Binary Search

* While loop contains an id, else if, else statements
* The first if condition is met when the target is found
* Algorithm usually performs 5 operations for each iteration of the loop
* Worst case
  + Target is not in the array
  + It is found when the search space consists of one element
  + n/2^k = 1
* Average case
  + It is more likely that the target will be found as the search space becomes small.

Selection Sort

* Unsorted elements: n , n-1, n-2…
* Comparisons: n, n-2,n-3…..

Insertion sort

* Look at notes you shit fuck
* Worts case
  + The array is in reverse order

BIG FUCKING O HERE WE GO

* Linear Search
  + Best Case O(1)
  + Avg case O(n)
  + Worst Case O(n)
* Binary Search
  + Best Case O(1)
  + Avg Case O(log n)
  + Worst case O(log n)
* Selection Sort
  + Best Case O(n^2)
  + Avg Case O(n^2)
  + Worst Case O(n^2)
* Insertion Sort
  + Best Case O(n)
  + Avg Case O(n^2)
  + Worst Case O(n^2

Quick Sort

* Partitions an array until it is sorted
  + Until all partitions consist of at most one element
* Each sub array is divided in half in each partition
  + Watch time a series of sub-arrays are partitioned n comparisons are made
* Log(n) times n has to be divided in half before the result is 1
* Best case: nlog(n)
* Worst case: occurs when the array Is nearly sorted
* Worst case is the same as selection or insertion sort
* Randomize positions of array to get best case for Quicksort

Merge Sort

* Sorts two arrays.
* Divide and conquer
  + Merge two lists on one element each is the same as sorting them
  + Merge sort divides up an unsorted list until the above condition is met and then sorts the divided parts back together in pairs.
  + O(nlogn)
  + Requires additional storage

Heap Sort

* Also O(nlogn)
* Works in place, no additional storage
* Builds a tree in sorted order