Motivation

- For insertion sort (and other problems) as $n$ doubles in size, the quadratic quadruples!
- Can we decrease $n$?
- What if we **Divide** the sort into smaller pieces?
- We can then solve those (**Conquer** them).
- We need to be able to combine the pieces in a manner simpler than quadratic.

Divide and Conquer

- Divide (into two equal parts)
- Conquer (solve for each part separately)
- Combine separate solutions
- Merge sort
  - Divide into two equal parts
  - Sort each part using merge-sort (recursion!!!)
  - Merge two sorted subsequences

Merge Sort

```c
MergeSort(A, left, right) {
    if (left < right) {
        mid = floor((left + right) / 2);
        MergeSort(A, left, mid);
        MergeSort(A, mid+1, right);
        Merge(A, left, mid, right);
    }
}
```

// Merge() takes two sorted subarrays of A and // merges them into a single sorted subarray of A // (how long should this take?)
**Merge Sort: Example**

- Show `MergeSort()` running on the array

```java
A = {10, 5, 7, 6, 1, 4, 8, 3, 2, 9};
```

**Analysis of Merge Sort**

**Statement**

```java
MergeSort(A, left, right) {
    if (left < right) {
        mid = floor((left + right) / 2);
        MergeSort(A, left, mid);
        MergeSort(A, mid+1, right);
        Merge(A, left, mid, right);
    }
}
```

**Effort**

- So \( T(n) = \Theta(1) \) when \( n = 1 \), and
- \( 2T(n/2) + \Theta(n) \) when \( n > 1 \)
- So what (more succinctly) is \( T(n) \)?

**Recurrences**

- The expression:
  
  \[
  T(n) = \begin{cases} 
  c & n = 1 \\
  2T\left(\frac{n}{2}\right) + cn & n > 1 
  \end{cases}
  \]

  is a recurrence.

  *Recurrence: an equation that describes a function in terms of its value on smaller functions*

**Recursion Tree**

- n comparisons per level
- \( \log n \) levels
- total runtime = \( n \log n \)
Recurrence Examples

\[ T(n) = \begin{cases} 
0 & n = 0 \\
1 + T(n-1) & n > 0 
\end{cases} \]

Recurrence Examples

\[ T(n) = \begin{cases} 
1 & n = 0 \\
1 + T(n-1) & n > 0 
\end{cases} \]

Recurrence Examples

\[ T(n) = \begin{cases} 
2T\left(\frac{n}{2}\right) + c & n > 1 
\end{cases} \]

Recurrence Examples

\[ T(n) = \begin{cases} 
c & n = 1 \\
aT\left(\frac{n}{b}\right) + cn & n > 1 
\end{cases} \]
Chapter 4 will look at several methods to solve these recursions:
- Substitution method
- Recursion-tree method
- Master method