More Sorting Algorithms

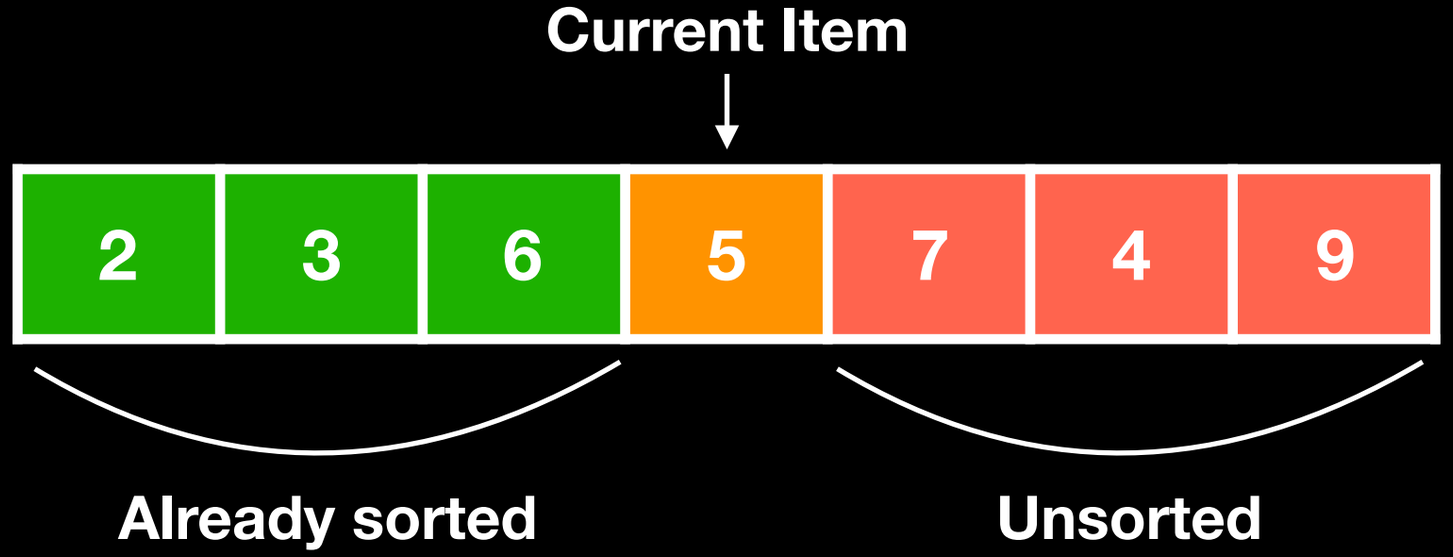
With the right algorithm, I can sort the world!

Simple Sorts

Insertion Sort

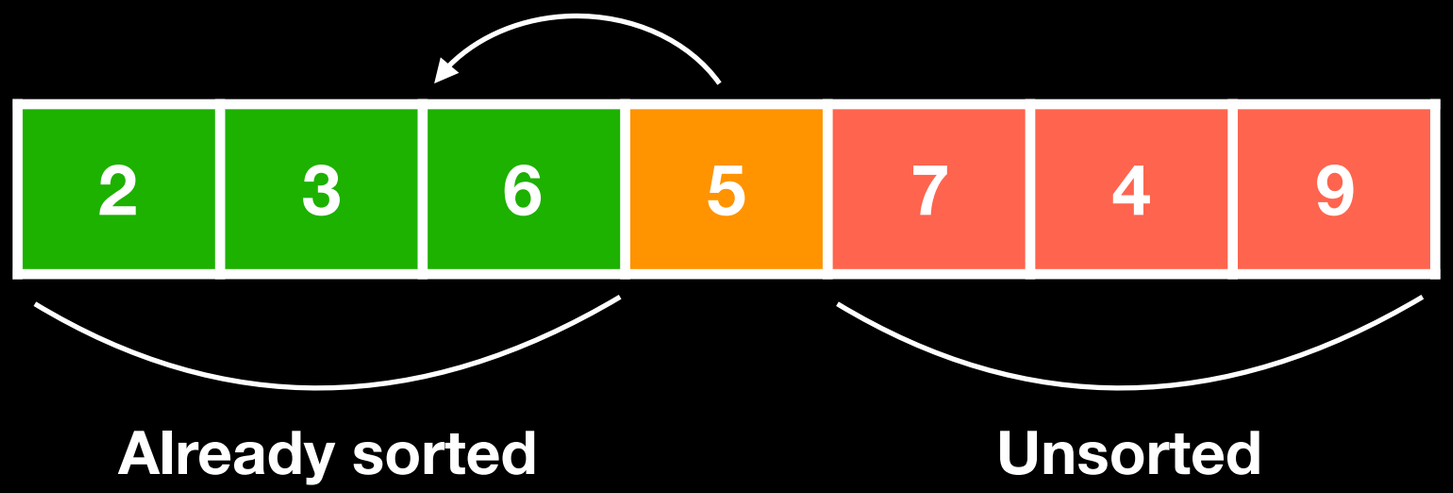
Simple Sorts

Insertion Sort



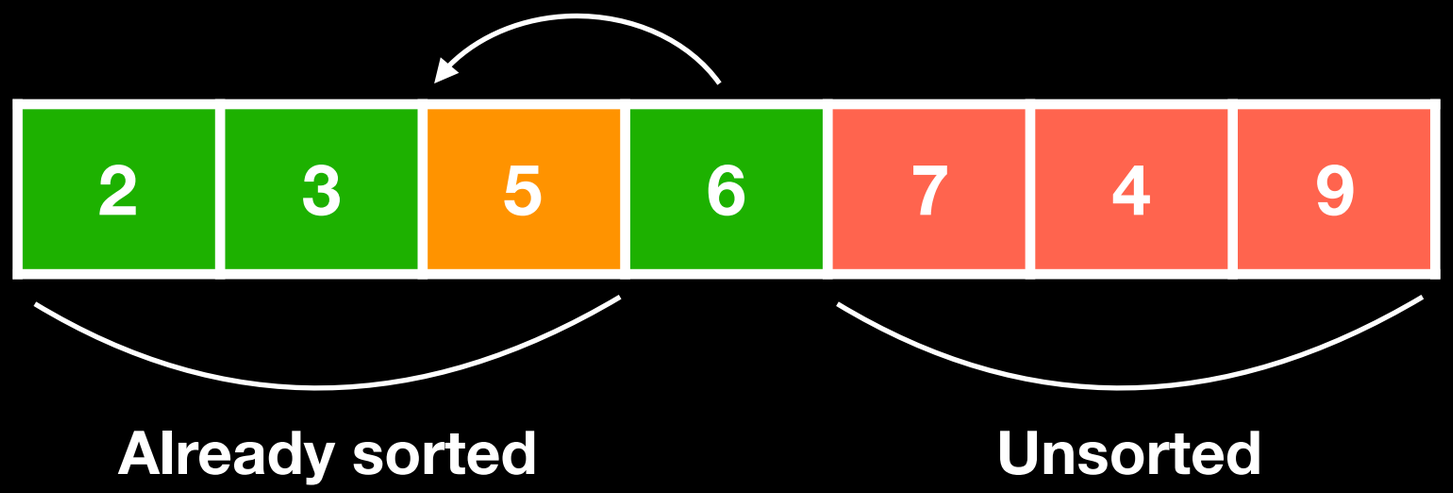
Simple Sorts

Insertion Sort



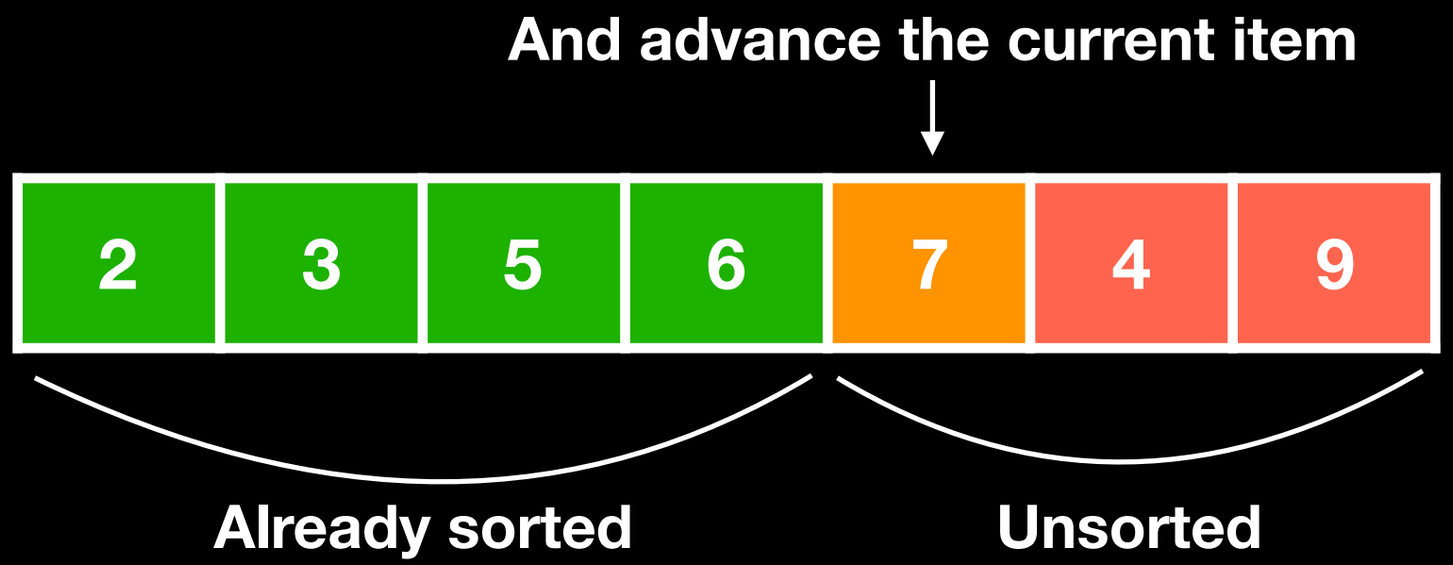
Simple Sorts

Insertion Sort



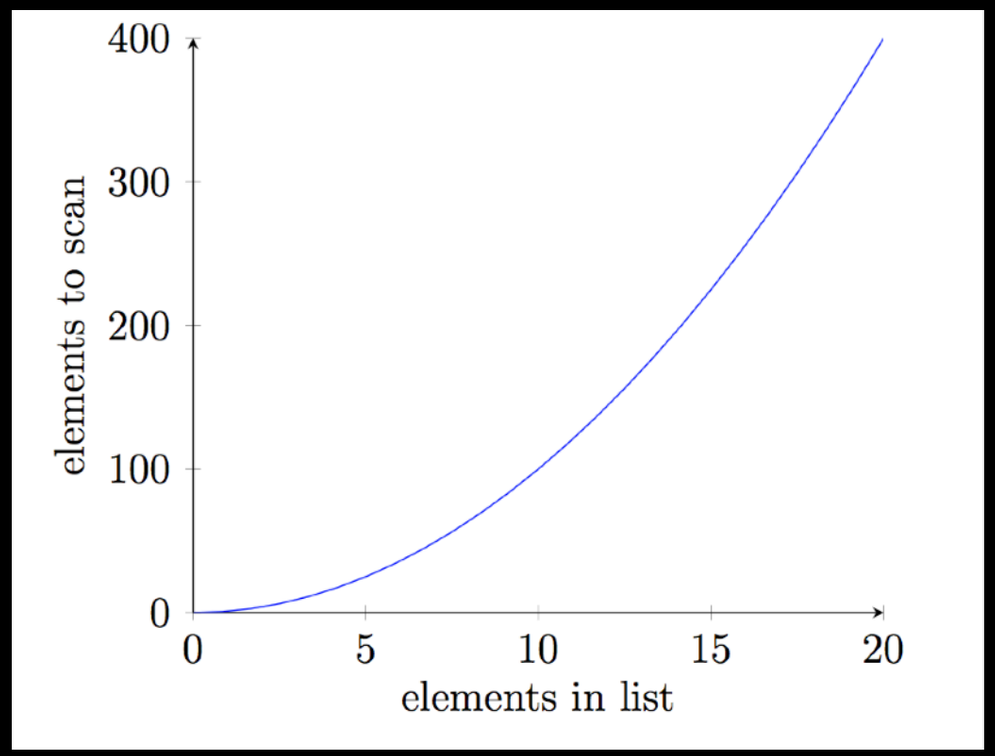
Simple Sorts

Insertion Sort



But this is slooooooow....

