Richard Hayes Crowley

06/26/2021

CSC\_242\_Lab\_05\_QA

**(1)** Which of these orders of magnitude will overpower the others, i.e. which increases in value, as *n* increases?

*O* ( *n* ) , *O* ( *n* 2) , *O* ( log2( *n* ) )

**O(n^2) quadratic is the most expensive order of magnitude in this list, O(log 2 (n)) is the least expensive.**

**(2)** There are two classes of Sorting Algorithms, as listed below.

*O* ( *n* 2 ) *O* ( *n* log ( *n* ) )

Bubble Sort Heap Sort

Insertion Sort Merge Sort

Selection Sort Quick Sort

Shell Sort

Do these orders of magnitude ( Big *O* ) represent Best Case, Worst Case or Average Case?

**These classifications represent the worst case. Big 0 classifications are typically ascribed to the worst case of complexity.**

**(3)** What is a bucket sort? Is it an effective sorting routine?

**A bucket sort splits a list into sublists (buckets), sorts each bucket, and then stitches the list back together. It’s worst case complexity is O(n^2) and it’s average complexity is O(n), which means it’s not as fast as quicksort. However, reading up on it, it seems to have its place when sorting huge lists that cannot fit into memory, and may be used to sort chunks of a huge list and then stream those chunks through e.g. a network service**

**(4)** Haveyou ever studied any searching processes different than a Linear Search or a Binary Search? If so, comment on the type of process.  **I’ve studied bisection search in the open source MIT 6.00.1 python course on EdX, which makes use of an “epsilon” value, or an acceptable margin of difference, for calculating the closest possible square or cubed root of a number, which inevitably involves working with floats. Having an “epsilon” or “close-enough” value is an interesting concept to integrate into searching algorithms, especially those that deal with numbers / floats.**

**(5)** What have you learned from performing and coding for this lab assignment?

**I had an excellent review of the algorithms and complexity analysis that I studied in Chapter 3, and also built a tool / Class that I can refer to in the future whenever I need to reference or use a binary search or quick sort!**