# CSCI 241

Scott Wehrwein

Incremental vs. Divide-and-Conquer algorithms

# Goals

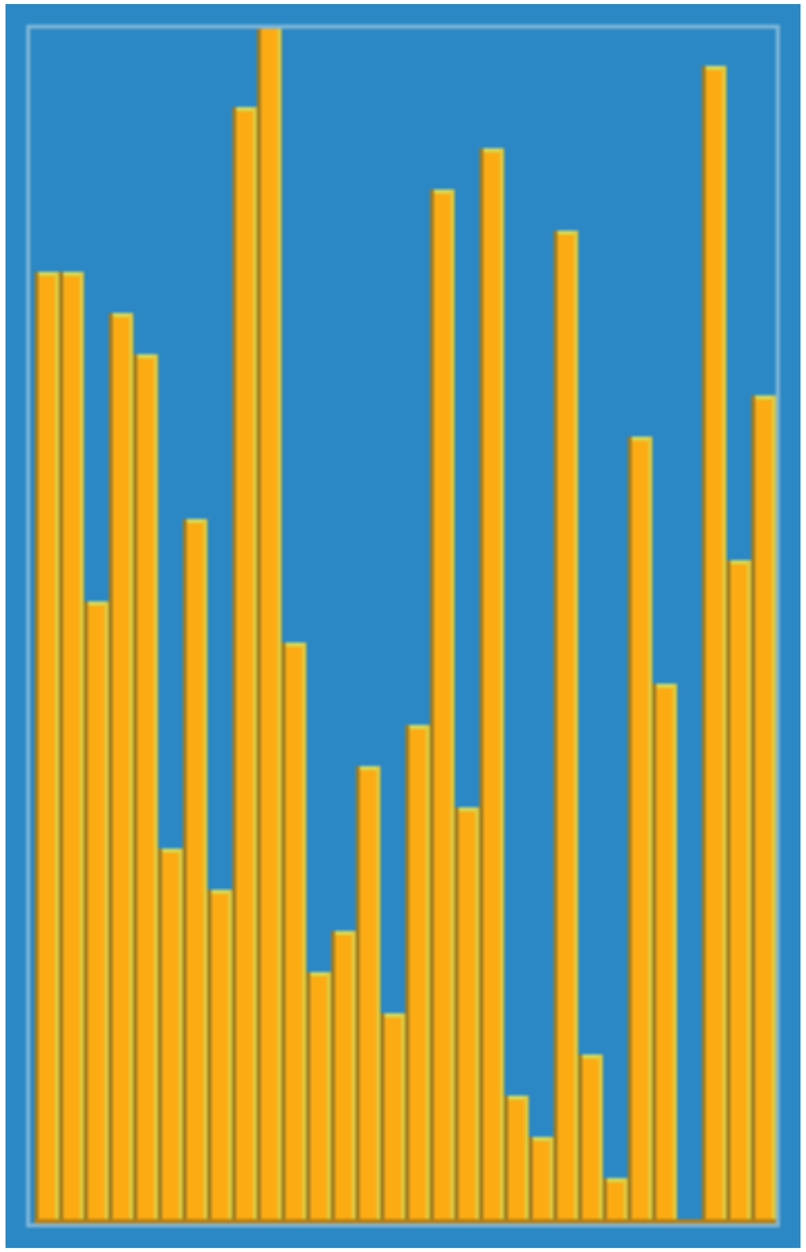
Understand the distinction between incremental and divide-and-conquer algorithms.

Know the generic steps of a divide-and-conquer algorithm.

# **Incremental** Algorithms

solve a problem a little bit at a time.