- In the worst case, we need to look at, or move around, n^2 numbers for a list containing only n numbers!

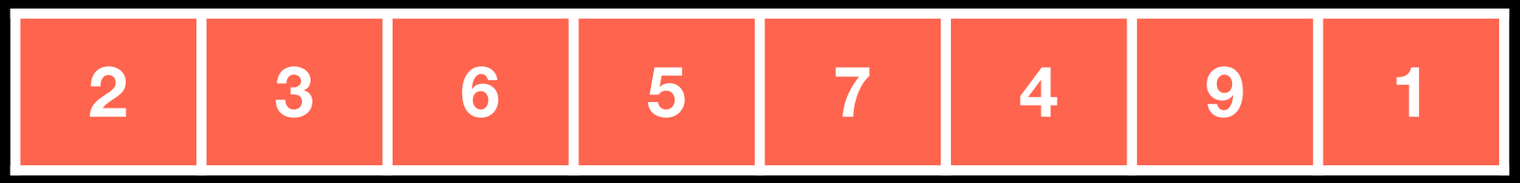


- There must be a faster way...

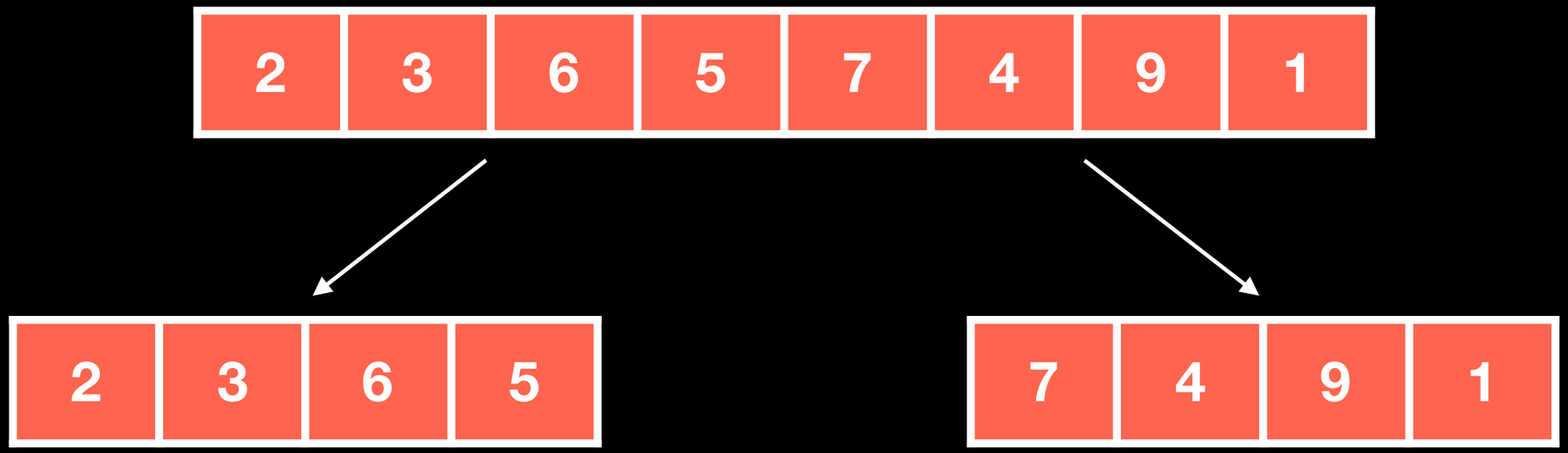
Enter... Merge Sort!

- Merge sort works on the principle of "*divide and conquer*"
    - Instead of doing all the work yourself, split it up into smaller pieces and handle them independently.
- Let's see it in action!

