Searching and sorting algorithms

An important concept for both searching and sorting algorithms is the algorithm's running time. The running time is how we compare one algorithm to another, and determine which algorithm is more efficient. We will also consider storage requirements when comparing two algorithms.

Running time = the number of operations needed to complete the search or sort (typically, the number of comparisons we have to make)

Sequential search

- Easiest and most straightforward searching method
- Start with unsorted array
- Test each slot in the array, starting with the first slot
- Continue until the item is found or until the end of the array is reached
How long does sequential search take?

- Best case: find item at the beginning of the array
  - 1 operation
- Worst case: find item in the last slot in the array, or number is not in the array
  - N operations (array has N items)
- Average case: find item somewhere in the middle of the array
  - N/2 operations

How long does binary search take?

- Best case: item is in the middle of the array
- Worst case: item is at the beginning or end, or on either side of the middle
- Average case: anywhere else
- Q: how many operations are needed?

Binary search

- Start with a sorted array
- Each time through, we compare the midpoint of the array to the item we're trying to find
- We reset the midpoint depending on whether the item is greater than or less than the midpoint
- We repeat this until we find the item or until there are no more items to be tested

Binary search running time

- Let N be the size of the array we're searching
- 1st iteration: array size is N/2
- 2nd iteration: array size is N/4
- 3rd iteration: array size is N/8
- kth iteration: array size is N/2^k
Binary search running time (cont.)

- Size of array is $N = 2^k$
- Solve for $k$:
  
  \[ k = \log_2 N \]

- The number of comparisons is logarithmic!
- Binary search is much faster than sequential search
  - any logarithmic algorithm is a very good algorithm