name: Linus R.

3.2.3 Databases -- How they are Organized and How they are Queried

There are really two main points in this activity, and they are both very introductory. The first has to do with how data is stored and organized in databases, and how data is added to, modified in or retrieved from a database.

The second concept is the concept of algorithmic efficiency, how long (or how many resources) an algorithm takes to run.

DATABASES

Data management is a whole subfield with IT. Careers in database management are generally in high demand and the pay is lucrative. You could take whole courses on how to manage databases (I did take a one semester course on it at OHSU), but it's really not very difficult to learn. It works very logically. Once you learn the rules of the logic and practice you can pick it up fairly quickly.

What is a great exercise to learn the power of data and the internet is to make a web site and a database that can be accessed from the website. Several years ago we used to do that in this course, but we just don't have the time this year. The important thing to know is that you can write programs for your website that will allow your website's users to send data to be put in the database or to send data back to users who requested certain information. Another word for asking to see if something is available or to ask a question is to make a **query**. The language that is used to make queries on databases is called Structured Query Language, SQL.

Unfortunately, we don't have the time to make and populate our own databases this year, but we will get some practice accessing data from publicly available data sets using SQL.

There is a certain way that all data in databases are stored. First of all, the data is stored in **tables**. Each table is made up of rows (**records**) and columns (variables or **fields**). Each row is a separate data entry and includes values for each of the variables or fields you are keeping track of for each entry. For example, the Department of Motor Vehicles would have a database on vehicle registrations in the state. One table would be of all of the registered vehicles. Each row would be a unique vehicle (with its unique VIN). Each row would have values for the vehicle's type, make, model, year, VIN and owner. This table would be associated with a Owners table that would include records of owners.

Each record would include values for the owners first name, last name, address, phone number, driver's license number, etc.

OK, let's get going. Open up PLTW activity 3.2.3 and read through the introduction, though it kind of states the obvious that correlation does not imply causation.

Information from Data

A quick review: How many bytes are in a terabyte (TB)?
About 1 trillion

b) How many bits are in 1 TB?
About 8 trillion

Go through steps 1-16 to set up your table for analysis.
5. a) Look at the schema for the table. How many fields are there?
29

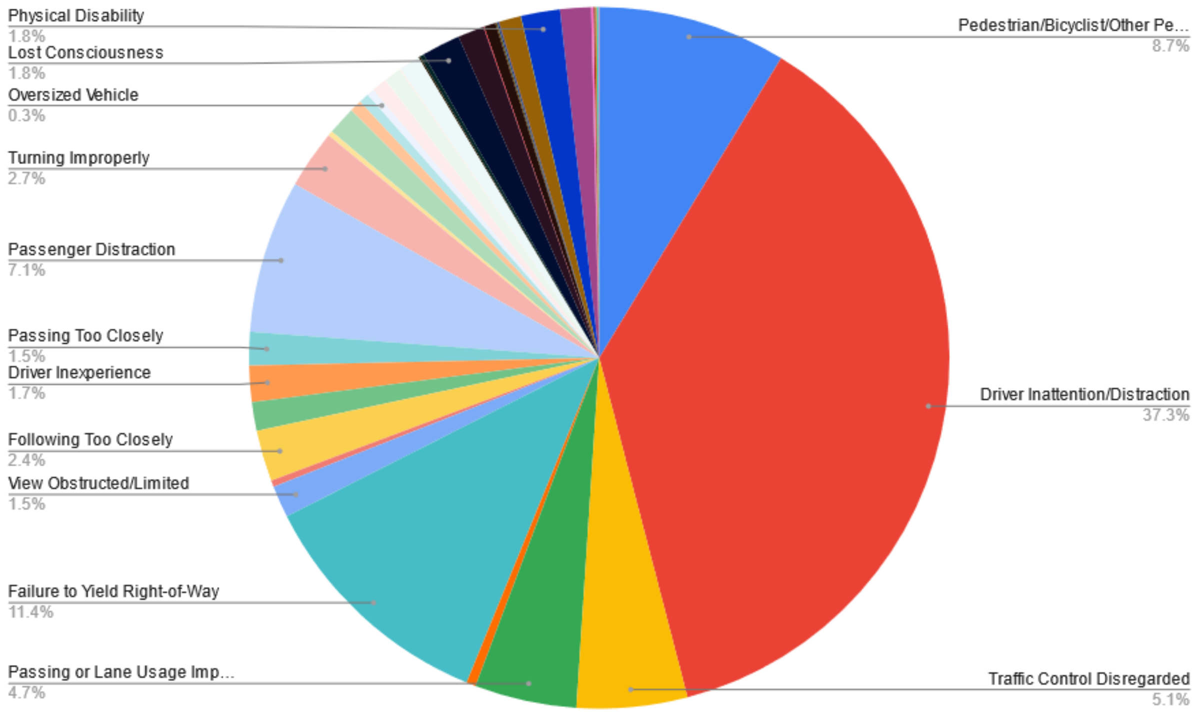
b) What are the four datatypes for this collection of fields?
Plain text, number, location, date and time