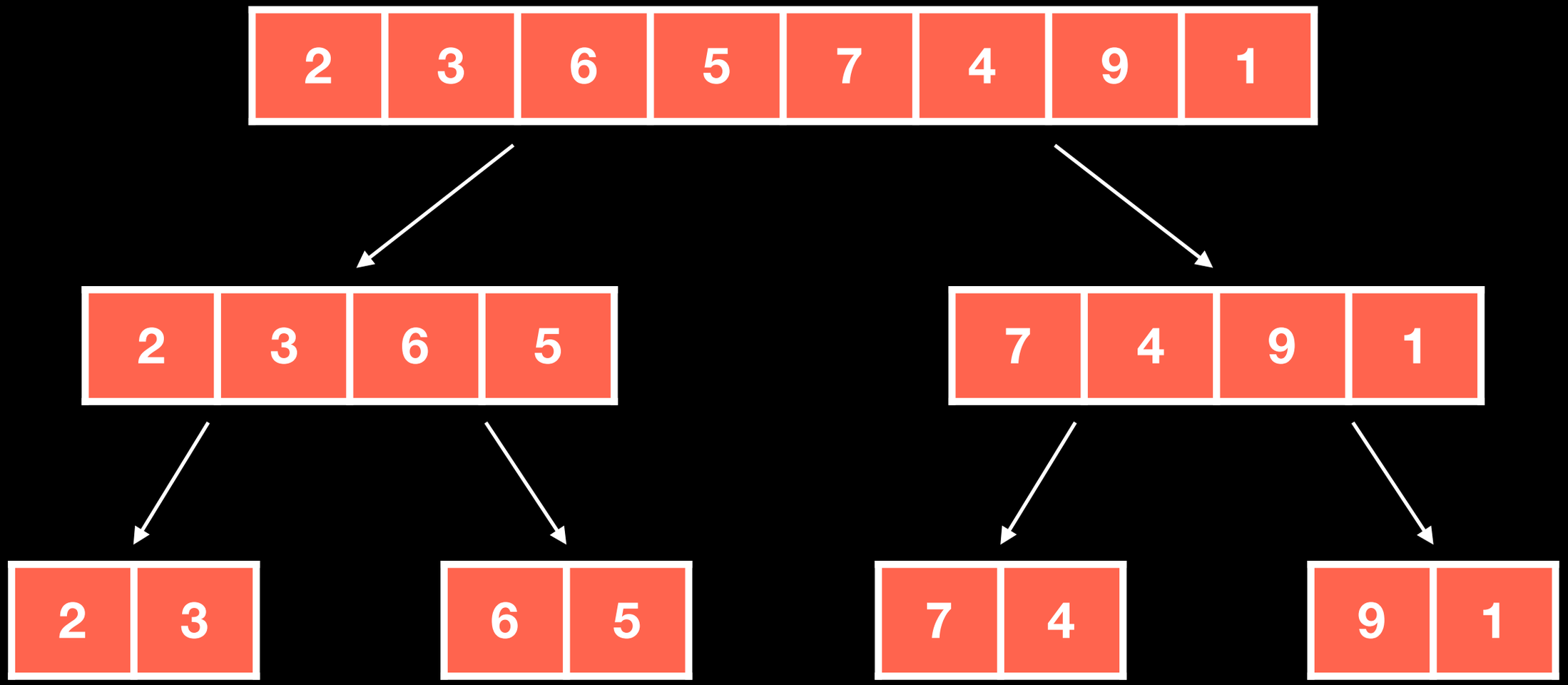
Merge Sort Example



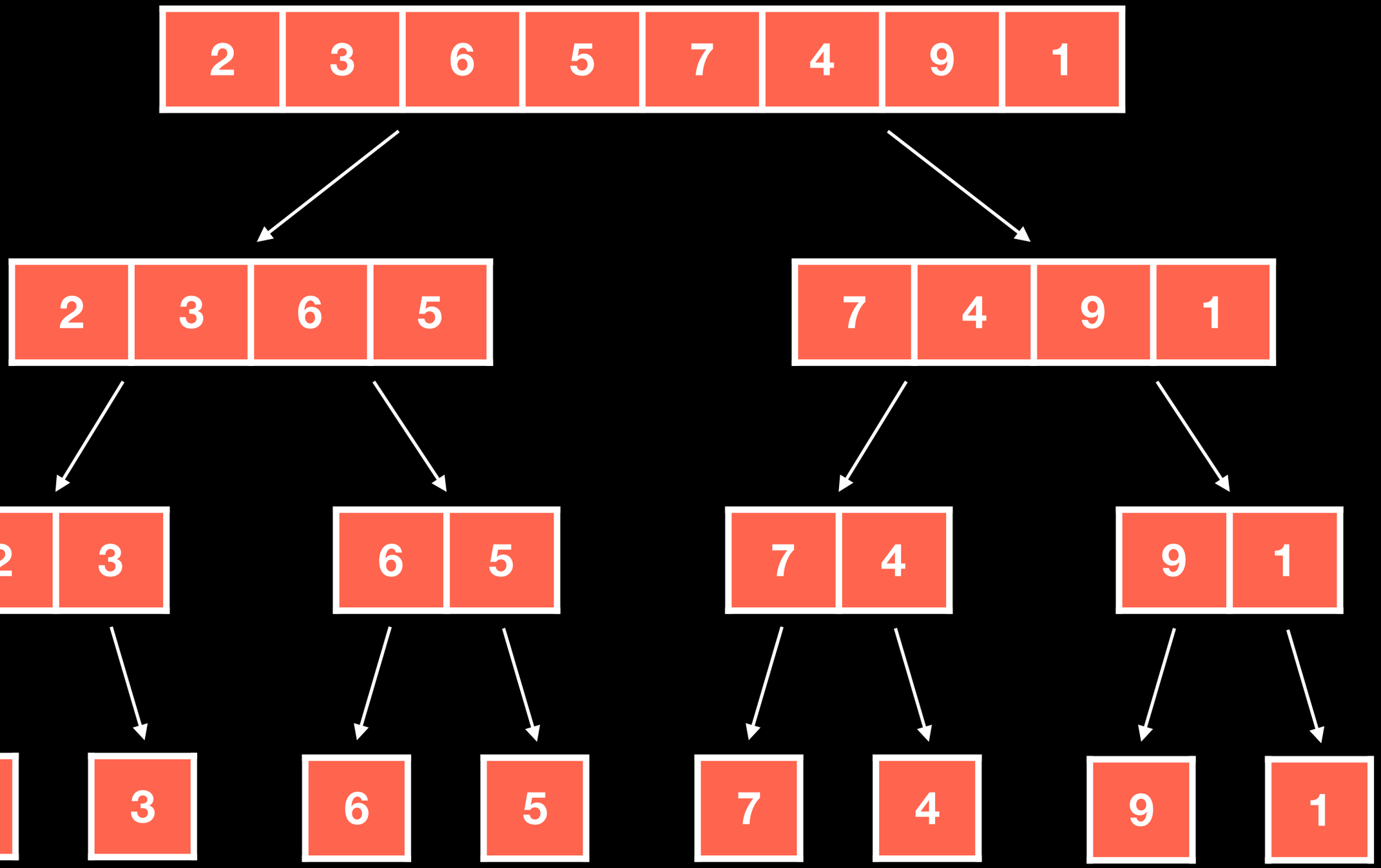
Merge Sort Example



Merge Sort Example



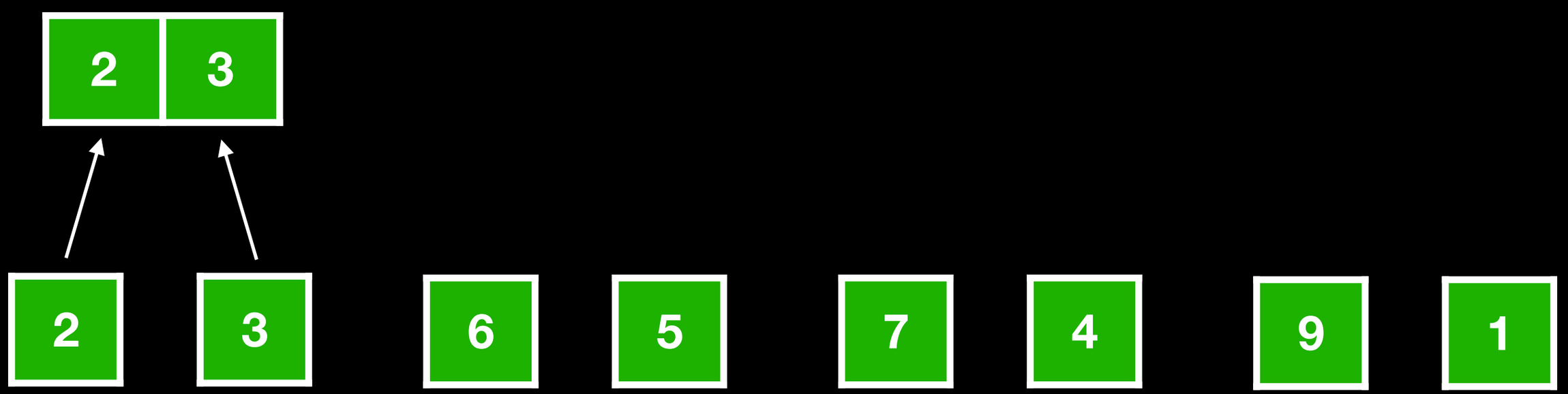
Merge Sort Example



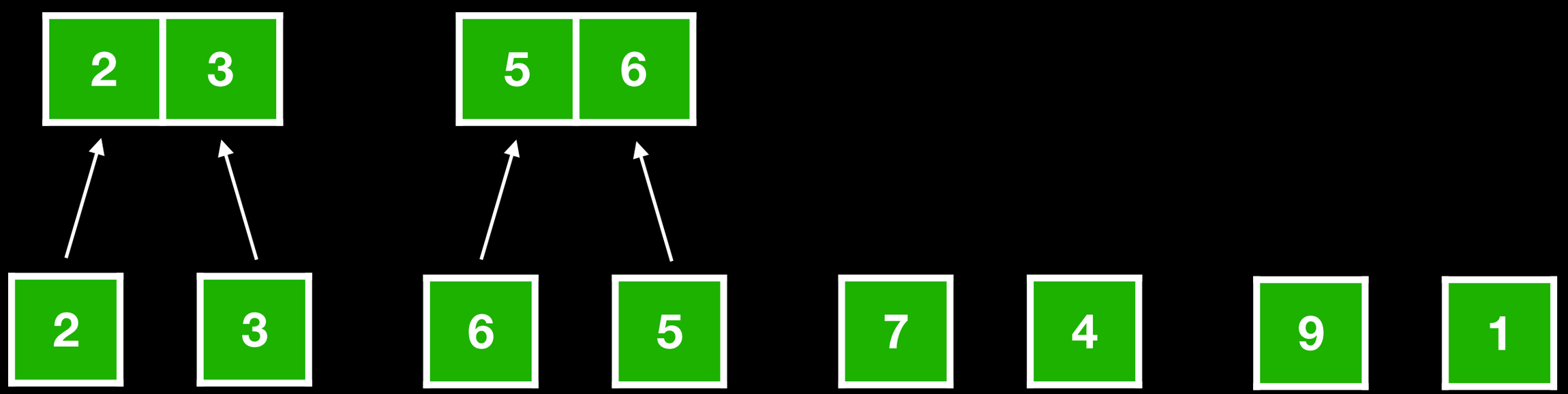
Merge Sort Example

Merge Sort Example

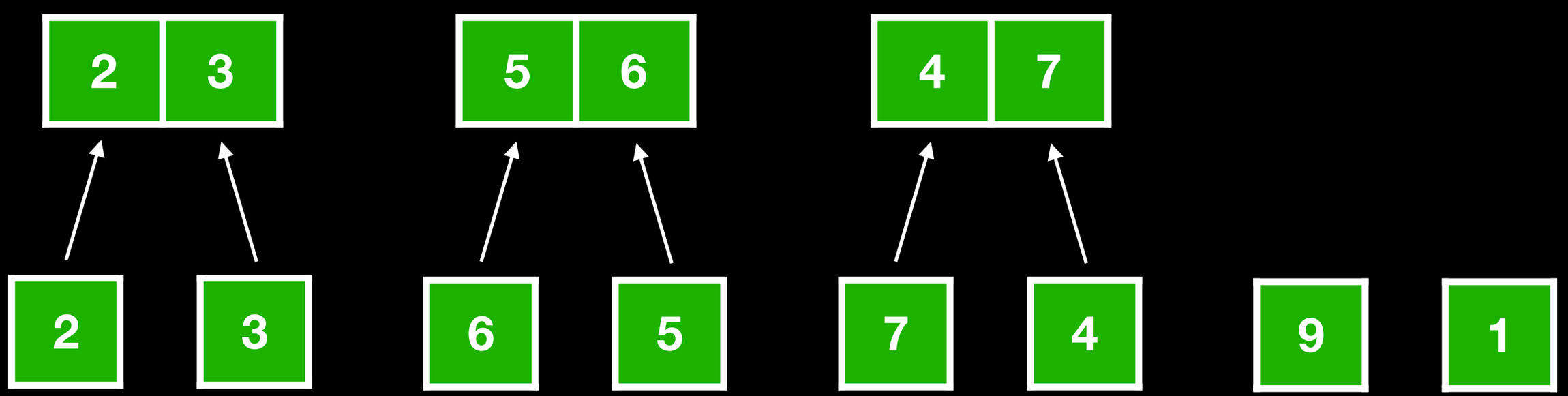
Merge Sort Example



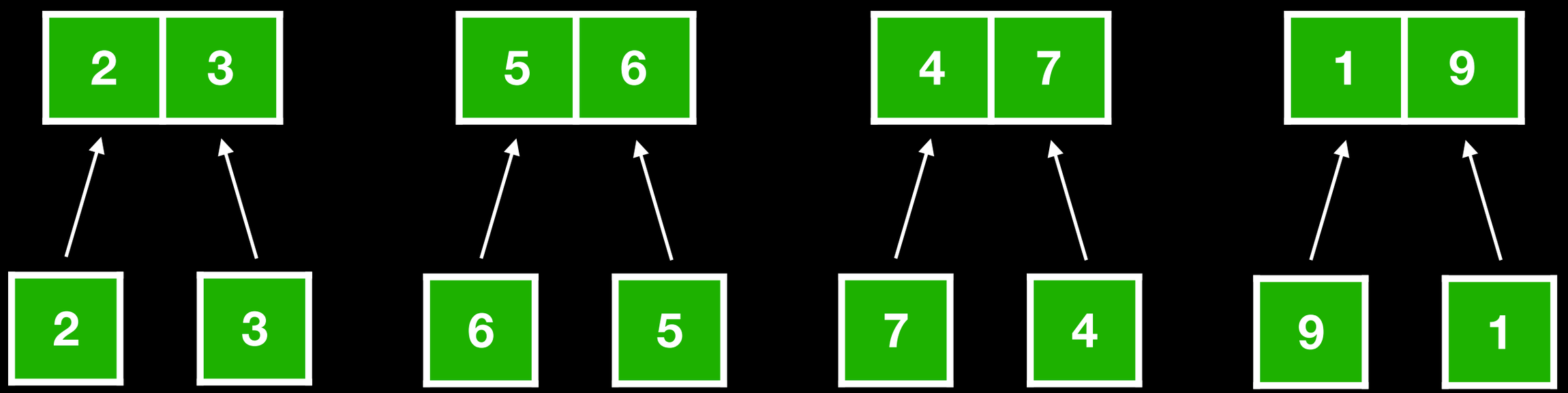
Merge Sort Example



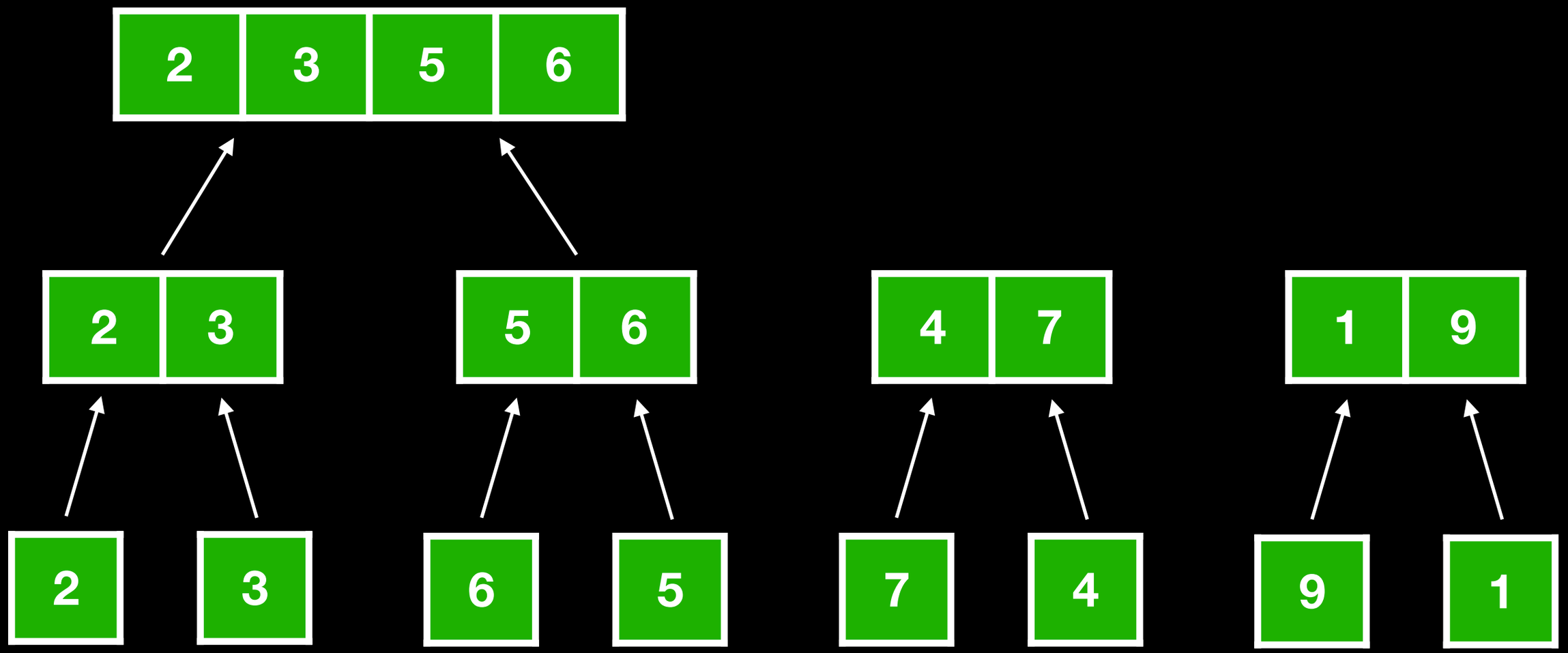
Merge Sort Example



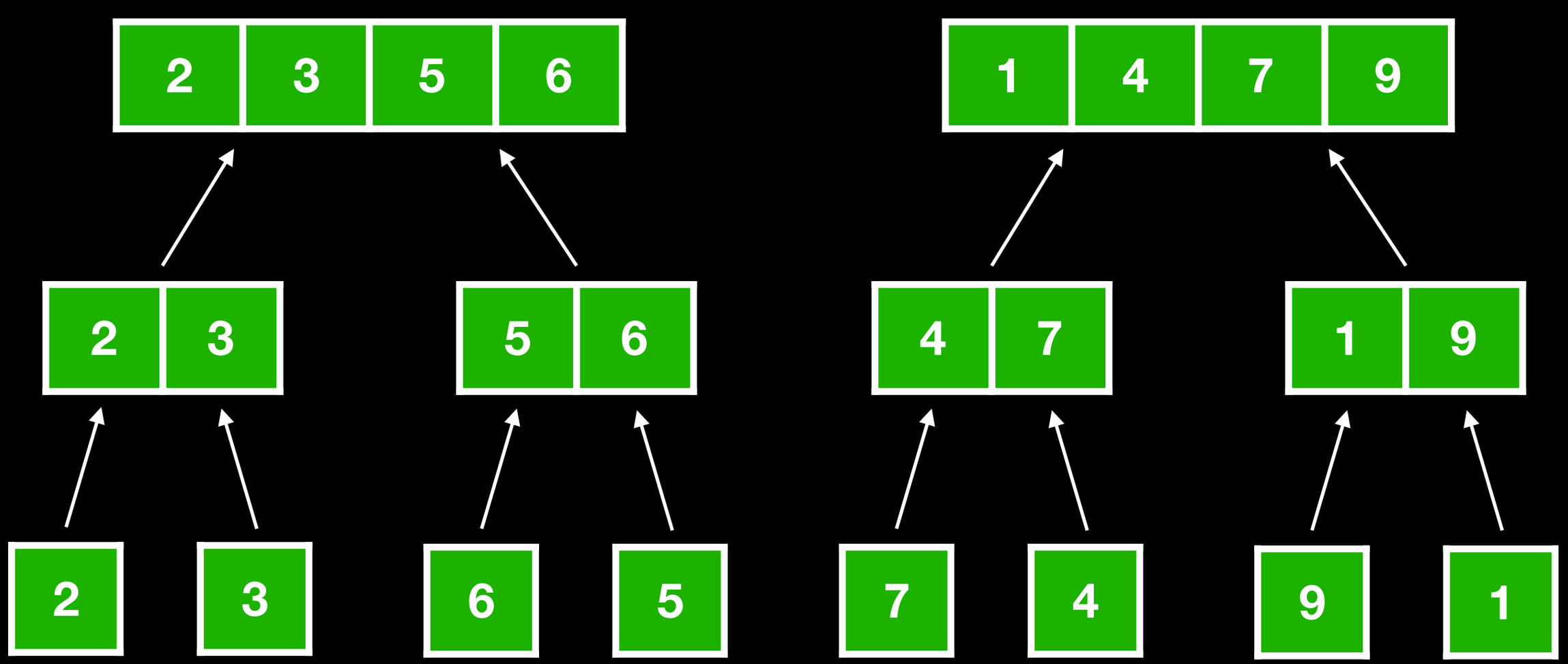
Merge Sort Example



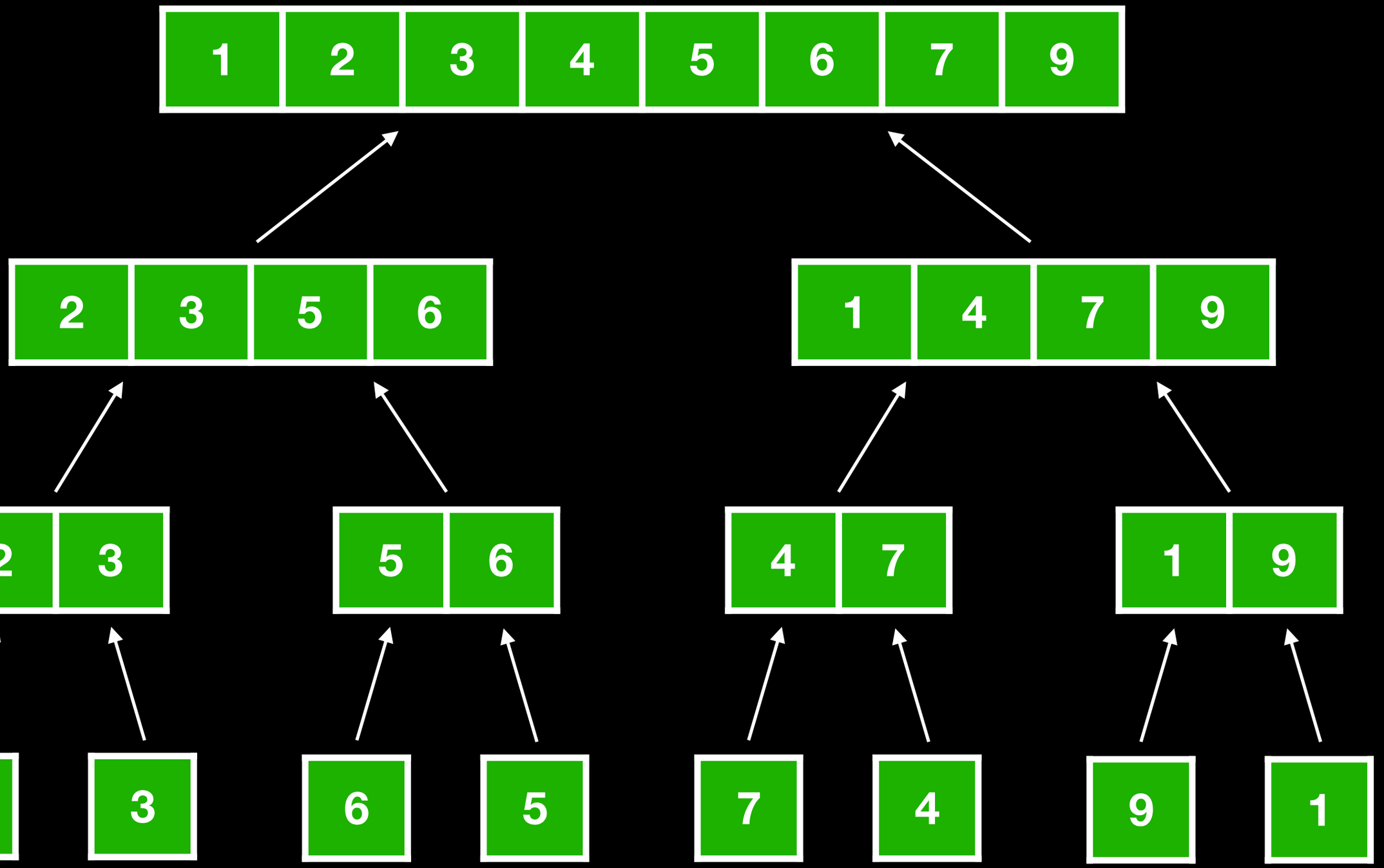
Merge Sort Example



Merge Sort Example

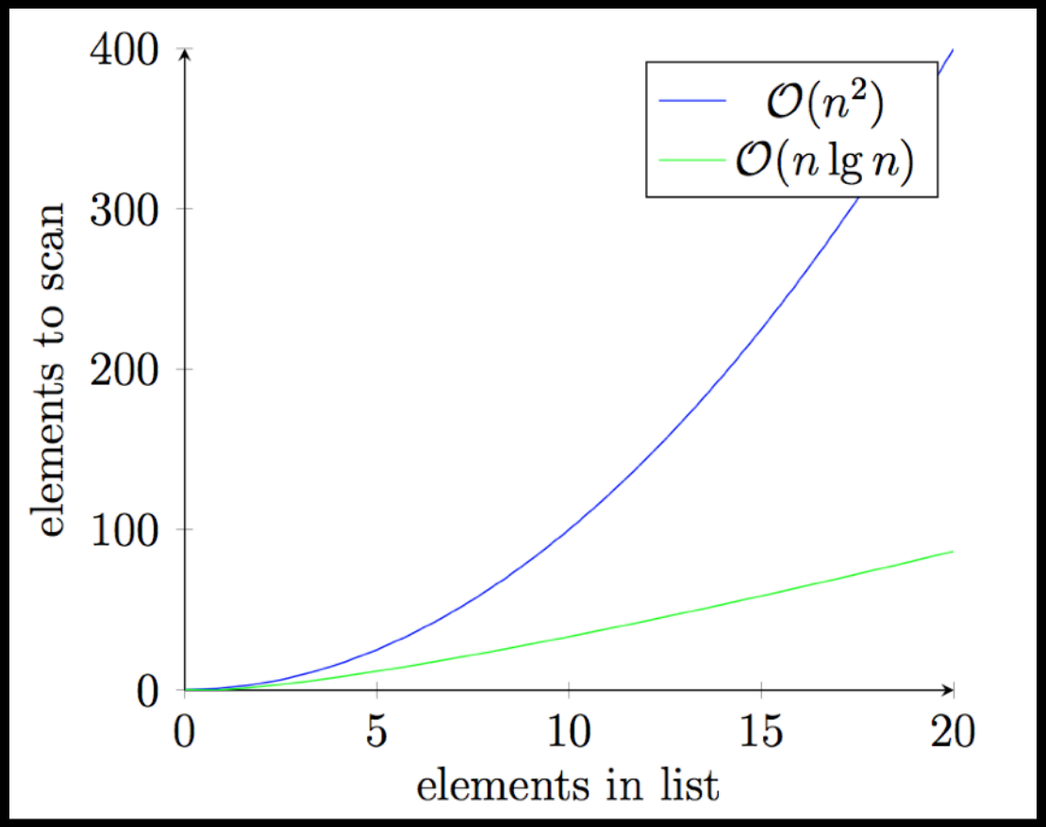


