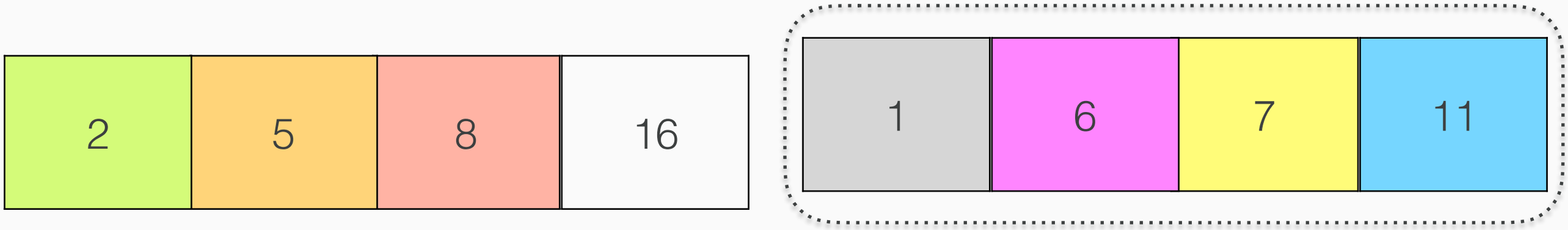
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



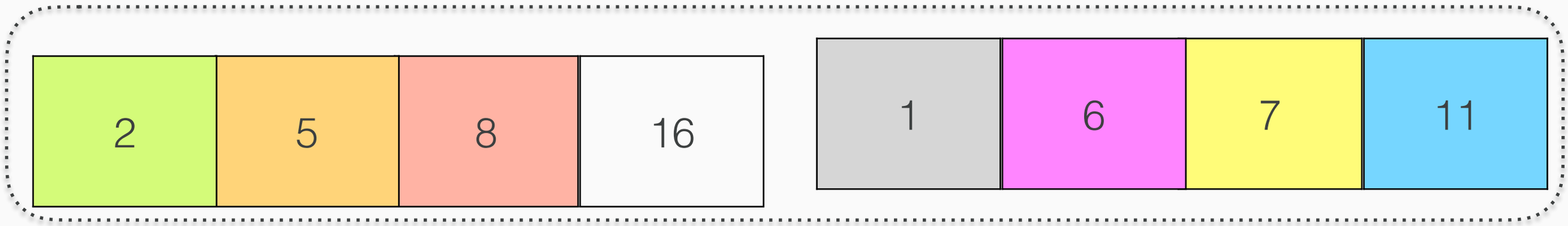
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



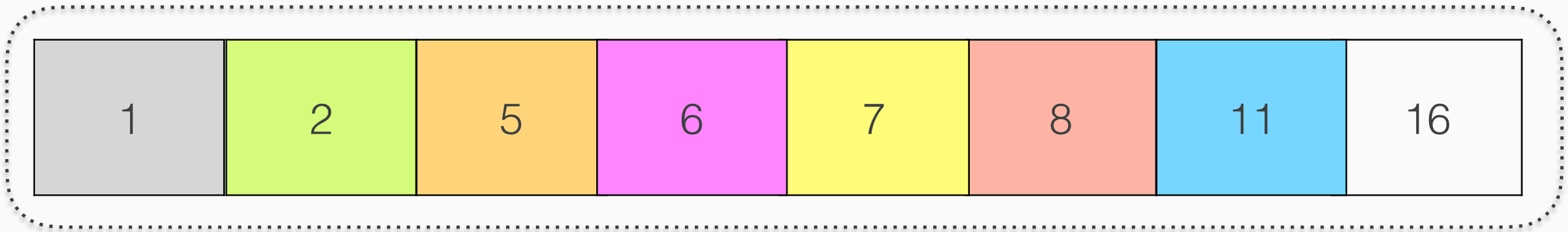
- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort



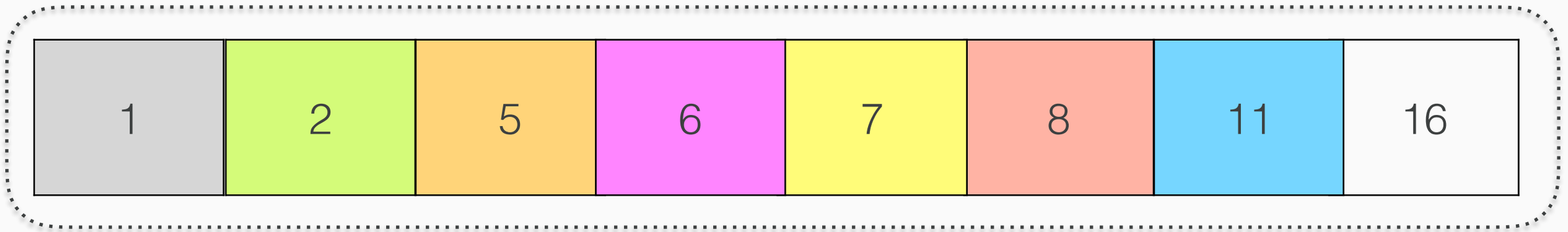
- Split the list in half, merge sort each half.
- **Merge the two together.**

