TIME COMPLEXITY ANALYSIS

* Binary Search: searches an array for a specific number by repeatedly cutting the array in half.
  + Best case: the best case scenario when it comes to time is 1. You might be able to find the number you are looking for on the first try. Therefore binary search would return the index number on the first try.
  + Average case: The average case for binary search is log of n. After every iteration of binary search the array is cut in half. This continues until the number is found
  + Worst case: The worst case for binary search is that the number we are looking for is not in the array.
* Bubble Sort: Swaps adjacent numbers if the first one is bigger than the second. The largest number in the array is at the end after each iteration.
  + Best Case: The best case for this sorting algorithm is that the array is already sorted. Bubble sort still goes through the entire array so the best time is n (the size of the array).
  + Average and worst case: In any other case the algorithm will have to make at least one swap. It may have to swap every number if the array is backwards. For both of these cases the time complexity is n2.
* Insertion Sort: This sorts numbers the same way we sort cards. It compares numbers from right to left and inserts the number where it belongs in the array.
  + Best Case: The array may already be sorted and nothing needs to be inserted. In this case the run time would be n.
  + Average and Worst Case: In all other cases a number will need to be inserted somewhere else in the array. One case may even be that the array is backwards. For both of these cases the run time is n2.
* Selection Sort: Repeatedly finds the smallest number and puts it at the beginning.
  + Best, average, and worst case: No matter what the algorithm goes through every single element in the array to find the minimum. Even if the array is already sorted it will always go through every element in the array. This means the run time is n2.
* Quicksort: Picks an element in the array and uses it as a partition. All numbers smaller than the partition will go on the left side and all numbers bigger than the partition will go on the right side.
  + Best Case: The best case occurs when the middle element is always picked as the pivot point. The run time for this case is n log of n.
  + Average Case: The average time for quick sort is n.
  + The worst case: In the worst case the performance for quick sort is n2.