Chapter 11

1. Basic Sorting Algorithms
   1. Organized a collection of data into ascending or descending order
   2. Searching for data is the most common tasks performed by computers
   3. When the data size is larged an efficient technique for searching is desirable
   4. However data searches such as binary search require the dta to be sorted
   5. Sorting the data is a step that must precede and are quite valuable
   6. Two sorting algorithms
      1. Internal sort
         1. Requires the collection of data fit entirely on the main memory
      2. External sort
         1. Collection of data will not fit in the computer’s main memory but will reside in secondary storage
   7. sorting through objects
      1. may contain more than one data member tha determines the order of the entire object within the collection of data
   8. Selection Sort
      1. Sort an array into ascending order, you must search it for the largest item, since you want the largest item to be in the last position of the array, swap the last item with the largest item
      2. Sorting compares, exchanges, or moves items, such operations are more expensive than control loops or manipulating array indexes, particulary when the data is more complex than integers or characters
      3. N\*(n-1)/2 to find the largest integer
         1. (n-1)+(n-2)+…1 = n\*(n-1)/2
         2. triangle numbers
            1. first number (1) + n-1 (last) = sum of all pairs of numbers \* number of pairs (n-1)/2 (since pairs)
         3. +3\*(n-1) number of exchanges times 3
   9. Bubble Sort
      1. Adjacent items and exchanges them if they are out of order
         1. Compare the two items in the array. If they are out of order, you exchange them. You then compare the items in the next pair
   10. The Insertion Sort
       1. insertion sort takes the first item of the unsorted region and places it into the proper position.
       2. First step is to move the first element into the sorted region.
          1. Next you take the next item and move it into the next region that would be sorted
2. Faster Sorting Algos
   1. For extermeley large array
   2. Merge and quick sort are divid and conquer sorting algos
   3. Merge Sort
   4. Quick Sort
      1. Consider the first two steps of the pseudocode function kSmall that solves the problem of finding the kth smallest item of the array theArray[first…last]
         1. Choose a pivot value p from anArray[first…last]
         2. Partition the values of anArray[first...last] about p
         3. Choose a pivot and place numbers below and above it. Two sections of smaller than p and greater than p will exist
   5. Median of three pivot selection
      1. Take our pivot as the median of three entries in the array: the first entry, the middle entry, and the last entry