CS585 Final

Spring 2018: 5/3/18
Duration: 1 hour

Instructions/notes

the exam is closed books/notes/devices/neighbors, and open mind :)
there are 10 questions, plus a bonus
there are no ‘trick’ questions
please do NOT cheat; you get a 0 if you are found to have cheated
when time is up, stop your work; you get a 0 if you continue
good luck, hope you do well!

Q	Your score	Max possible score
1		2
2		4
3		4
4		4
5		2
6		3
7		2
8		4
9		5
10		5
Bonus		1
Total		**36**

Q1 (1+1=2 points). The usual ‘MapReduce’ steps are mapping, shuffling and reducing. But sometimes, an extra ‘combining’ step is inserted.

a. where does this occur (in the sequence of steps)?

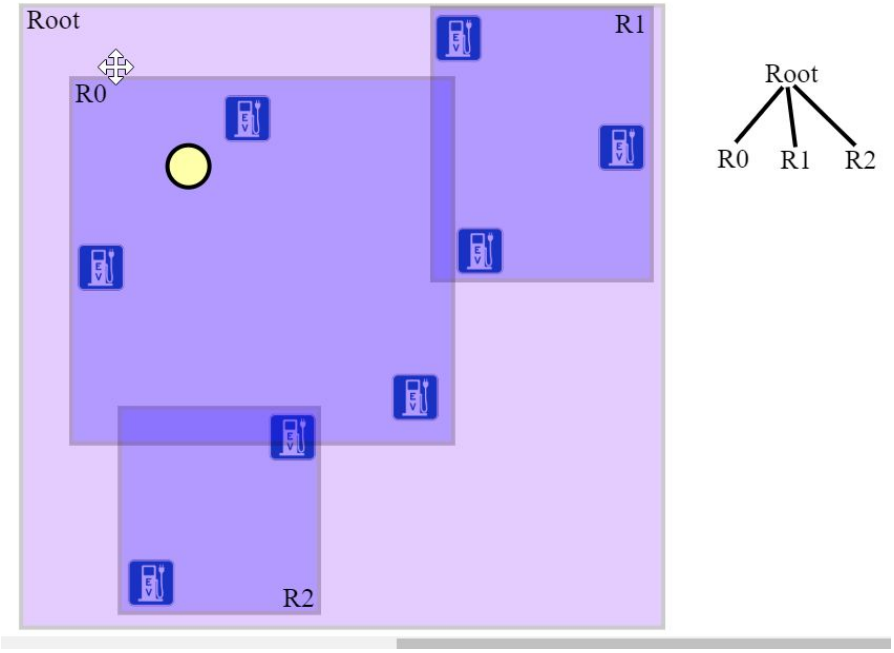
A. The combining step occurs at the end of a mapping stage.
1 point

b. what is its purpose (why include it)?

A. Combining helps aggregate values that belong to identical keys; this enhances efficiency by reducing network traffic during shuffling, since fewer keys need to get forwarded to reducers.
1 point

Q2 (2+2=4 points). You are running low on battery, driving around in an unfamiliar neighborhood - you need to recharge soon. You ask your navigation software for the nearest electric charging station, it instantly obliges - an R-tree helped, behind the scenes. Draw a simple map that shows your location, a scattering of 8 charging stations in the surrounding area, and an R-tree to contain the 8+1=9 locations; explain how the R-tree helps in quick retrieval of the result.

A. Our map and R-tree could look as follows:



2 points for the tree.
deduct 0.5 if missing root.
deduct 0.5 if missing if missing user location
deduct 0.5 if any gas station is missing. If more than 1, deduct 0.5 only
deduct 1.5 if not showing any rectangle over the locations

Our location is indicated by the yellow circle. Starting at the root of the R-tree, we identify which child node our location is in - for us this will be R0. Then we retrieve the three charging stations within R0, compute the closest one of the three. We don't need to consider the five stations that are in R1 and R2 - that is how we speed up the lookup (by processing only 3 locations, instead of 8).

1 point for saying process less location
1 point for describing how it processes less locations

Q3 (4 points). In standard 'basket analysis', we search for rules of the type A->B, where we specify 'support' threshold for A and a confidence for the ->. In other words, what are the (A,B) pairs such that given that A occurs, B occurs as well.

In 'differential basket analysis', we can take this a big step further - we can compare the occurrence of (A,B) pairs, given other (unrelated to (A,B)) factors. This can help make better use of the (A,B) associations. Eg. does (A,B) only occur (or NOT occur) in a certain store or groups of them?

What other factors (than store locations) can you think of? You can assume (bet!) that vast amount of customer data is available. List two factors related to customers, and two factors not related to customers.

