Nathan C. Walk, MD
10/8/20
BIS 634 HW 2

Exercise 1:
see code

My strategy for testing was to to use the PMID you provided and ensure results came back consistent with you as an author on 3 of those papers. I also ran other PMIDs and got back answers, but I did not cross reference to verify them, I'm afraid.

Exercise 2:
see code: sort.py works, and is the serial problem.
sort6.py does not work, but is as good as I got….there is something wrong with the pool.map statement.

The place easiest (and I believe largest independent pieces) to separate is during the sorting -into left and right halves, then sending the sorted lists to a final merge. I read some ingenious ways to do this online with limiting depths of layers corresponding to number of processes, etc, but it was beyond my capabilities.

Time for serial: 9.0623556 seconds

10th number: 32.308558
353,237 number: 38.119757
[both the 10th and 353,237 have more significant digits in the script]

If we were to do 3 or more processes, it would depend on how many. Less than the number of cores?...the same way, I think. More than 4, the number of cores my computer has, a bit trickier. Assuming they were all similar in turns of computational demand (memory,etc), divide them accordingly by 4???? So 8/4 - send 2 processes to each core. Repeat as you ramp up.

Exercise 3:

a. What is the other quantity that governs the runtime and memory usage?

The size of the difference between the largest and smallest integer - this size dictates the size of the counts array - which is filled, initially, with zeros.

b. How does the run-time of this algorithm scale with respect to the 2 or 3 quantities?

The run time of this algorithm will scale O(n) with the number of integers, and O(magnitude of difference between largest and smallest integers) - the magnitude of this difference would be the size of the counts array.

c. Example of a problem where it would be faster to use this sorting algorithm than the merge sort.

I believe it would run O(n) time for situations where the difference between largest and smallest integers is not too great (O(2n) would reduce to O(n)). In these situations it would be faster than a merge-sort algorithm of O(nlogn). How great of a difference?...I think anything less than an order of 10^9 difference would be faster with this algorithm.

I'll tell you the truth though, I'm concerned about my answer here. I can't find this algorithm anywhere on the web, and there must be a reason that it isn't used. Soooo, …..

d. Example of a problem where it would be preferable to use a merge sort instead of this algorithm.

Well, the converse of the above, I suppose.

Exercise 4:

As I'm doing an Informatics fellowship, and trying to get published, I'd like to use the CDW's dataset at the VA. This dataset is sourced from and accurately reflects the contents of VistA, the VA's EMR. Data is available from October 1998 to the present. It is an SQL based relational database. The data domains are divided into production and raw, broadly speaking, with the production domain formatted for easier use and updated nightly. CDW has abundant metadata to help a user find and scope their queries. CDW collects 60 domains of data (e.g. demographics, laboratory results, medication orders, barcode medication administrations, vital signs, etc.) selected by clinical experts and operational leaders.

There are millions of rows of data, and with 60 domains, I'll have to estimate the variables at thousands. I cannot find definitive information on this.

One has to apply for permission, and I'm told I need an IRB to do work on it.

I hesitate to search for another data source which I could explain in more detail - the point of my fellowship and learning through these courses, which I'm grateful for, is to publish to achieve my career pivot in academics/biotech.

Thanks for your patience in reading over this - I realize it's not complete, but I've worked diligently, and am out of gas. After all, I'm only auditing:) I sincerely appreciate your time.