University of California, Los Angeles
Department of Computer Science

Computer Science 143 **Prof. Ryan Rosario**

Homework 3

Due Thursday, May 16, 2019 11:59pm ***via CCLE***

Please remember the following:

1. Homework is mostly graded on completion. We may grade a few parts, but it will never be the majority of the grade on the assignment. So try your best, and focus on solving the problems. Consider homework (and studying the solutions) as practice for the final exam.
2. Homework must be submitted digitally, on CCLE. We will not do any paper grading. You can use a text file, but if you use Word, a PDF is preferred rather than a DOC file.
3. If there are any exercises that are difficult to do digitally (such as diagrams or math), consider scanning your drawing or math, or using a graphics program (even a readable MS Paint is fine) or Equation Editor.
4. **For the sanity of the grader** we will ask you to run the queries and submit the result. You may lose points if you only provide a query.
5. Solutions will be posted.

Part 1: Go Long or Go Wide?

The website `keyvalues.com` allows startups and some large companies to discuss their company culture, something that some of you will find to be much more important than a huge paycheck. The purpose of this site is to provide candidates and other interested parties an in depth look at the company culture on a variety of dimensions. The content tends to discuss cultural values that are often overlooked in interviews. The table below shows some of them:

Team Values	Personal Health	Daily Routines	Engineering	Career Growth
Engages with Community	Work/Life Balance	Eats Lunch Together	Open Source Contributor	Promotes from Within
Team is Diverse	Ideal for Parents	Light Meetings	Start-to-Finish Ownership	Good for Junior Devs
Risk Taking > Stability	Fosters Psychological Safety	Thoughtful Office Layout	Fast-Paced	High Retention

Strategy	Company Properties
Engineering-Driven	Remote Work OK
Data-Driven	
Rapidly Growing Team	

Exercise

Help your professor (please)! (Though I've already done this) Take a look at the data. Write a query that creates a 0/1 matrix from this data. The rows should correspond to companies, the columns should correspond to cultural values. The value of each cell should be a 0 or 1 – a 1 if the value is associated with the company, 0 otherwise.

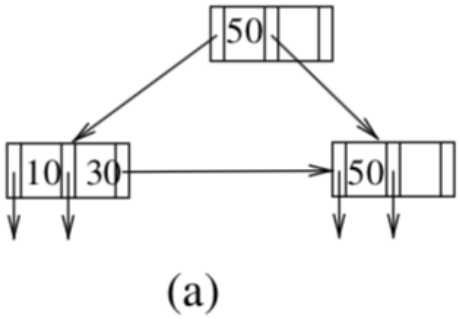
The query will be tedious, but in the "lots of copy/paste" way. You may want to write a script to generate the query (like in Project 1). Some of you may find a much better way to do this that does not require that. I do not know if it is possible, so if you are adventurous, you may want to research it.

To grade this problem, please submit your query, the number of rows in the output, and the number of columns in the output. Please do not submit the matrix.

With this 0/1 matrix, we can then create visualizations of which companies are similar to others, and which values are similar to others using singular value decomposition, principal component analysis, or multiple correspondence analysis. We won't do that for this class, but if you are interested, have some fun with the data.

Part 2: Seeing the Forest for the Trees

Consider the following B+tree for part (a). You may need to review chapter 14, or the lecture slides.



(a)

(a) Show the final B+tree structure after we insert 60, 20, and 80, in this order.

For (b) through (d), suppose there is a relation $r(A, B, C)$, with a B+-tree index with search key (A, B).

(b) What is the worst-case cost of finding records satisfying $10 < A < 50$ using this index, in terms of the number of records retrieved n_1 and the height h of the tree?

(c) What is the worst-case cost of finding records satisfying $10 < A < 50 \wedge 5 < B < 10$ using this index, in terms of the number of records n_2 that satisfy this selection, as well as n_1 and h defined above?

(d) Under what conditions on n_1 and n_2 would the index be an efficient way of finding records satisfying $10 < A < 50 \wedge 5 < B < 10$?

Part 3: If you Like it Then you Shoulda Put an Index on It

(a) Why is a hash structure not the best choice for a search key on which range queries are likely?

A range query cannot be answered efficiently using a hash index, we will have to read all the buckets. This is because key values in the range do not occupy consecutive locations in the buckets, they are distributed uniformly and randomly throughout all the buckets.

(b) Our description of static hashing assumes that a large contiguous stretch of disk blocks can be allocated to a static hash table. Suppose you can allocate only C contiguous blocks. Suggest how to implement the hash table, if it can be much larger than C blocks. Access to a block should still be efficient.