1. Time of the year

2. Time of the day

3. Customer's gender

4. Customer's income

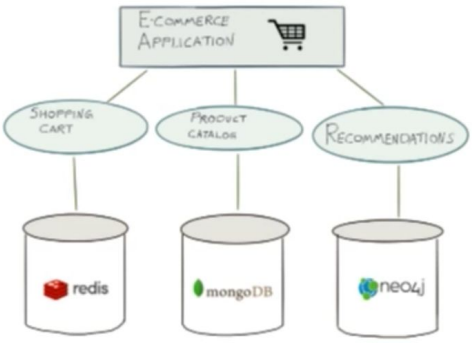
1 point each. There can be other correct factors as well.
2 points for customer related factors
2 points for non-customer related factors

Q4 (4 points). Before NoSQL, almost all databases were based on the relational model: tables (entities), and PK/FK relationships between them. The NoSQL paradigm offers us alternatives. There is one specific 'freedom' that NoSQL offers, from an architectural standpoint (ie. in the design, and redesign, of a large, complex application). What is it? Be specific, and describe it in a paragraph or two, with an illustration.

A. We have the freedom to implement a mixed-model architecture for our data storage, ie. we can have 'polyglot persistence'. With polyglot persistence, we don't need to decide on a single type of NoSQL database for the entire application; instead, based on the data to store, we can pick the appropriate type (k-v, column, document or graph).

1.5 point for polyglot persistence , or mixed-model architecture or similar meaning, such as multiple data storage technologies.
1.5 point for description
1 point for illustration that show multiple data stores (polyglot persistence)
Special cases:
if mention schema-less give 1.5 points
if a student mentions the (4) different types of DBs as the answer, they can get 2.5 points
[example from neo4j.com]

Polyglot Persistence



	Functionality	Database type
Shopping Cart	Rapid session reads / writes	Key-value store
Orders / Product Catalog	Frequent reads	Document
Customer social graph	Recommendation	Graph

Q5 (0.5*4=2 points). A highly effective, powerful and conceptually simple way to analyze data, is to employ ‘dataflow’. Name two uses of dataflow that you learned from the course (we covered three).

a. Pig Latin, for MapReduce tasks

b. TensorFlow, for neural network architectures
0.5 point for each
Another possible correct answer:
Weka, machine learning algorithms for data mining tasks

Data can also be processed in multiple stages (pipelining). Name two architectures we covered, that permit such multistage data analysis.

a. YARN, ie. MapReduce v2

b. BSP
0.5 point for each

Q6 (3 points). What is the most flexible way to model ('any') data? And, what structure can be used to do so?

A. A graph DB offers the most flexible way, via nodes and edges to model data and relationships.

Specifically, a 'triple store' ie. (subject,predicate,object), can be used.

2 points for graph DB.
1 point for structures: node, edges, etc.

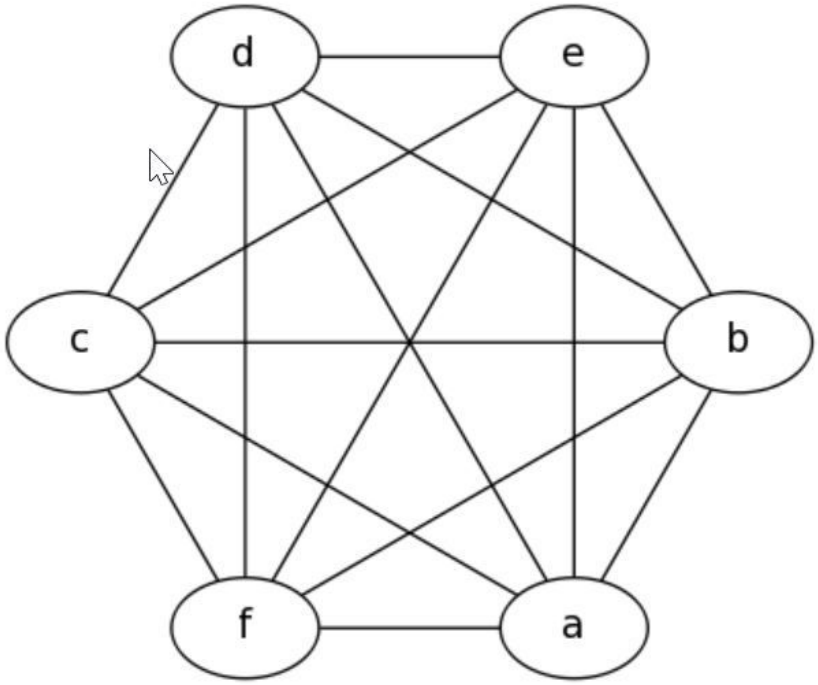
Q7 (2 points). Functional programming is an expressive, compact way of specifying data processing operations. You have seen two examples in the course - what are they?

a. MapReduce operations, via map(), reduce()

b. Graph processing, via Tinkerpop/Gremlin

1 point for each
deduce 0.5 if missing operations/tools for each one

Q8 (2+2=4 points). Represent the following graph in (valid!) JSON notation [suitable for storing in a text file, and reading it back to construct the graph] - do it two different ways!



