Sorting and Runtime

Matthew Mussomele

A Quick Note

This lesson is going to be very demo intensive.

As a result, there will be less content in the slides themselves.

Sorting

Sorting presents the follow problem:

> Given a set of elements, put them in some natural or specified total order.

A total order is defined as:

- $x \leq y$ or $y \leq x$ for all x, y
- $x \leq x$
- $x \leq y$ and $y \leq x$ iff $x = y$
- $x \leq y$ and $y \leq z$ implies $x \leq z$

Sorting

Sorting can also be viewed as fixing something called inversions.

An inversion is defined as a pair of elements that are out of their natural order.

2 3 4 5 1

4 inversions: 2-1, 3-1, 4-1, 5-1

Sorting brings the number of inversions down to zero.

Sorting Algorithms

There are a number of sorting algorithms in existence.

We are going to talk directly about some of the more straightforward ones.

You will get a chance to see some more advanced sorts on todays task, but will not be responsible for implementing them.

Bubble Sort

Bubble sort is one of the simplest (and slowest) sorts out there.

The algorithm goes as follows:

1. Go through the list, comparing every pair of items as you go.
2. If they are out of order, then swap them.
3. Continue this until no swaps are required on a given sweep.

Runtime

The runtime of an algorithm is some approximation of how long it takes to run depending on some input N.

In the case of bubble sort, we say that it's *worst case* runtime is on the order of N^2, where N is the length of the list.

The best case runtime of bubble sort is on the order of N.

Why is this?

Best Case: List is already sorted, one pass will finish.

Worst Case: List is in reverse order, need to go through it N times.

$$N \bullet N = N^2$$

Selection Sort

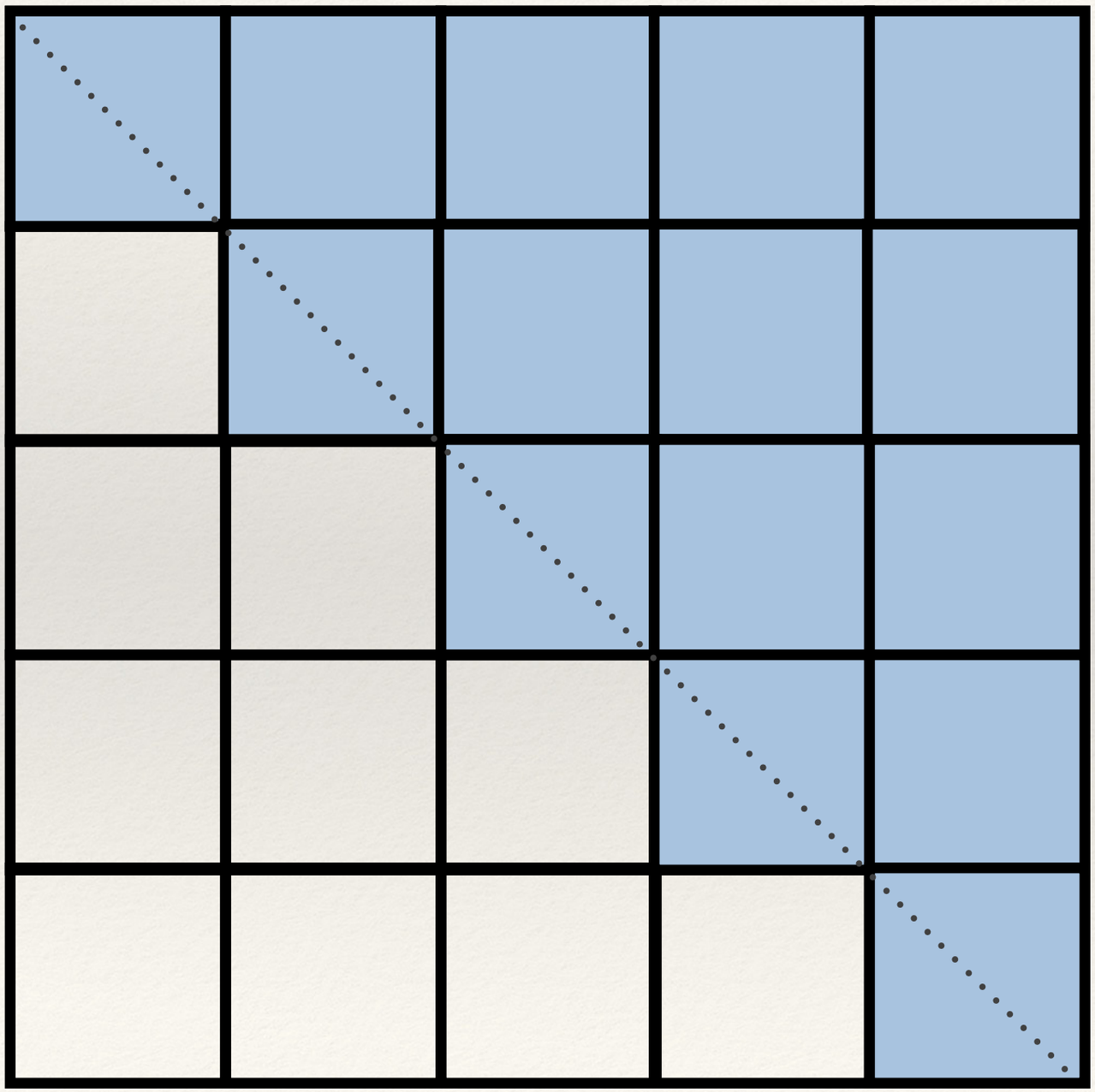
Selection Sort is another simple sort.