Merge Sort Example



This turns out to be a lot faster!

- We only need to look at $nlog(n)$ numbers in a list containing n numbers.



- As our lists grow, this is much less work.

Wow!

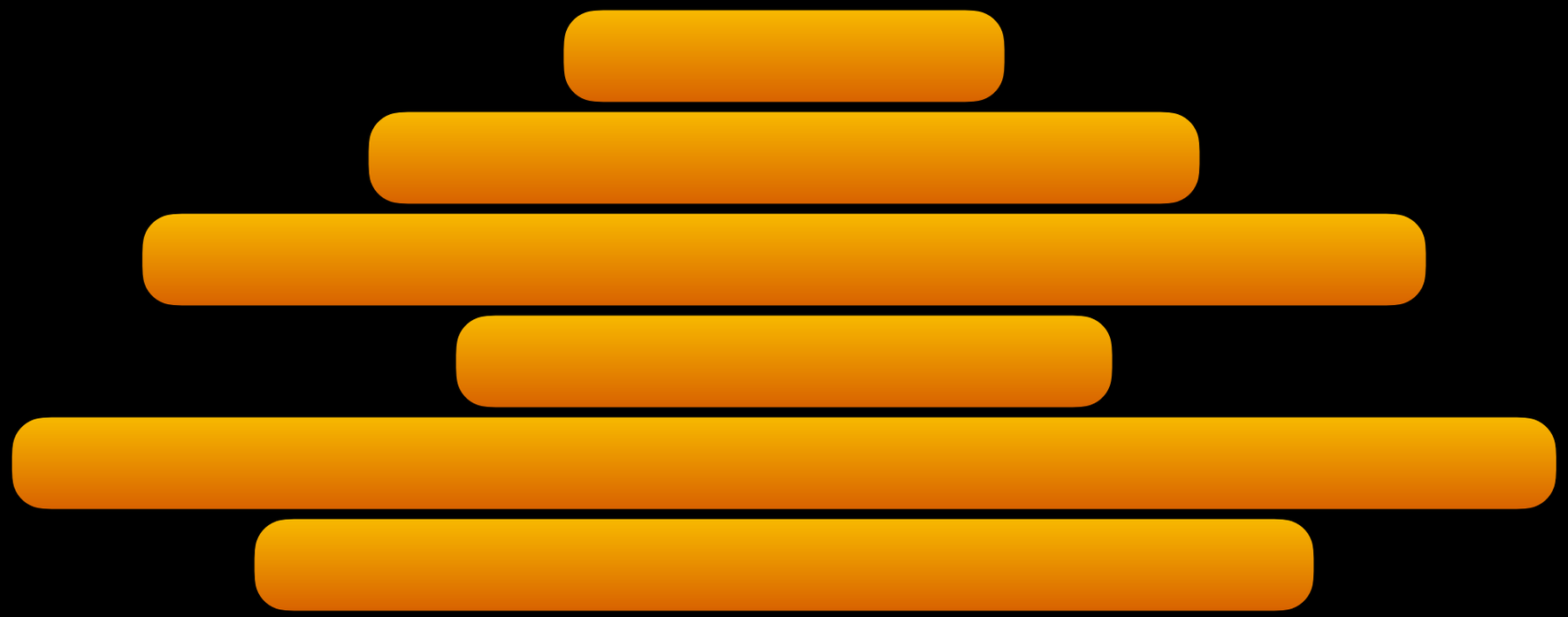
Let's get a little weirder...

Pancake Sort!

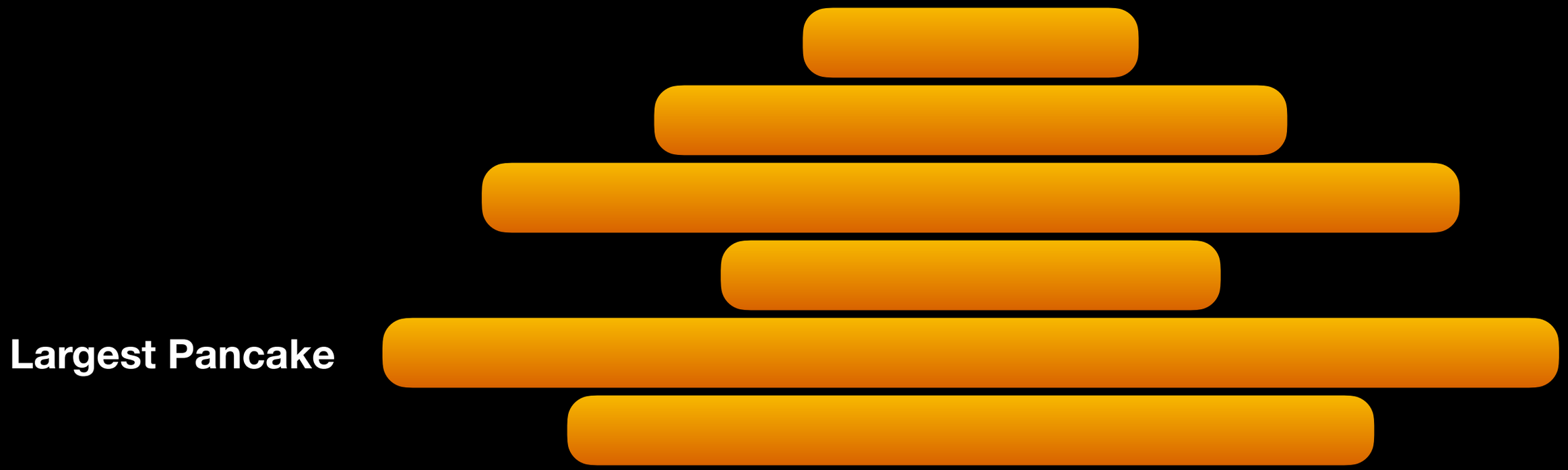
- Suppose I have a stack of pancakes, and I want to sort them by their diameter, with the smallest ones on top.
- However, the only "operations" that I can perform are:
  - I can stick my spatula anywhere inside the stack.
  - I can flip all the pancakes on top of my spatula.
- How can I sort them? Is it possible?