One way:

```
{
  "graphData": {
    "vertices":
["a","b","c","d","e","f"],
   "edges":[
      [a,b], [b,c], [c,d], [d,e], [e,f], [a,f], [a,c], [a,d], [a,e], [b,d], [b,e], [b,f], [c,e],
      [c,f], [d,f]
    ]
  }
}
```

Another way (no need for the verts list!):

```
{
  "graphData": {
    "edges":[
       [a,b], [b,c], [c,d], [d,e], [e,f], [a,f], [a,c], [a,d], [a,e], [b,d], [b,e], [b,f], [c,e],
       [c,f], [d,f]
     ]
  }
}
```

2 points for each
deduct 0.25 if missing one edge
deduct 0.5 if missing more than 1 edge
deduct 1 points if the JSON is not valid for each

Q9 (0.5*10=5 points). An autonomous car is deployed on the streets, having its neural networks trained (using tens of thousands of hours of hand-labeled traffic videos). 'Autonomous warfare' is a scenario that is increased being discussed by analysts, where in a (distant?!) future, self-guided soldier 'bots' would conduct war operations against 'live' enemies (who presumably don't have the means to deploy automated soldiers). Our auto-guided soldier bots could operate on land (with wheels or limbs), in the water, and be airborne as well ('killer drones').

What would the required training data be? Think broadly (including multi-sensory modes!), and name 10 different classes (detection targets, ie. things to learn to recognize) that would be useful. In other words, what can we teach our bots' neural nets?

1. Look of the enemy (eg. head dress)
2. Landmarks on the enemy terrain (eg. ridges, rivers, bridges)
3. Look of enemy ships
4. Look of enemy aircraft
5. Look of enemy fuel depots
6. Look of enemy missile launchers
7. Faces of high-value targets
8. 'Smell' (chemical signature) of explosives
9. Land features indicating IEDs and landmines
10. Look etc. of 'friendlies'
0.5 point for each. There can be more correct answers

Q10 (5 points). Many events/phenomena in our lives have a spatial dependence - eg. if you live under the flight path of airplanes (eg. in Westchester, near LAX), your car would need to get cleaned quite often because of black, oily jet fuel getting deposited on it by landing planes! To prove such a causal link (landing airplanes cause oily buildups), you'd need to plot on a map: flight paths, and amount of soot (for example) per week found on cars near and away from the flight path. Cars closer to, and on the flight path, would show higher levels. Given the following list of causes and effects in no particular order, pick out 5 pairs of (cause,effect) relations that can be investigated for possible occurrence (or merely shown/documented), by plotting data on a map. Eg. the above example would be listed as (flight path, oily deposit). Note - your pairings need to be plausible, not frivolous.

Here is the list: flight path, freeways, particulate matter, power lines, noise, cancer, known gang locations, oily deposit, cellphone towers, wealthy neighborhoods, graffiti, BMW dealerships, low-income neighborhoods, crime, fast-food restaurants, vibrations (rattling), banks, pawn shops.

Among other things, such visualizations can be used to effect public policy, document social disparities, etc.

a. (freeways, crime)

b. (low-income neighborhoods, fast-food restaurants)

c. (wealthy neighborhoods, banks)

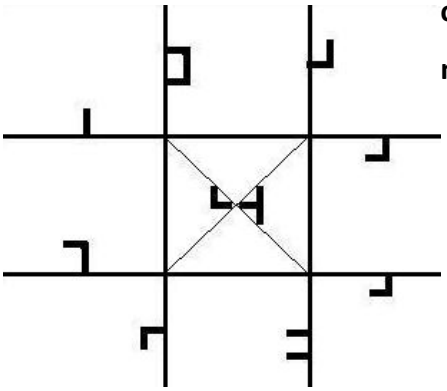
d. (wealthy neighborhoods, BMW dealerships)

e. (power lines, cancer)

1 point for each. There can be other correct answers. Give credit as long as it is from the list

Bonus (1 point). What is being indicated below?

A. Starting from the top-left and going counter-clockwise, they indicate 1,2,3,4,5,6,7,8,9 [with 9 at the center], as expressed using 7-segment LED displays



oriented appropriately, with each letter's lower-half missing!

1 point if mention numbers: 1,2,3,4,5,6,7,8,9 .