InsertionSort

- Idea: At step `k`, insert the `k`th element in the correct position
  - Sort first two elements
  - Now insert 3rd element in order
  - …
- Loop invariant ("when loop index is `i`"):
  - First `i` elements are in sorted order
- Time:

  Best-case: _____ Worst-case: _____ "Average" case: _____
- Characteristics:

  Stable: _____ In-place: _____

Demo: https://docs.google.com/presentation/d/10b9aRqpGJu8pUk8OpfqUIEEm8ou-zmmC7b_BE5wgNg0/present

SelectionSort

- Idea: At step **`k`**, select the smallest elt and put it at **`k`**th position
  - Find smallest element, put it 1st
  - Find next smallest element, put it 2nd
  - …
- Loop invariant ("when loop index is **`i`**"):
  - First **`i`** elements are the **`i`** smallest elements in sorted order
- Time:

  Best-case: _____ Worst-case: _____ "Average" case: _____
- Characteristics:

  Stable: _____ In-place: _____

Demo:
https://docs.google.com/presentation/d/1p6g3r9BpwTARjUylA0V0yspP2temzHNJEJjCG41I4r0/edit

In-place HeapSort

- Idea: Put everything in a **MAX** heap ; successively `deleteMax`
  - insert each `arr[i]` –OR – better yet, use `buildHeap`
  - ```
    for(i=0; i < arr.length; i++)
        arr[arr.length - i] = deleteMax();
    ```
- Loop invariant ("when loop index is **`i`**"): same as naïve version
- Time:

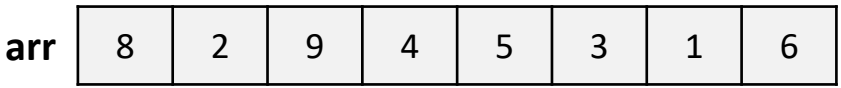
  Best-case: _____ Worst-case: _____ "Average" case: _____
- Characteristics:

  Stable: _____ In-place: _____

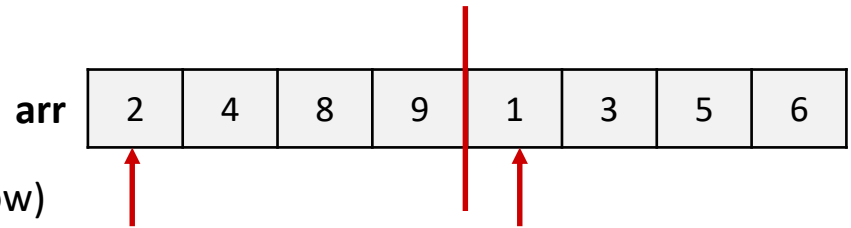
Demo:
https://docs.google.com/presentation/d/1SzcQC48OB9agStD0dFRgccU-tyjD6m3esrSC-GLxmNc/present
    ```

MergeSort: Merging Example (1 of 10)

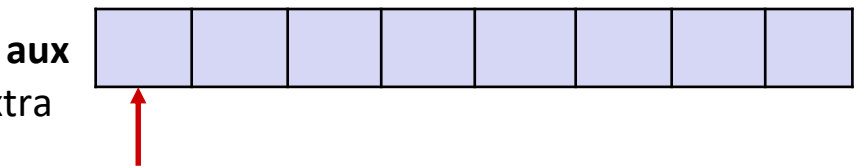
- Start with:



- Return from left and right recursion
  - (pretend it works for now)



- Merge
  - Use 3 cursors and an extra auxiliary array
  - When done, copy the extra array back to the original



Lecture Outline

- Comparison-based Sorting
  - Review
  - Simple algorithms
    - InsertionSort
    - SelectionSort
  - Fancier Algorithms: HeapSort
  - Fancier algorithms using Divide-and-Conquer
    - Intro
    - MergeSort
    - **QuickSort**

Step #3: Recursion (2 of 3)