The algorithm goes as follows:

1. Go through the list and find the smallest element.
2. Swap it with the first ‘unfixed’ element. That element is now fixed.
3. Continue until the entire list is ‘fixed’ and sorted.

What is the runtime of selection sort? N^2

Selection Sort Runtime: Visual



What is the shaded area?

We can approximate it with a triangle.

Area of a triangle: $N^2 / 2 \sim= N^2$

Insertion Sort

Insertion sort is the last sort you are expected to understand.

Algorithm:

1. Start at second element.
2. Swap it backwards until it's in place.
3. Whatever is at the start of the list is 'fixed'.
4. Continue with every element until sorted.

Best Case: N (list is already sorted)

Worst Case: N^2 (list is in reversed order)

Insertion Sort: A Closer Look

A more detailed examination of insertion sort reveals that it's runtime is not only dependent on the number of elements.

It is also dependent on the number of inversions in the list (so is bubble sort, but it's so bad we aren't going to consider it)

We can see that insertion sort only swaps elements that are out of order (remember the definition of inversion?)

We also know that it must look at each element exactly once, and only swaps as long as there are inversions to the left.

Therefore, the actual runtime of insertion sort is dependent on N + K ,where K is the number of inversions.

In out of order lists, K tends to be around N^2, so we ignore the smaller N term.

Mergesort

Mergesort takes a different approach to sorting. I will talk about the recursive mergesort.

Algorithm:

1. If the length of the list is 1, return the list.
2. Else -
    1. Split the list in half.
    2. Mergesort the left and the right.
    3. Now merge the left and right sorted halves.

Merging

Merging two lists is simple is principle.

