Announcements

- Program 5 Due 11/25 at 11:59 PM
- Program 5 Clarification
  - Resizing in add
  - Capacity increment
- Reminder: no class Nov 27 due to Thanksgiving Holiday

Quiz Time ;)

Searching Algorithms

Linear search may require searching all list elements, which can lead to long runtimes if “N” is large enough.

If the list of elements is sorted, there exists a search algorithm that is faster than the linear search algorithm: **Binary Search**

Searching Algorithms (cont.)

Binary search is an algorithm for searching a list if the list's elements are sorted and directly accessible (such as an array).

- **Binary search** first checks the middle element of the list.
    - If the search key is found, the algorithm returns the matching location.
    - If the search key is not found, the algorithm repeats the search on the remaining:
        - left sublist (if the search key was less than the middle element)
        - right sublist (if the search key was greater than the middle element).
        - When searching a sublist, binary search begins in the middle of that sublist and repeats itself until the item is found or there are no more sublists

Searching Algorithms (cont.)

Binary search is incredibly efficient in finding an element within a sorted list.

During each iteration or step of the algorithm, binary search reduces the search space (i.e., the remaining elements to search within) by half.

The search terminates when the element is found or the search space is empty (element not found).

Let’s look at an example...

Searching Algorithms (cont.)

How do we implement this algorithm in code?

Let’s work through it together on the board...

Searching Algorithms (cont.)

Looking at runtime:

If each comparison takes **1 µs** (1 microsecond), a **binary search** algorithms runtime is at most **20 µs** to search a list with 1,000,000 elements, **21 µs** to search 2,000,000 elements, **22 µs** to search 4,000,000 elements, and so on.

Ex: Searching Amazon's online store, which has more than 200 million items, requires less than **28 µs**; up to 7,000,000 times faster than linear search remember **linear search** took 3 minutes...

Sorting

Another common algorithm in computer science is **sorting**.

Sorting is the process of converting a list of elements into ascending (or descending) order.

For example: given a list of numbers: {17, 3, 44, 6, 9}, the list **after sorting** is

{3, 6, 9, 17, 44}.

Sorting (cont.)

Selection Sort

Selection sort is a sorting algorithm that treats the input as two parts, a sorted part and an unsorted part, and repeatedly selects the proper next value to move from the unsorted part to the end of the sorted part.

Let’s look at an example on the board…

Now let’s translate this to code...

Sorting (cont.)

Selection sort has the advantage of being easy to code, involving one loop nested within another loop.

Other sorting algorithms are not so easy to code.

Sorting (cont.)

Insertion Sort

Insertion sort is a sorting algorithm that treats the input as two parts, a sorted part and an unsorted part, and repeatedly inserts the next value from the unsorted part into the correct location in the sorted part.