- Lab section changed
- Lab 2 posted
- Lab 1 due

Mergesort

\[
\begin{array}{c}
A[\ldots \to r] \text{ unsorted} \\
\downarrow \\
A[\ldots \to q] \\
\downarrow \text{sort} \\
A[\ldots \to t] \\
\downarrow \\
A[\ldots \to \text{merge}]
\end{array}
\]

\[
\begin{array}{c}
A[\ldots \to r] \text{ unsorted} \\
\downarrow \text{divide} \\
A[\ldots \to q] \\
\downarrow \text{sort} \\
A[\ldots \to t] \\
\downarrow \\
A[\ldots \to \text{merge}]
\end{array}
\]
Dot tree: n=4

```
      0 1 2 3
     /     /
     4 3 2 1
    /     /
   4 3   2 1
  /     /   /
 4 3   2 1
 /     /   /
3 4   1 2
 /     /   /
1 2 3 4
```

down

uv (return)
Box trace n = 7

0 1 2 3 4 5 6
2 7 4 1 6 3 5

down

3

up

1 2 4 7

3 5 6

1 2 3 4 5 6 7
How do we merge?

1. Let \( n_1 = \text{length}(A[p...q]) = q-p+1 \)
2. Let \( n_2 = \text{length}(A[q+1...r]) = r-(q+1)+1 = r-q \)
3. Copy \( A[p...q] \) into \( L[0... (n_1-1)] \)
4. Copy \( A[q+1...r] \) into \( R[0... (n_2-1)] \)
5. Place items in \( LL \) and \( RL \) back into \( A[p...r] \) in correct order.

**Exercise**: Re-write MergeSort.java so as to sort a String array.
Chapter 4: Abstract Data types (ADT).

An ADT consists of 2 things:

1. A set $S$ of mathematical structures called `states` or `values`. This is the data.

2. An associated set of `operations` to be applied to the states in $S$. 
Ex: List of Integers ADT

states:
S is the set of finite sequences of integers, of some fixed max length.

operations:
- `createList()`:
  creates an empty list.
- `isEmpty()`:
  determine if a list is empty.