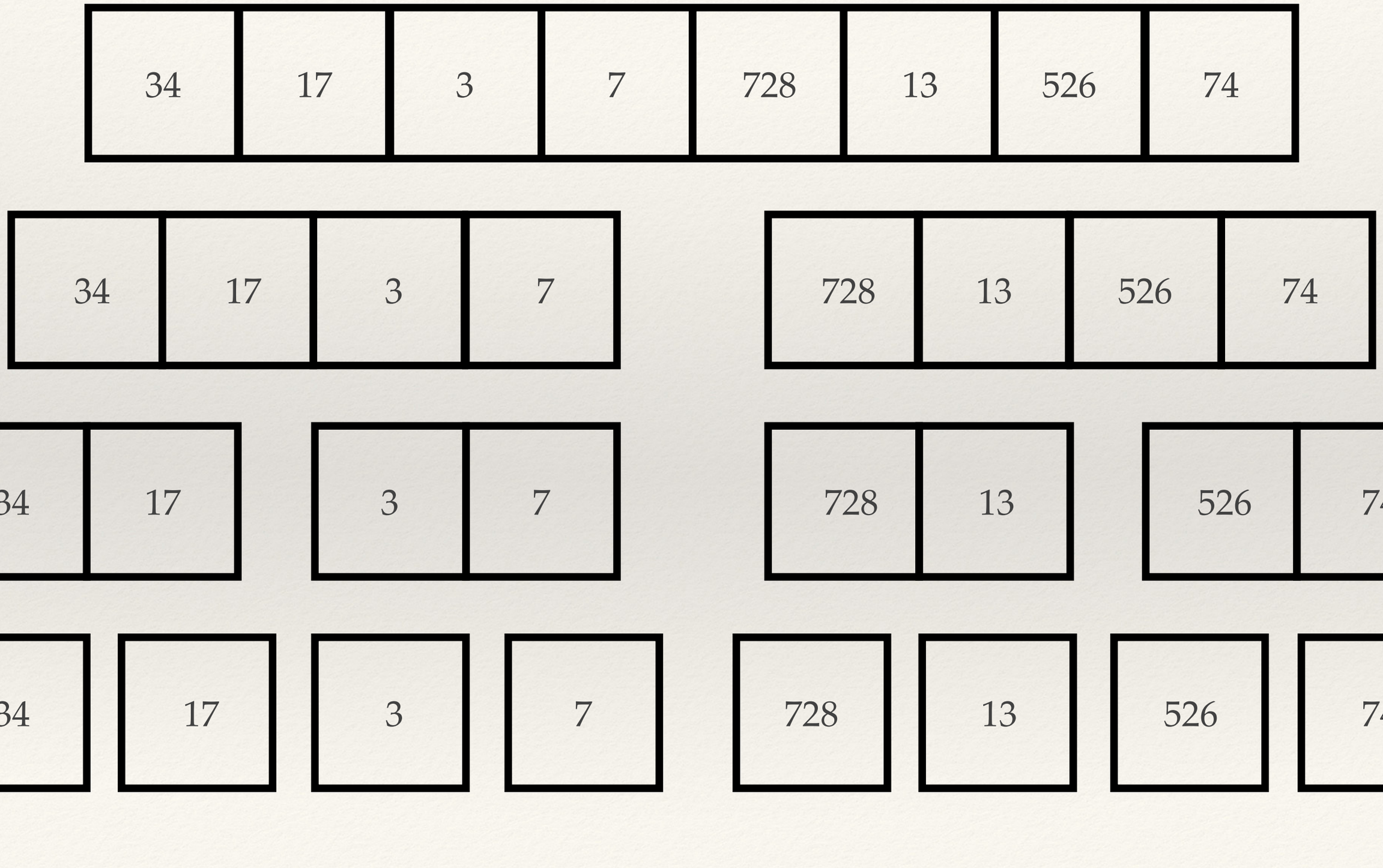
1. Create a new, empty list that can hold both the lists.
2. Starting at the lowest element of each list, insert the smallest at the start of the new list.
3. Move to the next element of the new list and the list you inserted from.
4. Repeat until the new list is full and return it.

