Computers are handy at reordering elements of a list and for finding a specific element of a list.

- **Sort** -- reorder elements in increasing or decreasing order
- **Search** -- Find a specific element, or determine that it is not in the list

- **How long does it take to perform these techniques?**
  - **Big-O notation**
Sorting and Searching Algorithms

• Sorting list of objects into alphabetical or numerical order
  ➢ Classic application of arrays
  ➢ Three simple algorithms
    – Bubble Sort
    – Selection Sort
    – Insertion Sort

• Searching list of objects for a specific object
  ➢ Another classic application of arrays
  ➢ Three simple algorithms
    – Sequential Search
    – Sort plus Sequential Search
    – Sort plus Binary Search
Bubble Sort

• Bubble sort is a simple algorithm to convert a random list into an ordered list (i.e., sorted)

\[
\text{grade}[0] \leq \text{grade}[1] \leq \text{grade}[2] \leq \text{grade}[3] \leq \ldots \leq \text{grade}[n-1]
\]

• Bubble sort performs the following steps
  ➢ Compare adjacent elements, starting with the first two elements
  ➢ Interchange them if first is larger than second
  ➢ Repeat until you reach the end of the array. End of pass.
  ➢ Perform n-1 passes over the array.
  ➢ To optimize, check if any adjacent elements were interchanged after each pass. If no, terminate.
Bubble Sort Example
Big-O Notation

- Arrays can have very large number of elements \(\rightarrow\) time required to process all elements becomes significant
- Important to determine relative efficiency of different algorithms
- Approximate efficiency of an algorithm as a function of number of items \(N\)
- **Big-O notation** - relationship between processing time and \(N\)
  - \(O(1)\) means "Order 1" which means constant time. Processing time is unrelated to number of items \(N\) (e.g., add 1 to first item)
  - \(O(N)\) means that the growth rate is linear - as \(N\) increases, the processing time increases at the same rate (e.g., add 1 to every item)
  - \(O(N^2)\) means that the growth rate is quadratic - time is proportional to the square of the number of elements (e.g., add every item to every other item)
Rules for Using Big-O

• Big-O provides an upper bound

• To simplify the bound
  ➢ Ignore constant factors: If relationship between processing time and N is determined to be N + 5, then the algorithm is an O(N) algorithm
  ➢ Ignore smaller terms: If relationship is determined to be N^2 + N, then the algorithm is an O(N^2) algorithm
  ➢ Base of log doesn't matter in an asymptotic bound: If relationship is log₂N, then the algorithm is an O(logN) algorithm

• For a sort algorithm, determine Order by examining how many comparisons are required
  ➢ For Bubble Sort,
    \[(N-1) + (N-2) + \ldots + 3 + 2 + 1 = N(N-1)/2\]
    \[O(N(N-1)/2) \Rightarrow O((N^2-N)/2) \Rightarrow O(N^2)/2 \Rightarrow O(N^2)\]
Selection Sort

• A selection sort is a simple algorithm that is more efficient than the sequential sort

• A selection sort works as follows
  ➢ Compare each element with the first element. If the compared element is smaller than the current first element, swap the two.
  ➢ The resulting array is an unsorted list of n-1 elements plus a sorted list of 1 element (the first element is the smallest)
  ➢ Repeat the procedure on the unsorted portion of the array (n-1 elements)
  ➢ On each repetition, the size of the unordered list will be reduced by 1
Selection Sort Example
Order (Big-O) of Selection Sort

• Determine Order by examining how many comparisons are required
  ➢ For Selection Sort,

  \[(N-1) + (N-2) + \ldots + 3 + 2 + 1 = \frac{N(N-1)}{2}\]

  \[O(N(N-1)/2) \rightarrow O((N^2-N)/2) \rightarrow O(N^2)/2 \rightarrow O(N^2)\]
Insertion Sort

• An **insertion sort** sorts each element of a list with respect to the previously sorted elements

• An insertion sort works as follows
  - Compare the first two elements and sort them.
  - Compare the next element in the unsorted list to the elements in the sorted list
  - Insert this element into the appropriate place in the sorted list
    - Question: How is this accomplished? How is the "appropriate place" identified?

• This algorithm is similar to the one you use when ordering a hand of playing cards
Insertion Sort Example

•

Fesq. 4/6/01 1116.070
Order (Big-O) of Insertion Sort

• Determine Order by examining how many comparisons are required
  ➢ For Insertion Sort,

\[
1 + 2 + 3 + \ldots + (N-2) + (N-1) = \frac{N(N-1)}{2}
\]

\[
O(N(N-1)/2) \Rightarrow O((N^2-N)/2) \Rightarrow O(N^2)/2 \Rightarrow O(N^2)
\]
Sorting and Searching Algorithms

• Sorting list of objects into alphabetical or numerical order
  ➢ Classic application of arrays
  ➢ Three simple algorithms
    – Bubble Sort
    – Selection Sort
    – Insertion Sort

• Searching list of objects for a specific object
  ➢ Another classic application of arrays
  ➢ Three simple algorithms
    – Sequential Search
    – Sort plus Sequential Search
    – Sort plus Binary Search
Sequential Search

• A sequential search starts at the beginning of the list and searches through in sequence

• Simple, though inefficient

• Appropriate if list is not sorted

• If desired item is not in the list, must search entire list

  53 32 42 35 22 24 43 33 45 25 42 34 42 53 22 25 52 24 32 54 34 22 55 35 43

  – Is 54 in this list?
  – Is 23 in this list?

• Worst case, this search involves searching all n elements

• Order (Big-O) for Sequential Search

  \[ O(N) \quad \text{Average } N/2 \Rightarrow O(N) \]
Sort plus Sequential Search

• A sequential search can be improved by first sorting the list

• Search can now be terminated if
  ➢ Desired item is found, or
  ➢ The search reaches an element that is greater than the desired element

  22 22 22 24 24 25 25 32 32 33 34 34 35 35 42 42 42 43 43 45 52 53 53 54 55

  – Is 54 in this list?
  – Is 23 in this list?
Sort plus Binary Search

- Once a list is ordered, it can be searched using a divide-and-conquer strategy known as the **binary search** algorithm
  - Select the list element that is halfway down the list and compare it to the desired object
  - If match, search is successful and is terminated
  - If no match, determine if mid-point element is greater than, or less than the desired object. This will rule out half of the list.
  - Repeat procedure.
  - Continue until the desired object is found, or the list is exhausted
Sort plus Binary Search Example
Order (Big-O) of Binary Search

• Very efficient - in general, $n$ comparisons required for a list of $2^n - 1$ members

• Search interval is halved each time
  
  $N, \frac{N}{2}, \frac{N}{4}, \ldots, 4, 2, 1 \rightarrow O(\log_2 N + 1)$
Review

• Sort and search algorithms, and their complexity order - Big-O

• **Read C11.5-11.6** with this lecture. Nice coding examples

• Reference: Website address for Big-O Notation: http://academic.emporia.edu/pheattch/cs340f99/hand15.htm

• PS 8 is on the web, and is due **Thursday, 4/19** at Recitation!

• Next week
  
  ➢ **Review C11.4** on dynamic memory allocation
  
  ➢ Terry Smith from Draper Laboratory will be substitute lecturer Monday and Wednesday
Space News Interview with Dan Goldin, NASA Administrator
March 01

Question: What are the two or three things really contributing to the projected overrun on the space station program?

Goldin: It’s logistics cost, the integration costs and software maintenance. It’s enormous. Much bigger than we had projected.... I think all of America is learning some interesting lessons about software.