Of course it's possible...I wouldn't be telling you about it if it wasn't.

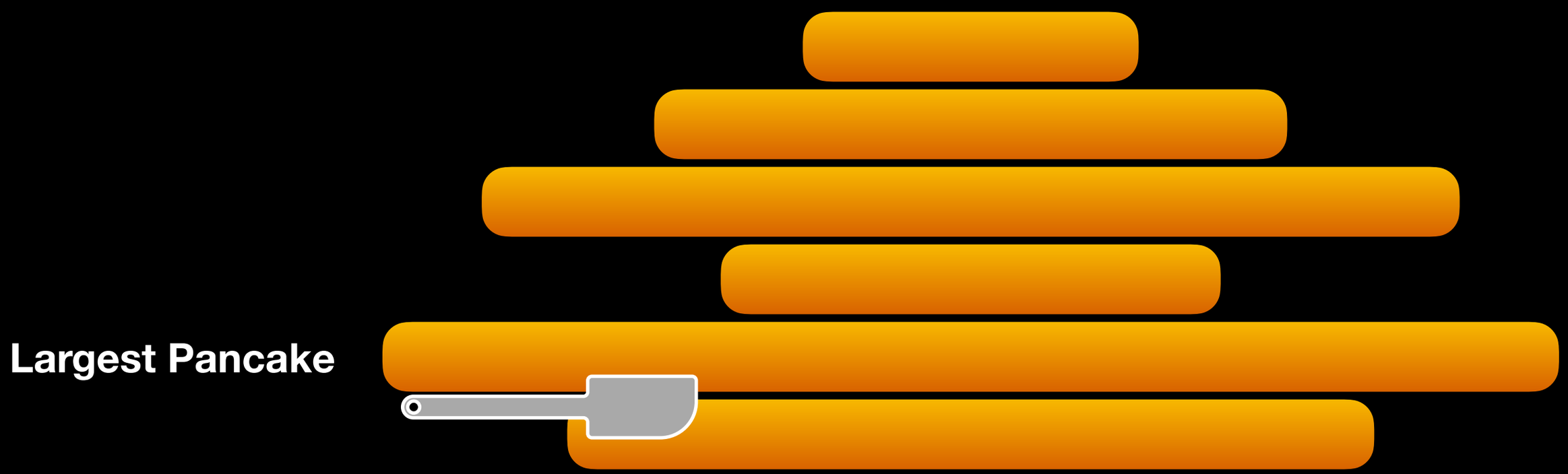
Pancake Sort Example



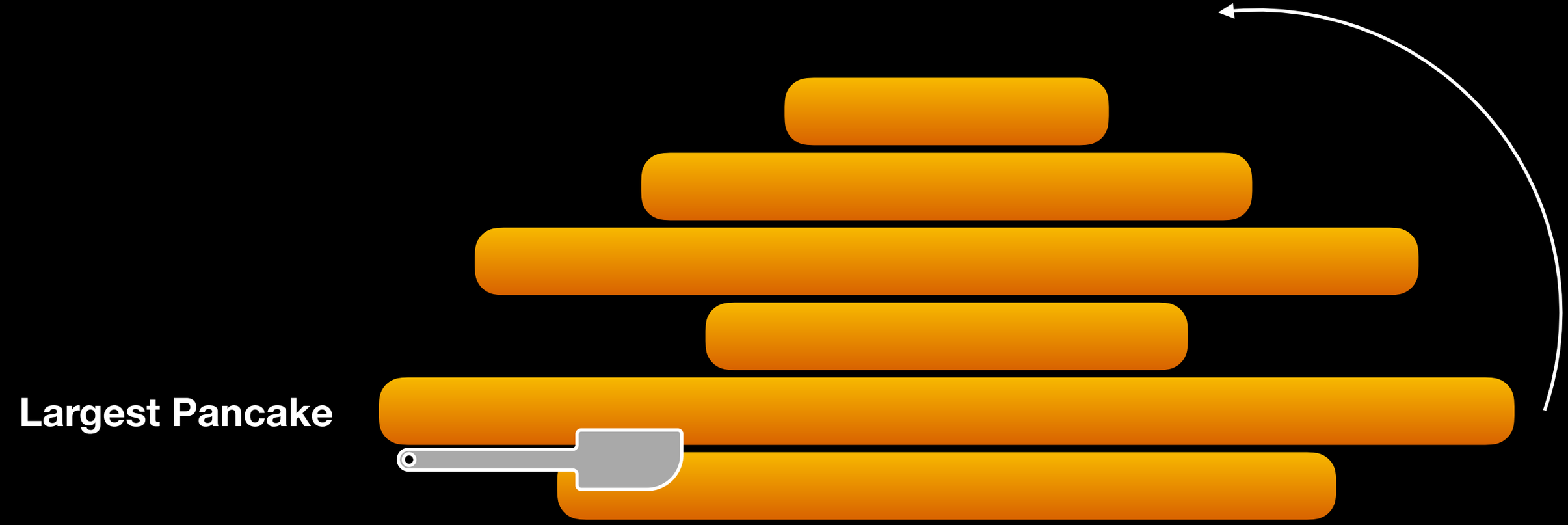
Pancake Sort Example



Pancake Sort Example

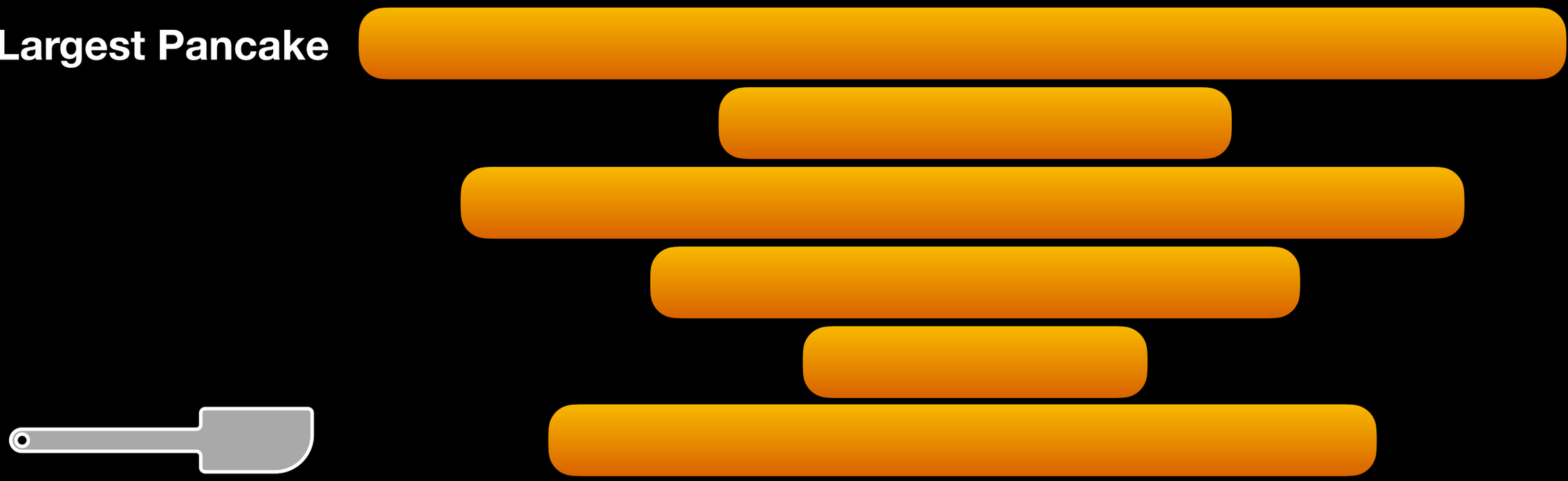


Pancake Sort Example

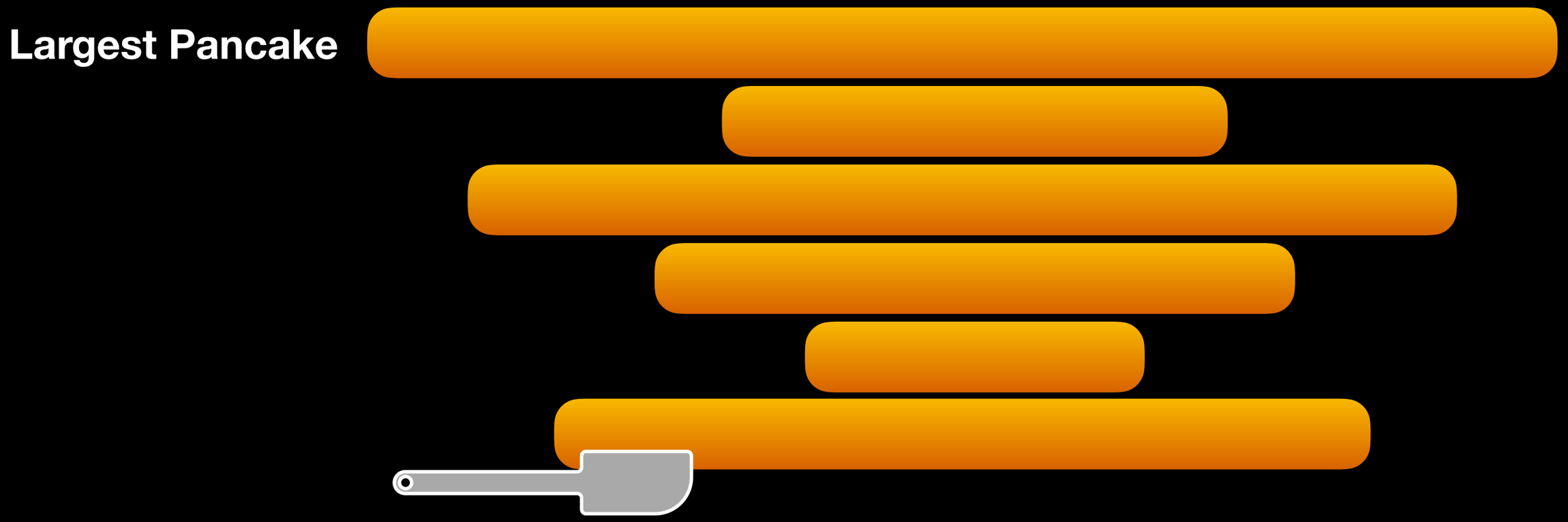


Pancake Sort Example

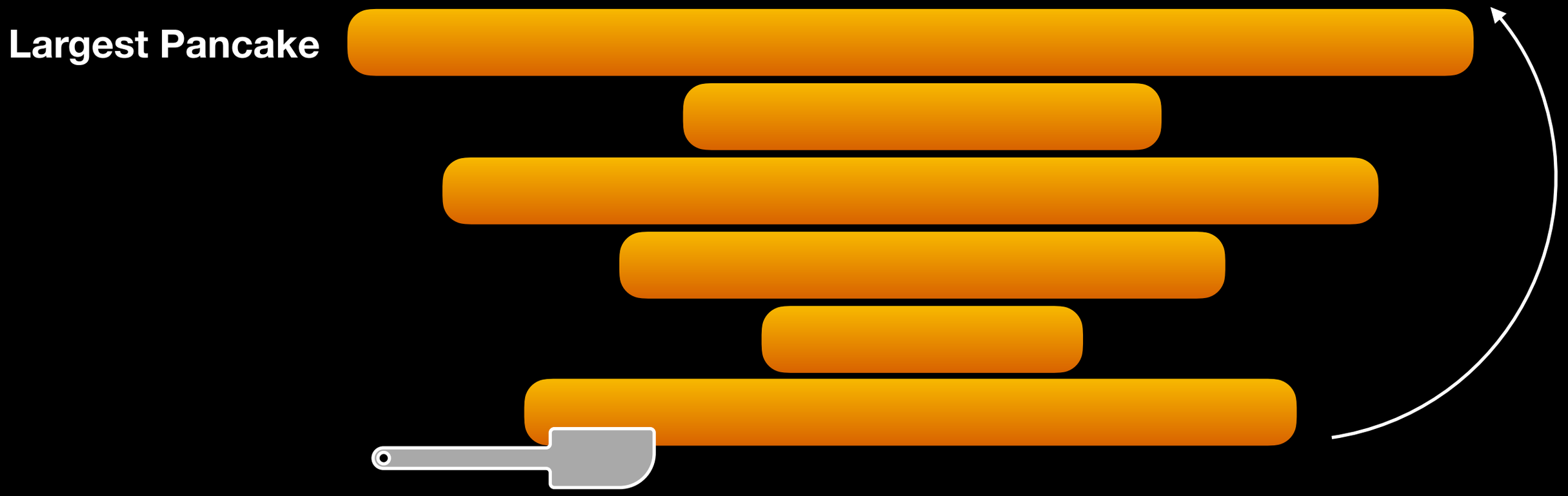
Largest Pancake



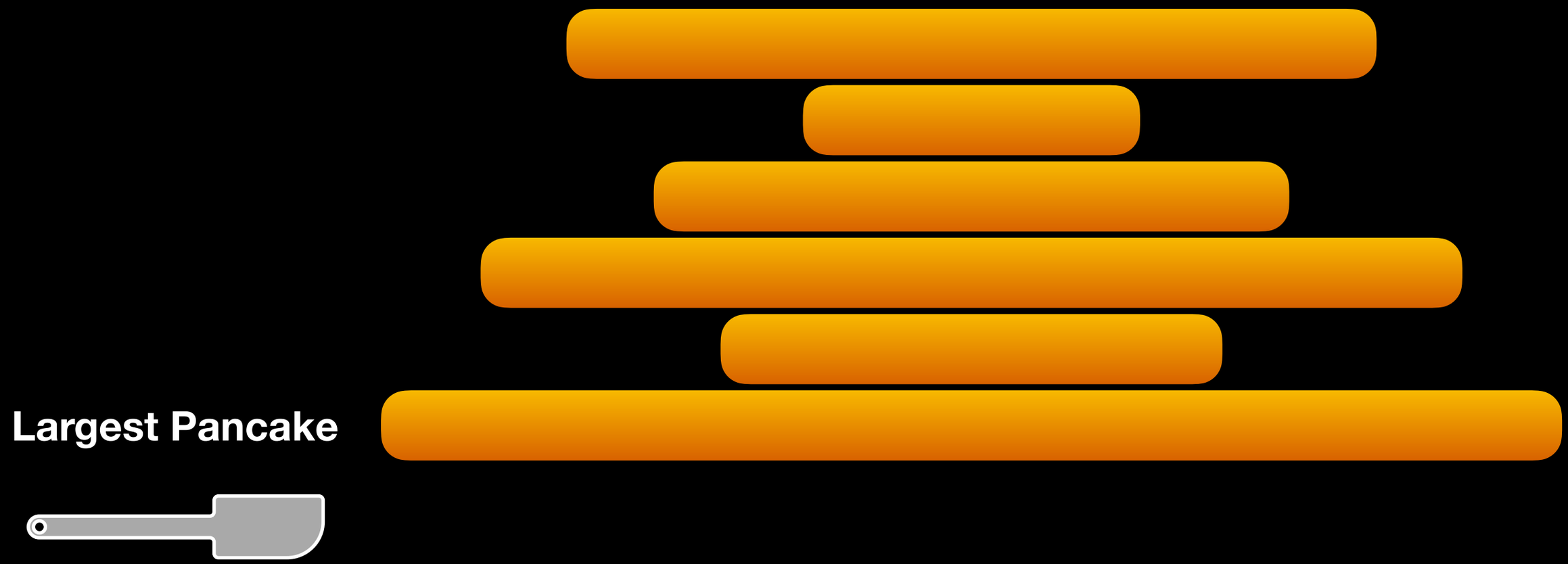
Pancake Sort Example



Pancake Sort Example



Pancake Sort Example

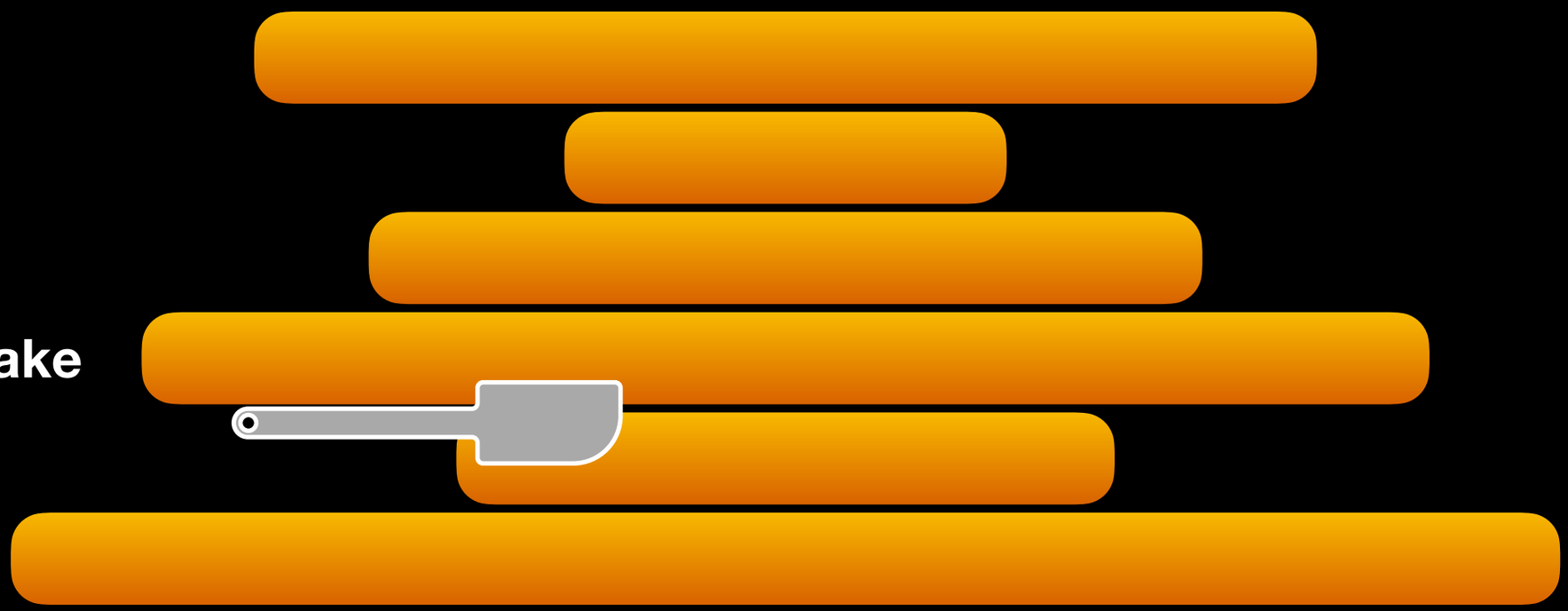


Pancake Sort Example

2nd Largest Pancake

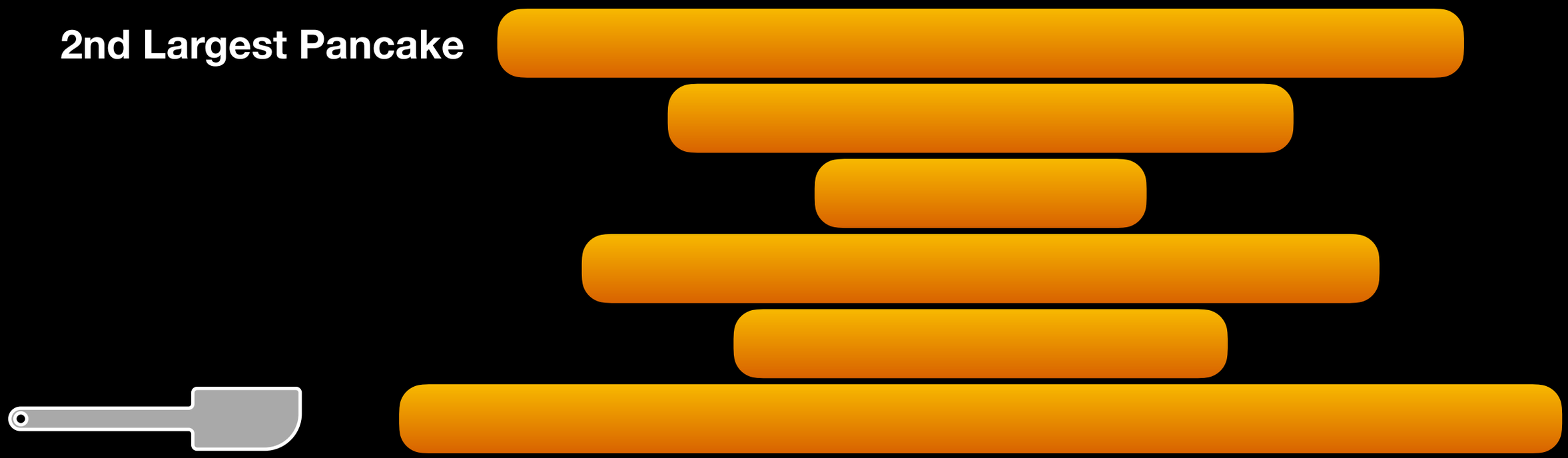
Pancake Sort Example

2nd Largest Pancake



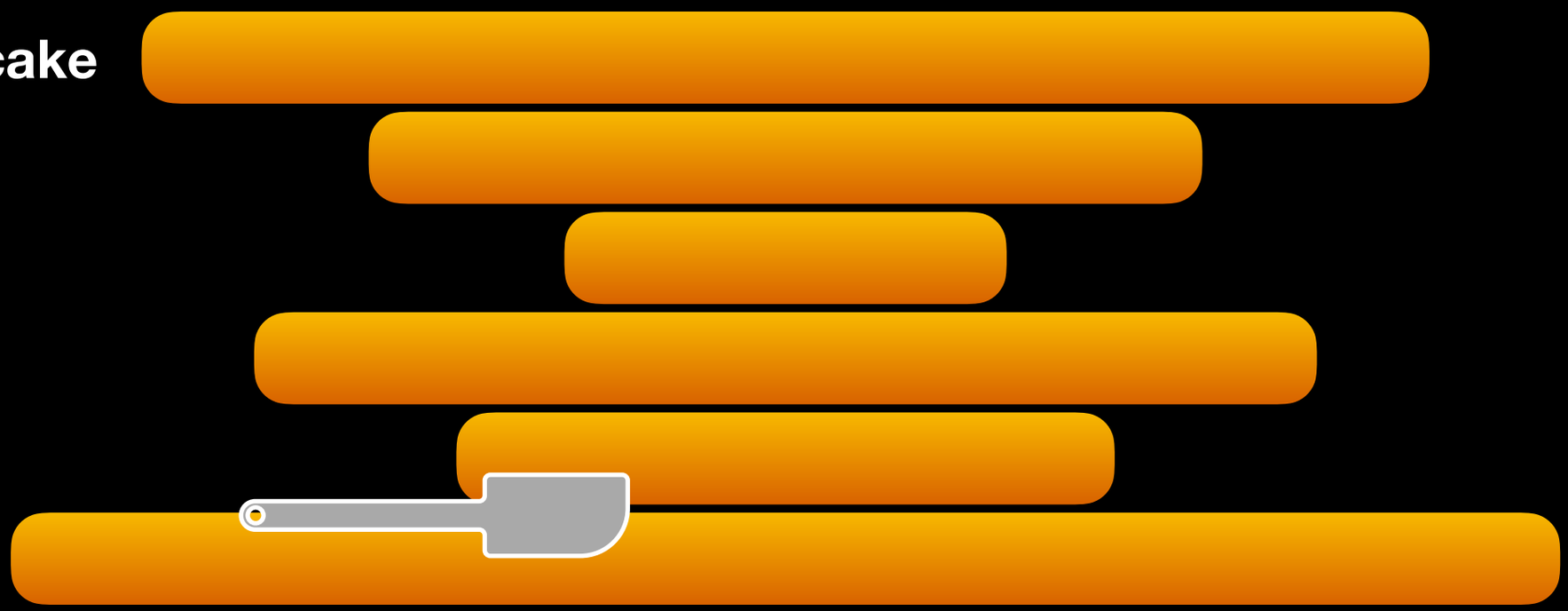
Pancake Sort Example

2nd Largest Pancake