17 a) What is the size of this table (in terms of bytes)?
1616000 bytes

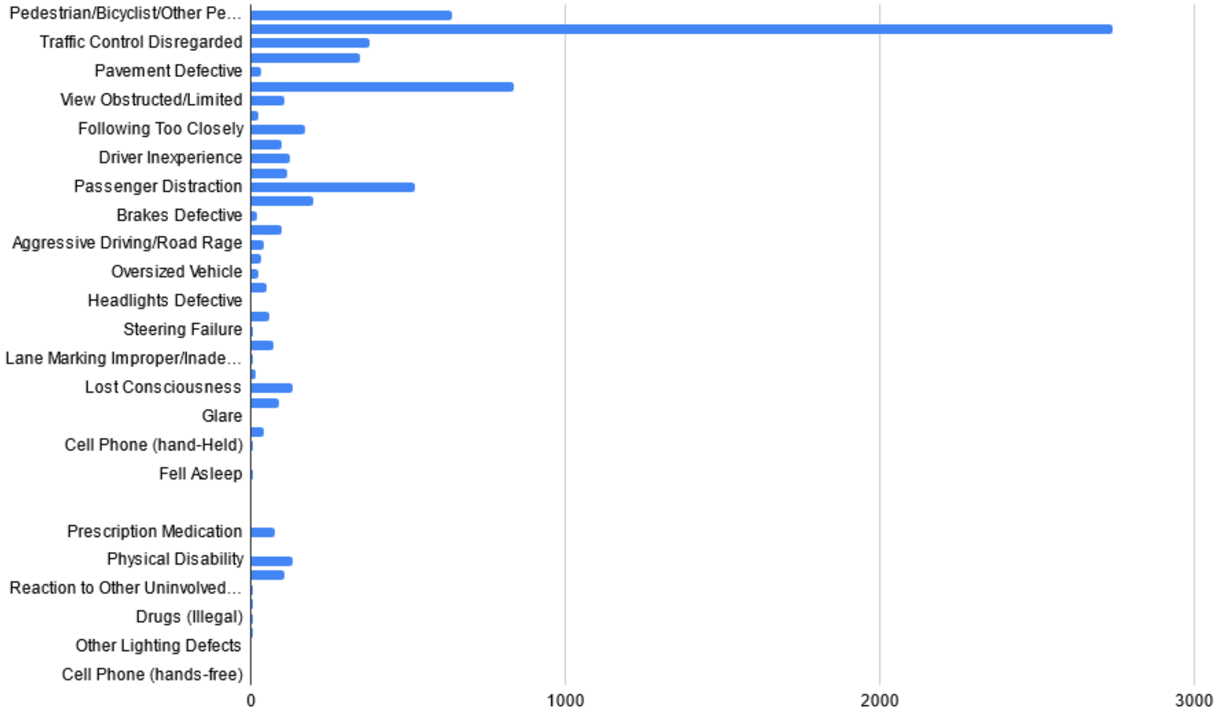
b) How many records are in the table?
7357

c) How many rows are in your sheet?
7357

d) Create that pie chart (much to my horror) and paste a snip of it here.



e) Create a bar chart with the same data. Paste a snip of your bar chart here.



17. a) OK, maybe there is some use in a pie chart to quickly show you the portion something is of the whole. What is the most common cause of collisions with cyclists in Manhattan leading to injury or death?
Driver inattention