Natural programming mechanism: loops



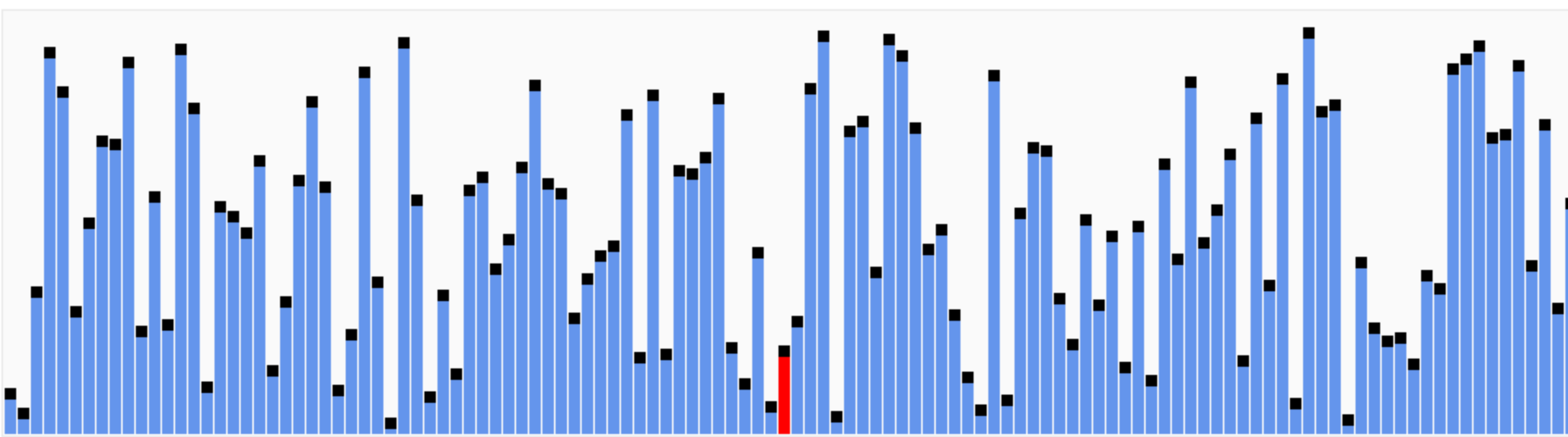
# Divide-and-Conquer

## Algorithms

solve a problem by breaking it into smaller problems.

**(easier!)**

Natural programming mechanism: recursion



https://upload.wikimedia.org/wikipedia/commons/f/fe/Quicksort.gif

Why are we talking about divide-and-conquer, I thought we were learning how to sort things?

# Divide-and-Conquer: By Example

```
/** sort A[start..end] using mergesort */
mergeSort(A, start, end):
  if (end-start < 2):
    return
  mid = (end+start)/2

  mergeSort(A, start,mid)
  mergeSort(A, mid, end)

  merge(A, start, mid, end)
```

**1. Divide**

**2. Conquer**

**3. Combine**

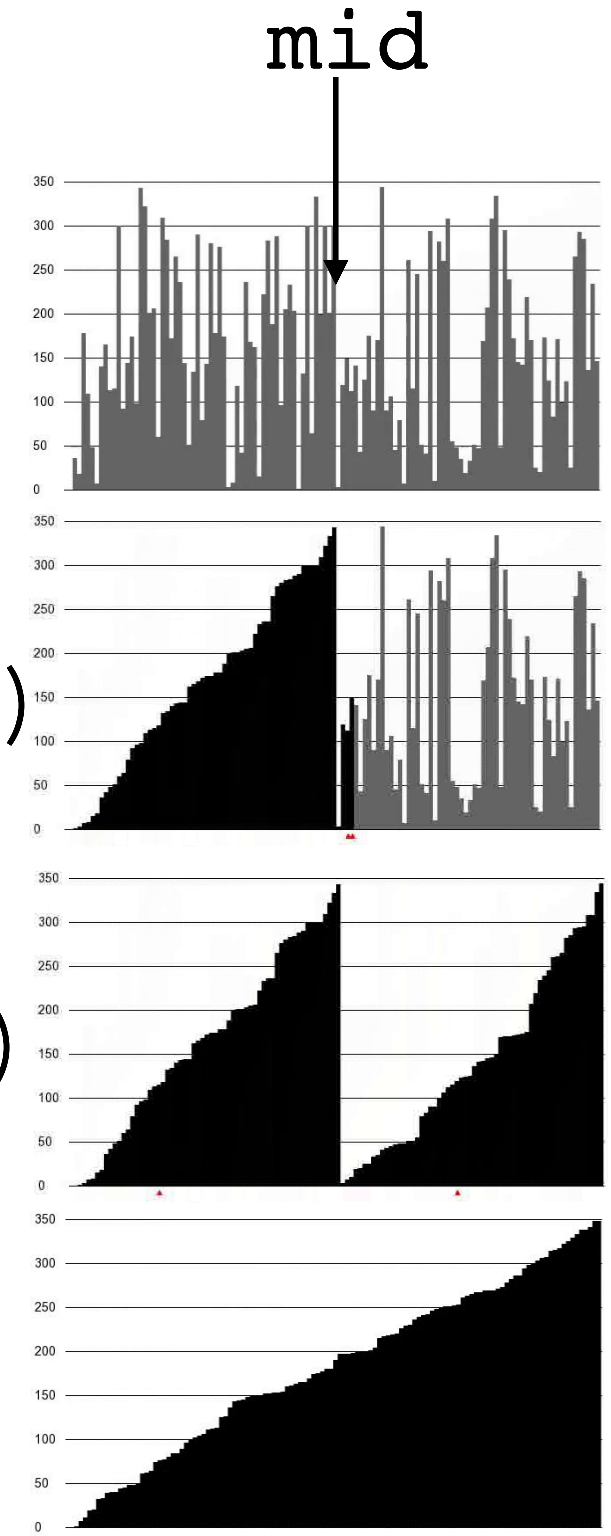
# Divide-and-Conquer: By Example

```
/** sort A[start..end] using mergesort */
mergeSort(A, start, end):
  if (end-start < 2):
    return
  mid = (end+start)/2

  mergeSort(A,start,mid)

  mergeSort(A,mid,end)

  merge(A, start, mid, end)
```




Divide

Conquer (left)

Conquer (right)

Combine

# Divide-and-Conquer can yield better runtimes, and not just for sorting

- Sort n values:
  $O(n^2)$ to $O(n \log n)$

- Multiply two n-by-n matrices:
  $O(n^3)$ to $O(n^{2.81})$

- Find the closest pair of n points in a plane:
  $O(n^2)$ to $O(n \log n)$