Sorting (Part 1)

Lab Recap

During the Runtime Lab we should have discovered:

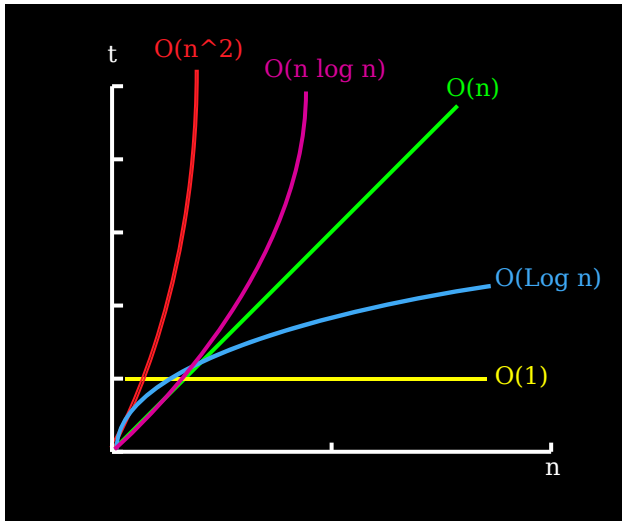
1. Binary Search is considerably faster than Linear Search (especially as the size of our array increases).

2. The time it takes to sort however, is much longer than Linear Search. This makes Linear Search more efficient than Binary Search if we are only searching an array one time.

Algorithmic Efficiency

This efficiency can be quantified using something called `Big Oh` notation (not on AP exam). Big Oh describes how the runtime of an algorithm will increase as the array size (n) increases.

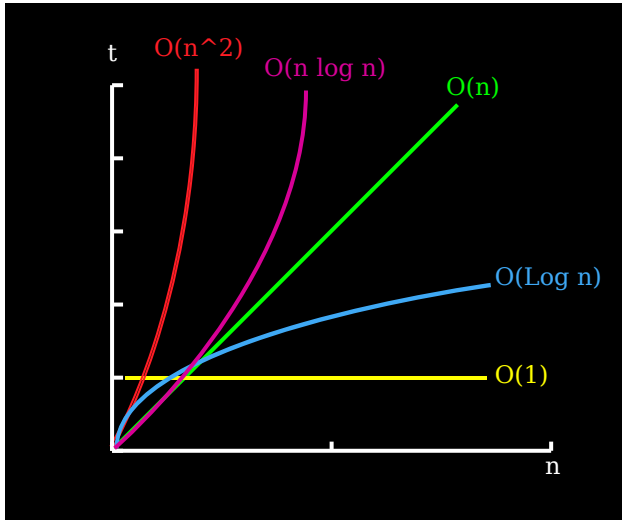
Efficiency	Meaning
O(1)	Takes a constant amount of time regardless of the size of n.
O(log(n))	Time increases with n, however more slowly than linear (e.g. Binary Search)
O(n)	Increases linearly (e.g. Linear Search)
O(n*log(n))	Increases with n, faster than linear (e.g. `Arrays.sort()`)
O(n^2)	Increases to the square of n. (e.g. Insertion Sort, Selection Sort)
O(2^n)	Increases exponentially



Sort Algorithms for the AP Exam

1. Selection Sort (`O(n^2)`) - Today
2. Insertion Sort (`O(n^2)`) - Today
3. Merge Sort (`O(n*log(n))`) - Chapter 10 (Recursion)

note: the `n^2` runtime comes from the fact that selection/insertion rely on *double loops*, resulting in having to iterate through n things n times.



Selection Sort Pseudo-code

```
1 for each position in the array
2     find the minimum value from the current position to the end of the array.
3     if the minimum is not at the current position
4         swap the current and the min
5     move to the next position in the array.
```

Simulation
http://www.cs.armstrong.edu/liang/animation/web/SelectionSort.html

Selection Sort Names

Insertion Sort Pseudo-code

```
1 for each position in the array (start @ index 1)
2     insert the current value into order relative to the sorted values on its left.
3     move to the next position in the array.
```

Simulation
https://www.hackerearth.com/practice/algorithms/sorting/insertion-sort/visualize/

Insertion Sort Names

LAB-027

Implement these to algorithms with the following `static void` methods

1. `selectionSort()`: takes an ArrayList of names and puts them into alphabetical order using the selection sort algorithm. (see `compareTo()` if stuck on how to order)
2. `insertionSort()`: takes an ArrayList of names and puts them into alphabetical order using the insertion sort algorithm.
3. `shuffle()`: takes an ArrayList of names and puts them into a random order. (see `Knuth shuffle` if you get stuck)

Notes:

- Use the starter/test code posted to Classroom to get started
- Try to code these algorithms on your own before resorting to Google. It's worth the effort.
- Google the `compareTo()` method to review how to put Strings in order.
- Your shuffle code needs to randomize the order, but doesn't have to be *mathematically perfect* (every permutation is equally likely)

Sample Output

When your code is working you should get something like this:

```
Console

<terminated> SortAlgs [Java Application] /usr/lib/jvm/jdk-12.0.2/bin/java (Feb 5, 2020, 10:23:54 AM)
Shufle Test:
[Justine, Ariha, Rishab, Shane, Abby, Tony, Charlotte, Leo, Christian, Cullen, Bill, Anthony, Daniel, Adam
[Bill, Adam, Zach, Rishab, Tony, Anthony, Charlotte, Spencer, Daniel, Toby, Ariha, Justine, Abby, Cullen,
[Adam, Daniel, Justine, Shane, Abby, Tony, Christian, Rishab, Bill, Toby, Saad, Spencer, Charlotte, Anthon
[Anthony, Anthony, Charlotte, Rishab, Bill, Justine, Abby, Inica, Cullen, Saad, Leo, Spencer, Shane, Ariha
[Christian, Bill, Cullen, Toby, Adam, Ariha, Zach, Leo, Anthony, Anthony, Inica, Spencer, Shane, Justine,
[Bill, Leo, Ariha, Christian, Charlotte, Cullen, Shane, Rishab, Saad, Anthony, Toby, Inica, Spencer, Zach,

Sort Test:
[Abby, Adam, Adithi, Anthony, Anthony, Ariha, Bill, Charlotte, Christian, Cullen, Daniel, Inica, Justine,
[Abby, Adam, Adithi, Anthony, Anthony, Ariha, Bill, Charlotte, Christian, Cullen, Daniel, Inica, Justine,
[Abby, Adam, Adithi, Anthony, Anthony, Ariha, Bill, Charlotte, Christian, Cullen, Daniel, Inica, Justine,
[Abby, Adam, Adithi, Anthony, Anthony, Ariha, Bill, Charlotte, Christian, Cullen, Daniel, Inica, Justine,
[Abby, Adam, Adithi, Anthony, Anthony, Ariha, Bill, Charlotte, Christian, Cullen, Daniel, Inica, Justine,
[Abby, Adam, Adithi, Anthony, Anthony, Ariha, Bill, Charlotte, Christian, Cullen, Daniel, Inica, Justine,
```