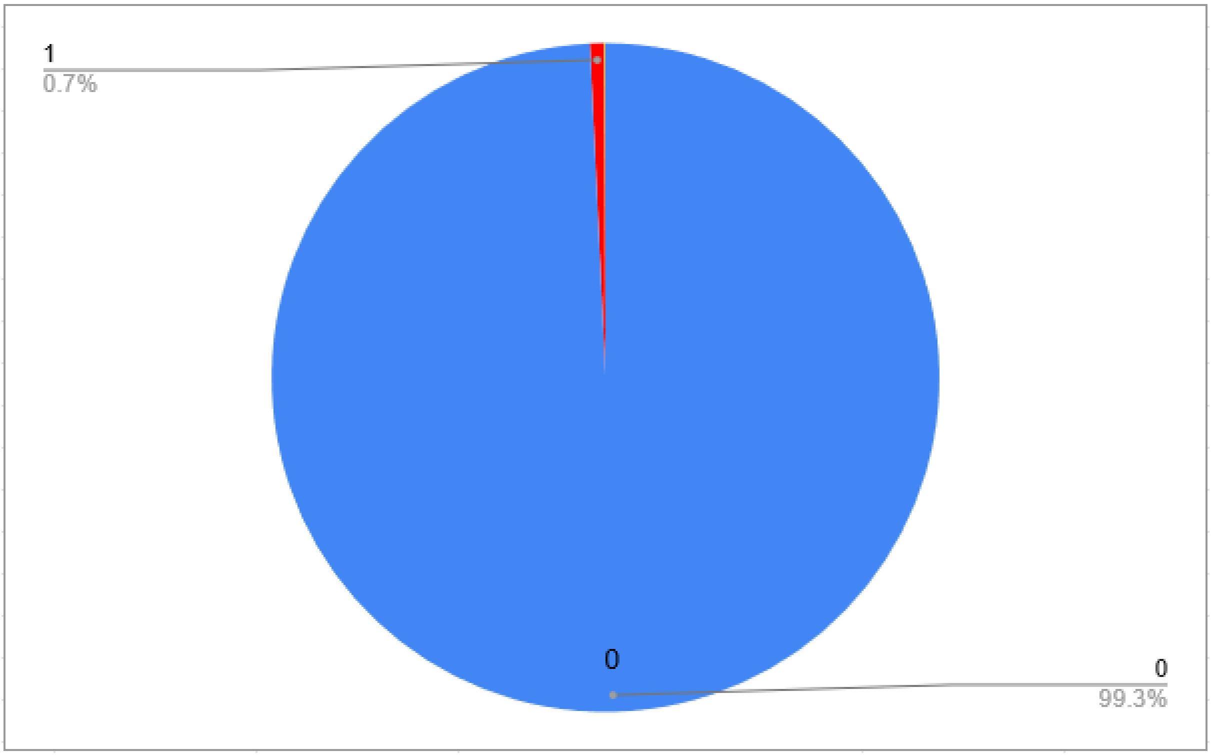
b) What was the percentage of that cause of all of the collisions shown?
37.3%

c) What was the second leading cause?
Failure to yield right of way

d) What was the percentage of collisions associated with this cause?
11.4%

19. a) Come up with a question you would like to explore from your table or from the NYC collision data set (or make a hypothesis). What is your question or hypothesis?
How often is a pedestrian injured from a crash?

b) Make a graph from the results of your query to show the information you found that is relevant to your question. Paste a snip of it here.



c) Give a response to your question in b) based on what you found.

Pedestrians are injured from a crash a bit over 0.7% of the time, as that 0.7 doesn't include the few crashes that injure more than one pedestrian, as that number is too small to show up on the graph.

Efficient Data Processing -- Run-time Analysis -- Algorithm Efficiency

For students taking the AP Exam: What is the name of the search you need to know that works much faster than a sequential search, but in order to do it the collection must be sorted?
Binary search

20. Open up the sorting algorithms page in a new tab. If you take the CSA course, you will write methods for three or four different types of sort algorithms, including insertion sort and merge sort. It is enough for us this year to analyze how efficient they are. And even in this there are many ways we can analyze an algorithm for efficiency: by time, by the number of comparisons, by the number of switches, etc.

Time is the most frequently used measure of efficiency. For this you can play the little videos they have for each sort method to get an idea of the time differences.

How long did it take to sort the list they had in the video for each of the sort methods:
i. insertion sort: 16s
ii. bubble sort: 15s
iii. merge sort: 6s
iv. quick sort: 3.3s

Which sort method seems to be most efficient?
Quick sort

Which seems to be least efficient?
Insertion sort

(btw, it depends on the list that's given…some sorts have strengths with some types of lists and weaknesses with others.)

(This section in blue is optional) The following part (b, d, f, g and 21) is optional, but I recommend it for those that want to work on their programming skills. You can put your answers right in the table underneath this section if you wish.

b) How many comparisons are made in the insertion sort for the list (i.e. how many times is line 9 executed)?
[12, 5, 11, 6, -3, -4, -11, 6, 3, 4, 1, -2]

d) How many comparisons are made in the insertion sort if you add the following items to the existing list?
`17, 9, 13, 8, 7, -5, 6, 11, 3, 4, 1, 2`

`f)` How many comparisons are made in the insertion sort if you add the following items to the existing list?
`-8, 15, 25, -2, 0, 3, -4, 27, 13, 15, 10, 8`

`g)` How many comparisons are made in the insertion sort if you double the existing list by copying all of the elements and then adding them again into the list?