Mergesort

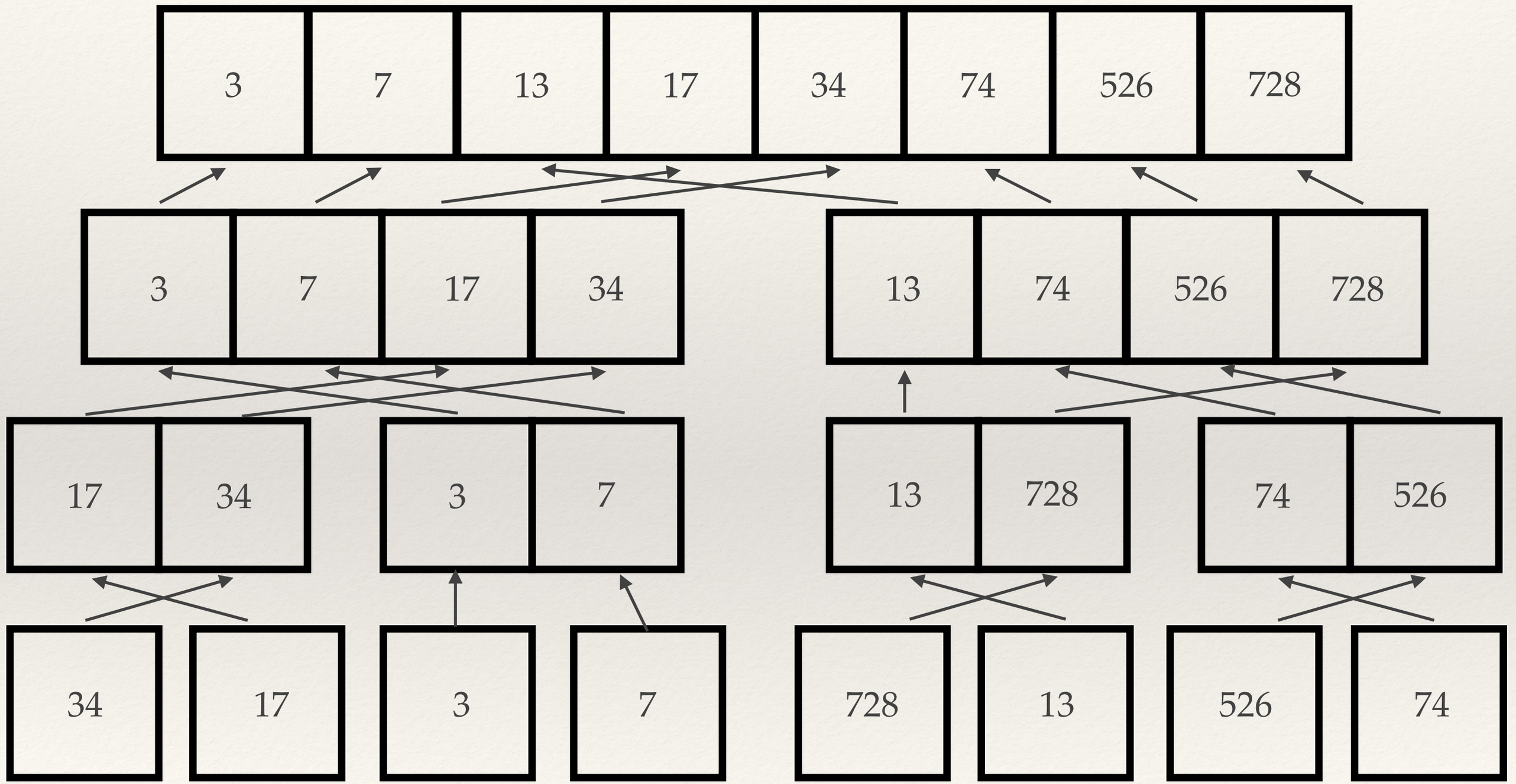
What mergesort does is break a list into sublists and sorts those.

It does this by the merging algorithm shown on the last slide.

Mergesort Demo



Mergesort Demo



Questions?

(Or extra demo requests)

Citations

- Hug, Josh. “Lecture 27: Sorting 1” Lecture 27: Sorting 1. Web. 17 June 2015.
- Hug, Josh. “Lecture 29: Sorting 2” Lecture 29: Sorting 2. Web. 17 June 2015.