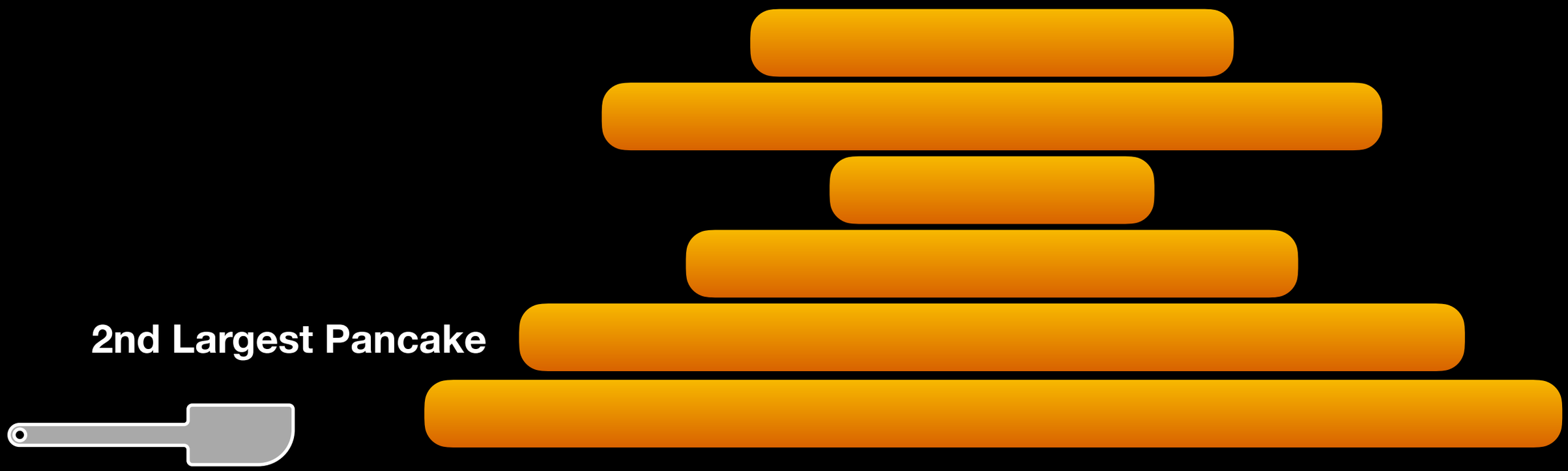


Pancake Sort Example

2nd Largest Pancake

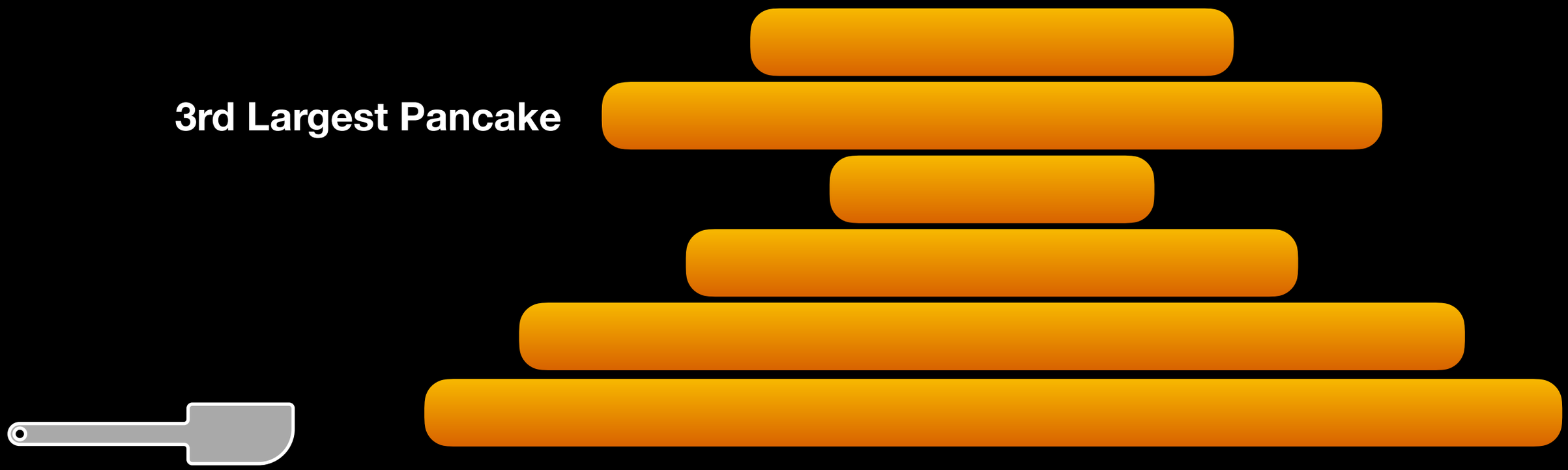


Pancake Sort Example

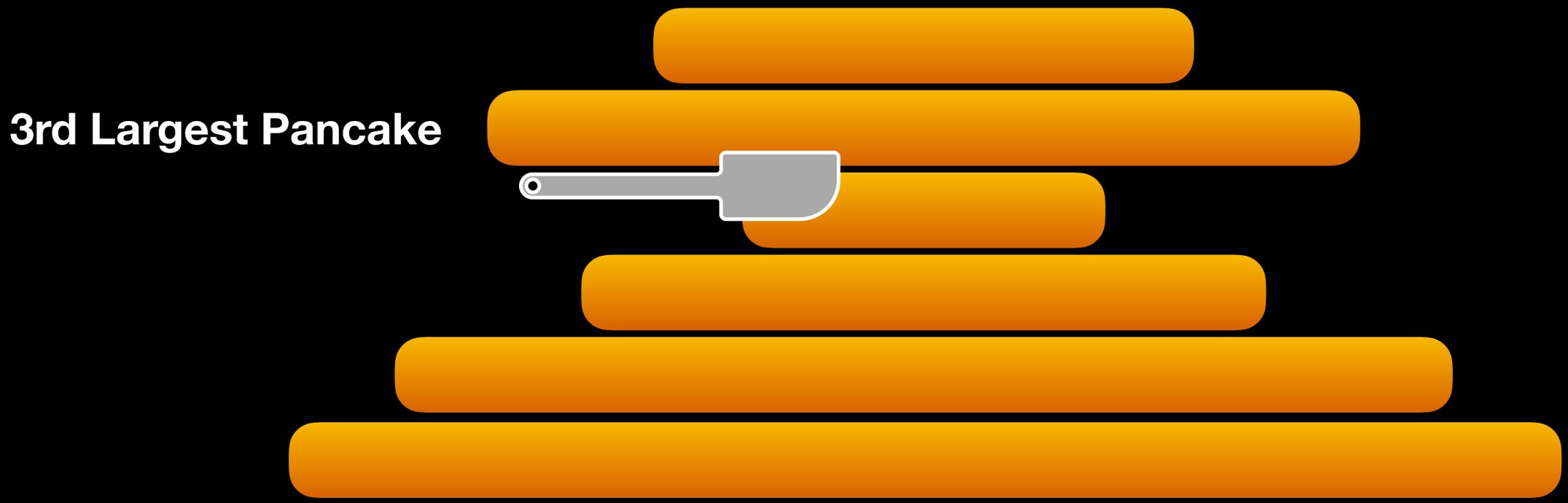


Pancake Sort Example

3rd Largest Pancake

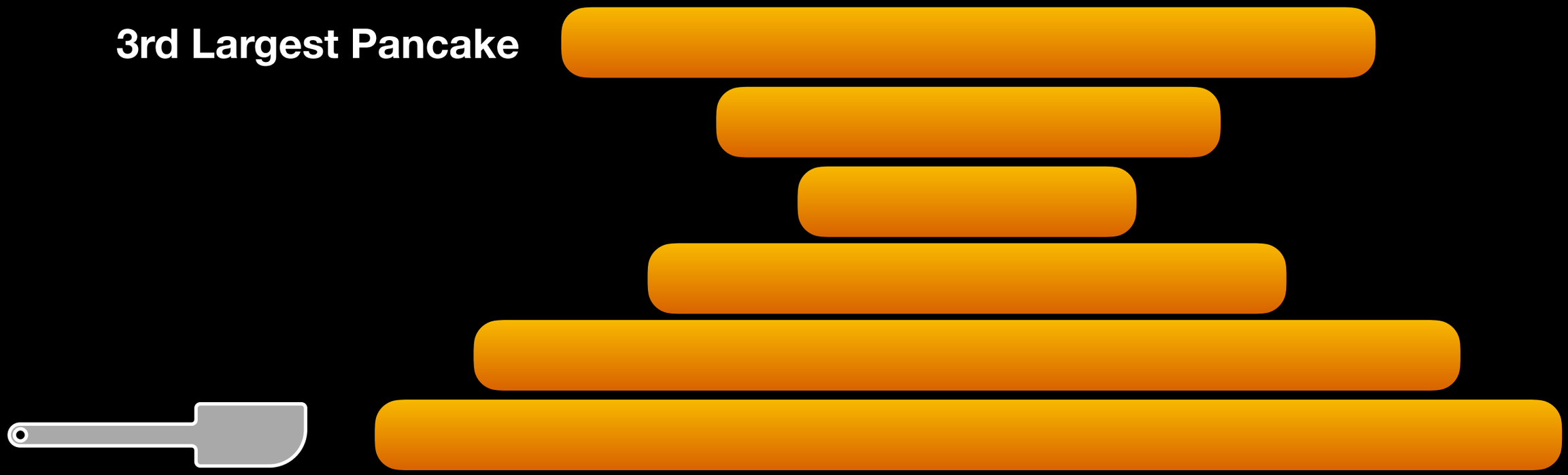


Pancake Sort Example

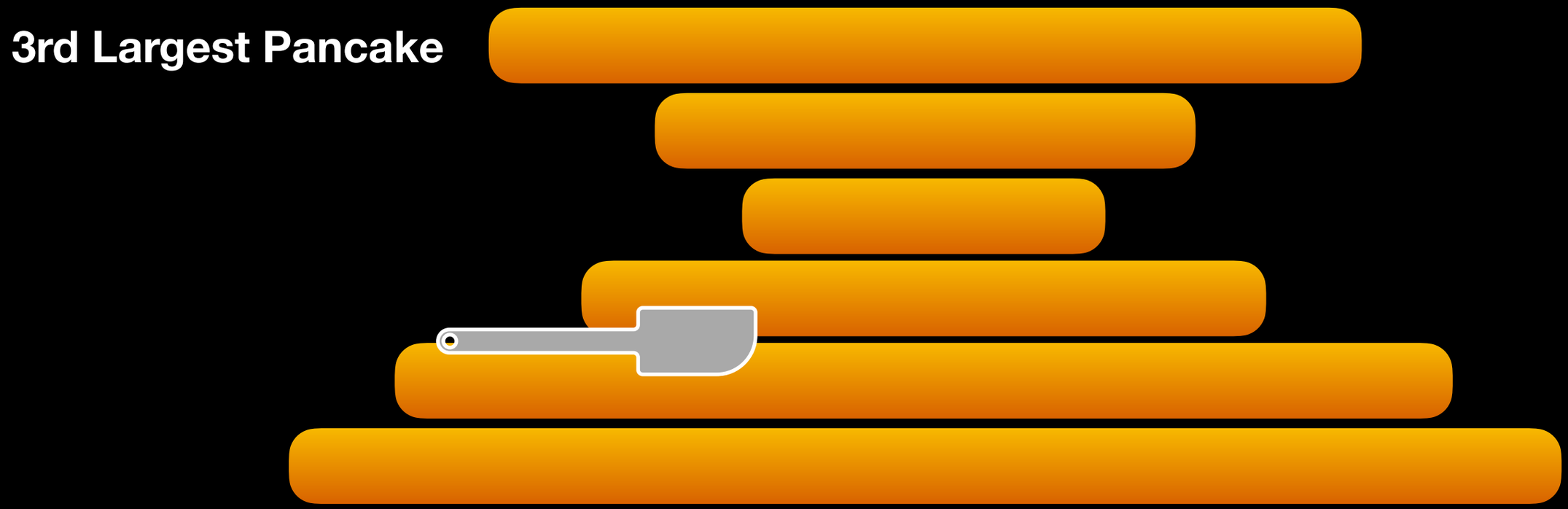


Pancake Sort Example

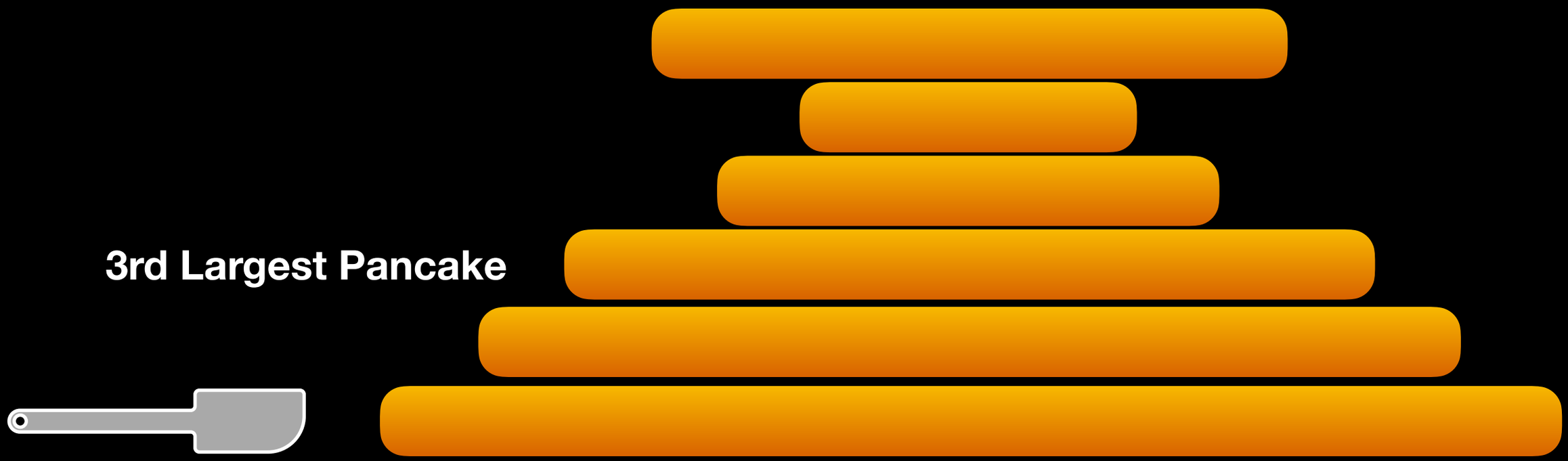
3rd Largest Pancake



Pancake Sort Example

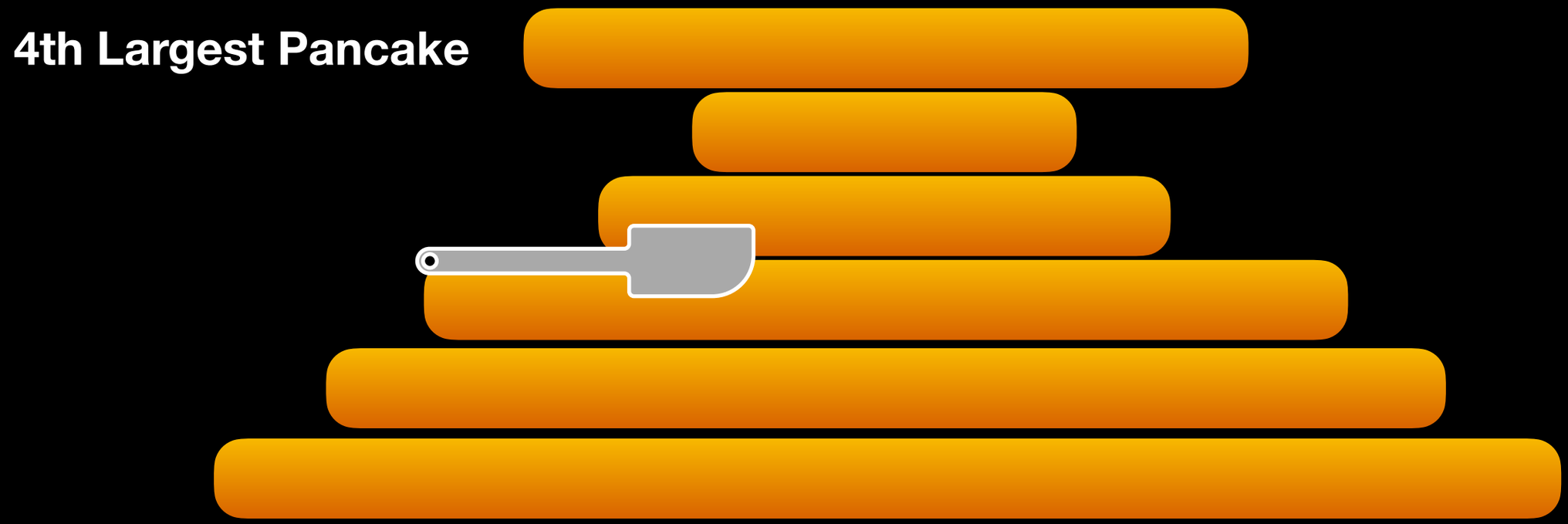


Pancake Sort Example



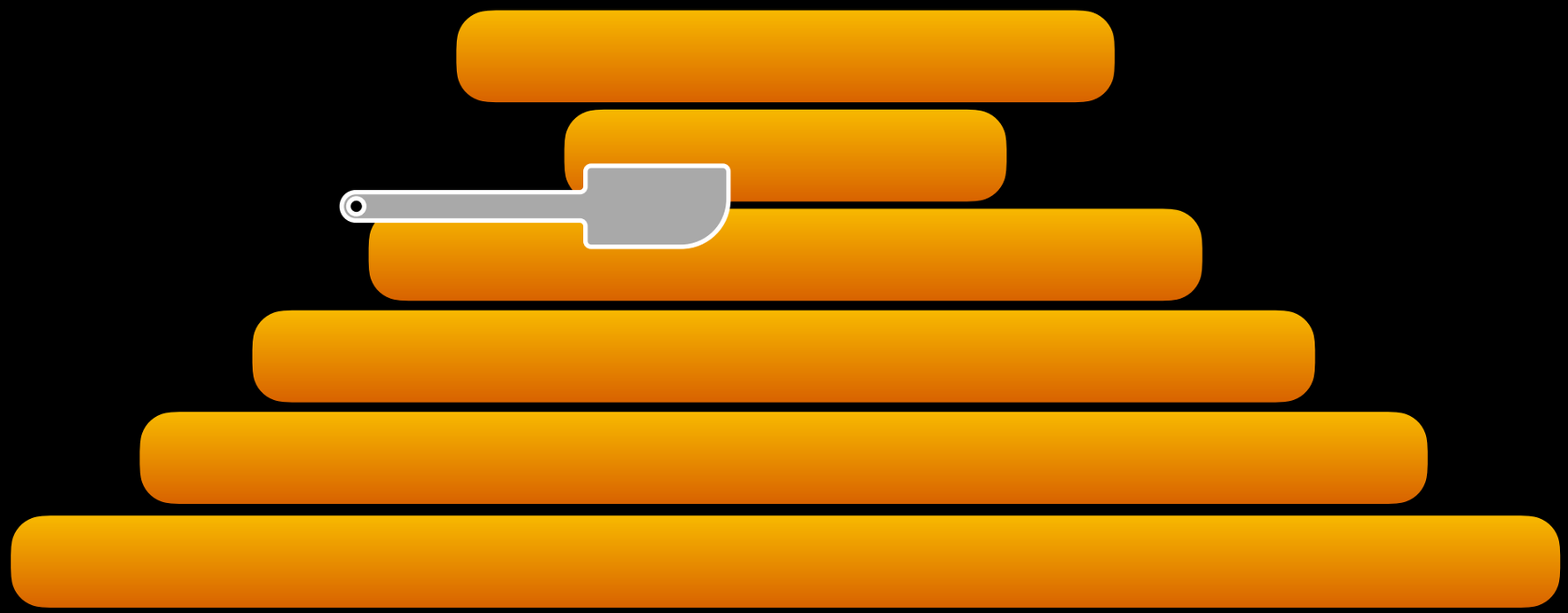
Pancake Sort Example

Pancake Sort Example



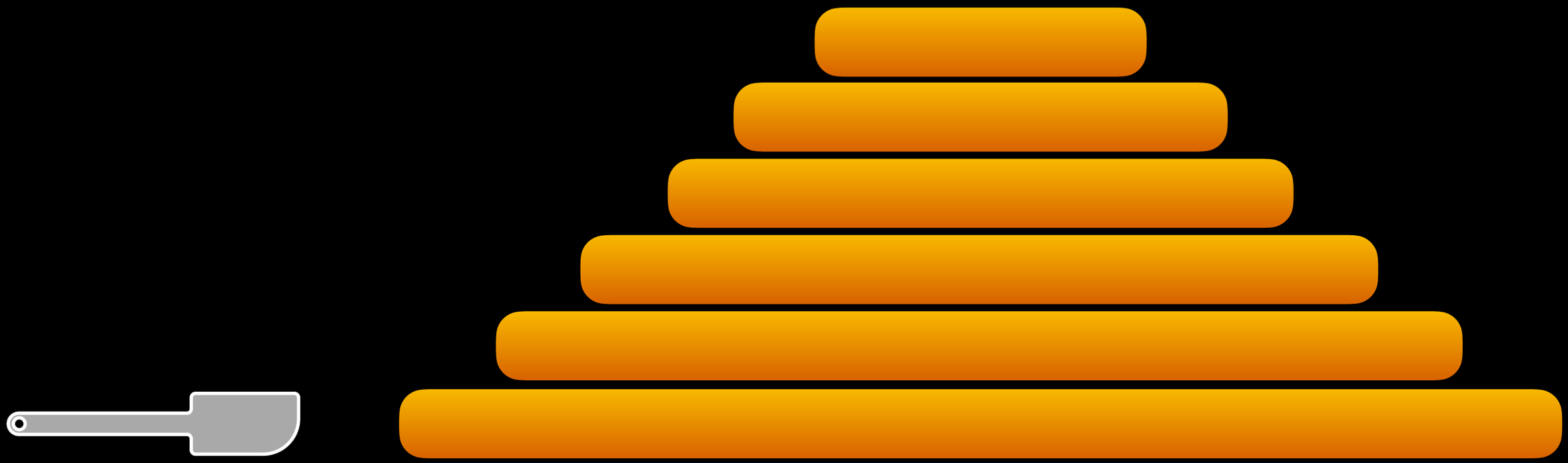
Pancake Sort Example

Pancake Sort Example



Pancake Sort Example

And we're done!



More On Pancake Sort