21. Now do the same for the merge sort (you will have to tally up the number of times lines 7, 23, and 25 are executed). Declare your variable initialized in the main as global in each of the functions.

b) How many comparisons are made in the merge sort for the list (i.e. how many times is line 9 executed)?
[12, 5, 11, 6, -3, -4, -11, 6, 3, 4, 1, -2]

d) How many comparisons are made in the merge sort if you add the following items to the existing list?
`17, 9, 13, 8, 7, -5, 6, 11, 3, 4, 1, 2`

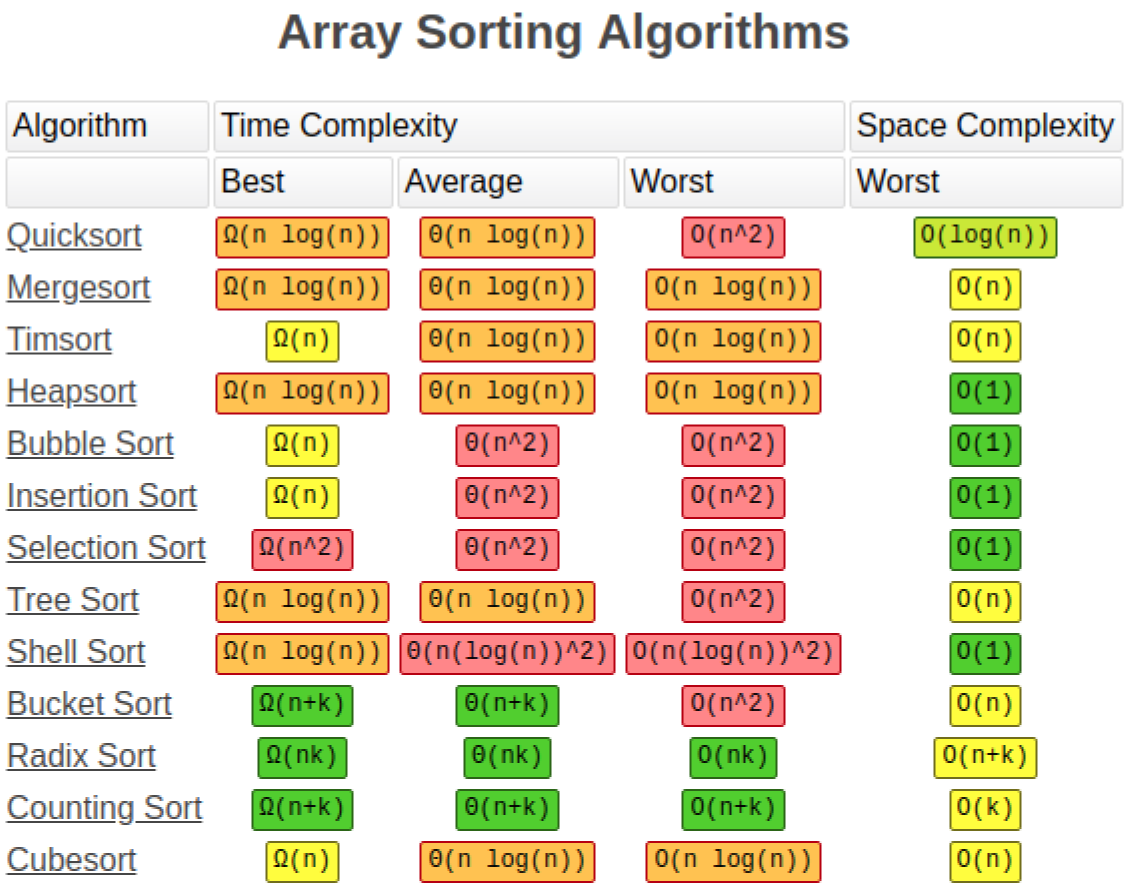
`f)` How many comparisons are made in the merge sort if you add the following items to the existing list?
`-8, 15, 25, -2, 0, 3, -4, 27, 13, 15, 10, 8`

`g)` How many comparisons are made in the merge sort if you double the existing list by copying all of the elements and then adding them again into the list?