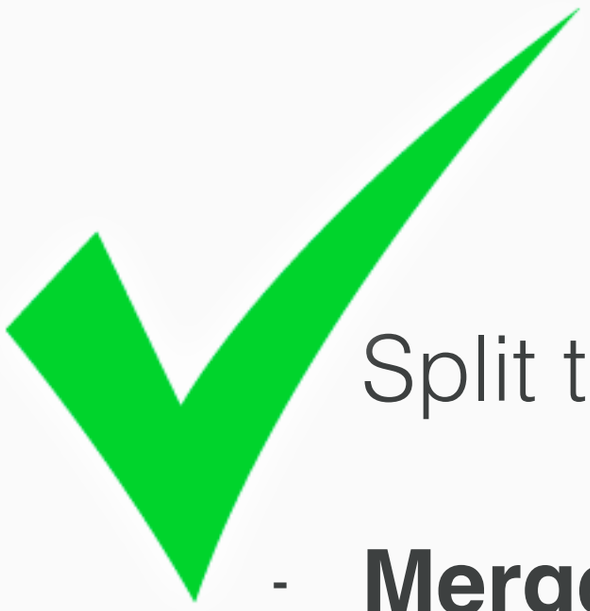
Merge Sort



- Split the list in half, merge sort each half.
- **Merge the two together.**

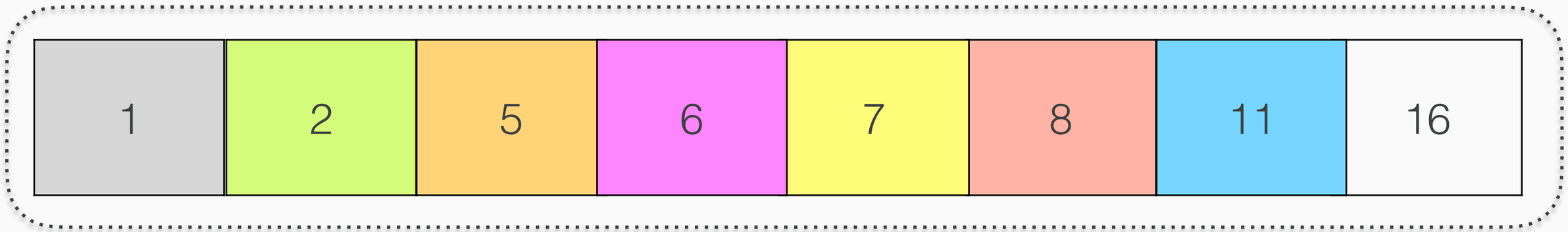
Merge Sort

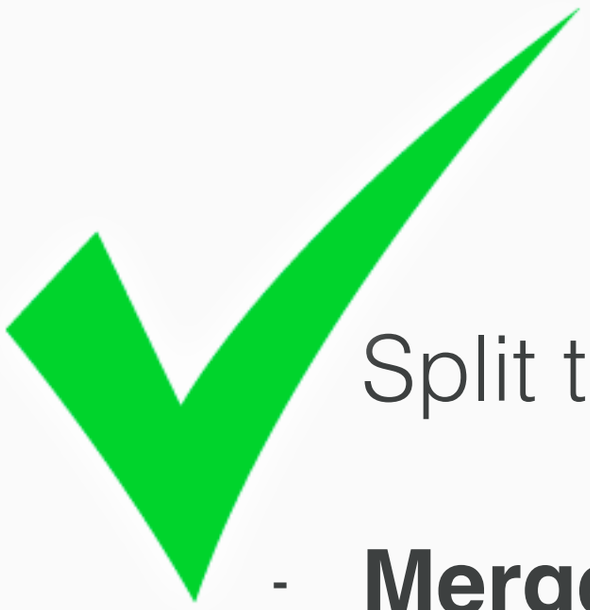




- Split the list in half, merge sort each half.
- **Merge the two together.**

Merge Sort





Neat Fact: growth rate of Merge Sort is N*log(N), fastest possible sort!

Split the list in half, merge sort each half.

- **Merge the two together.**

The Reading!

Recursive Algorithms

- Factorial
- Word Length
- Is a word a palindrome?
- Fibonacci
- Search, Sort

Recursive Algorithms

- Factorial
- Word Length
- Is a word a palindrome?
- Fibonacci
- Search, Sort
- Recursion order is important

Recursion: Recap

- **Definition**: a process, program, or object is said to be *recursive* if it involves repeated self-referenceSub bullet one
- In general, *recursive entities can be described as:*
  - **A simple step**
  - **A recursive step**
- Recursion can be *infinite*
- For recursion to be *finite,* we need a **base case.**
- Problems: Fibonacci, Factorial, Searching, Sorting, and more.
- Prefix Notation: 5 + 5 becomes + 5 5