- Though not very useful-seeming at first, it actually turns out to have some practical applications.
  - Routing tasks between parallel processors.
- Bill Gates published a paper about pancake sorting, called "Bounds for Sorting by Prefix Reversal"
- There are some variations on the Pancake Problem:
  - The burnt pancake problem, in which pancakes must end up burnt-side down.

And finally, there's Bogosort...

- Given an unordered list of elements, do the following:
  - Randomly shuffle the list
  - If the list is now sorted, we're done.
  - Otherwise, repeat.
  - For a list containing n elements, Bogosort will on average take $(n+1)!$ steps before finishing.

Yes, that's a factorial! $(n+1)! = (n+1) * n * (n-1) * (n-2) * \ldots * 1$

  - However, Bogosort is actually unbounded (since it just shuffles the list -- it's possible that it **may never finish**)

With Bogosort, you never know how long something will take!

```
$ ./bogosort 2 1 59 3 4 11 50 100 -5 99 40
------ BOGO-SORTER 3000 -----

Was array initially sorted? ⟶ No

OK, let's sort it!

Attempt: 4736113
## Sorting successful at attempt 4736113!

Sorted array = -5 1 2 3 4 11 40 50 59 99 100

Overall time taken = 16 seconds

------ FINISHED -----
$ ./bogosort 2 1 59 3 4 11 50 100 -5 99 40
------ BOGO-SORTER 3000 -----

Was array initially sorted? ⟶ No

OK, let's sort it!

Attempt: 44347702
## Sorting successful at attempt 44347702!

Sorted array = -5 1 2 3 4 11 40 50 59 99 100

Overall time taken = 256 seconds

------ FINISHED -----
```

These two lists are exactly the same.

There are no practical applications of Bogosort (at least that I know of...)