# of elements	comparisons in insertion sort	comparisons in merge sort

12	46	72
24	144	186
36	281	384
72	1178	770

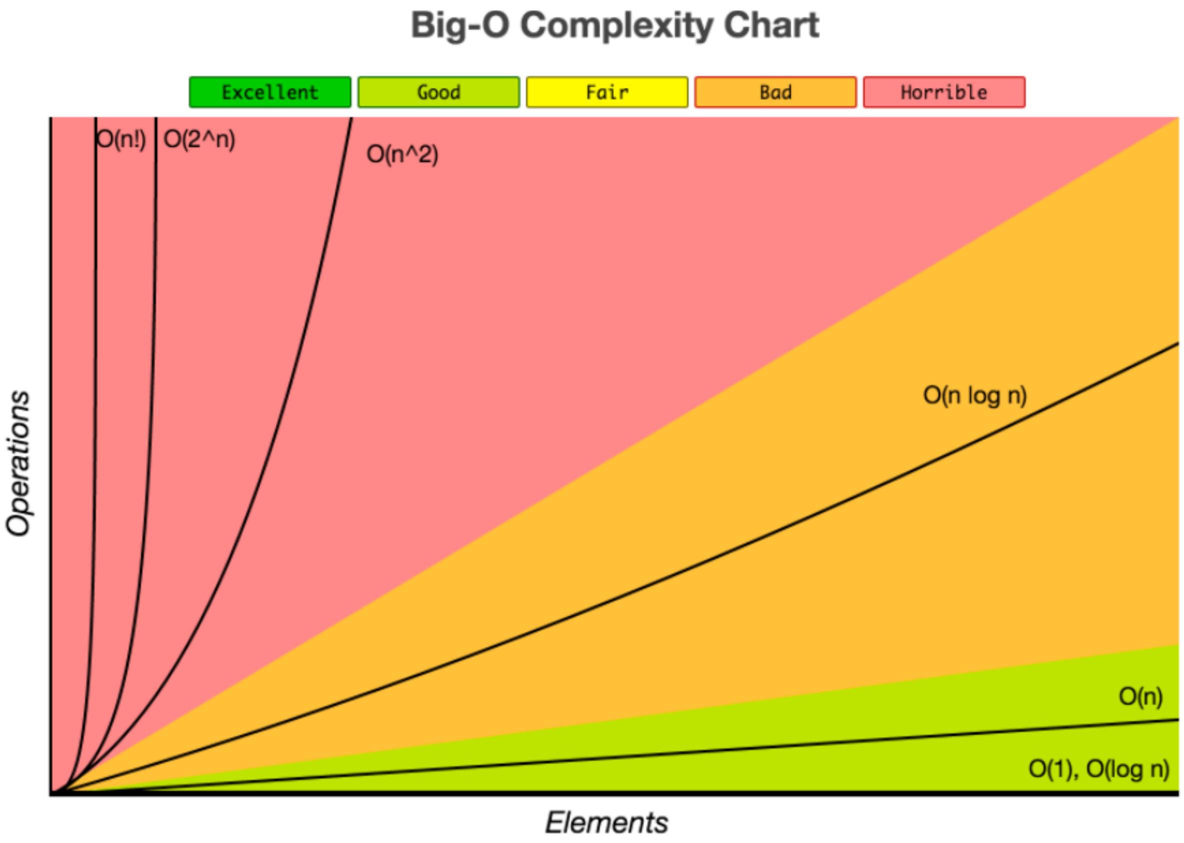
Big O notation is used to express the time complexity of an algorithm. Notice the time complexity for insertion sort is n^2 for the average and worst case scenarios; for merge sort the time complexity is n(log (n)) for all cases.



Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

What does this mean? Well, look at the graph below. You can see that a polynomial time where n is raised to at least a power of 2 seems pretty bad. And it is. But in computer science it is generally stated that anything that can be solved in polynomial time or less is considered to be solved in a “reasonable time”. That leaves algorithms that are solved in exponential or factorial time as not being solved in reasonable time.



Why discuss this now? Because n can be thought of as the number of records in the data set. As n gets really big the question becomes how long will it take for your algorithm to process the data?

22.(For everyone): Looking at the Array Sorting Algorithms chart above, which two sort methods are the best in all cases?
O(1) and O(log n)

23. (Optional, but required for those taking the AP Exam) For each big O notation below, describe whether it is run in reasonable time or not:

a) O(3n) yes

b) O(log (n)) yes

c) O(n (log n)) yes

d) O(2^n) no

e) O(n!) no

f) O(n^8) yes

g) O(10,000,000) yes