Carnegie Mellon University
Department of Computer Science
15-826 Multimedia and Data Mining
C. Faloutsos, Fall 2025

Homework 1
Due: **hard copy, AND e-copy, in class, at 2:00pm, on 09/05/2025**

VERY IMPORTANT - check-list:

1. Upload on **canvas**:
   - your answers (pdf or ascii), and
   - a separate file `queries.txt` with all your SQL queries, ready to run (`sqlite3 phonecalls.db < queries.txt` should give the correct results).
2. Just for this time, please also hand-in a **hard copy** of your answers and code, in class. For ease of grading, please **type** the full info on each page:
   - your *name* and *Andrew ID*,
   - *Course#* and *Homework#*.

Reminders:

- *Plagiarism*: Homework is to be completed *individually*.
- *ChatBots*: You may use chatGPT etc, but you are responsible for fixing their *hallucinations*.
- *Late homeworks*: please follow the announced policy (`www/.../826.F25`).

For your information:

- Graded out of **100** points; **2** questions total
- Rough time estimate: *2-6 hours*
- Weight: 1% of course grade.

Revision : 2025/08/29 00:13

Question	Points	Score
B-trees	10	
SQL	90	
Total:	100	

Question 1: B-trees . . . . . [10 points]

Consider B-trees of order d=2 (2*d+1 = 5 = maximum fanout). One such tree is in Figure 1.

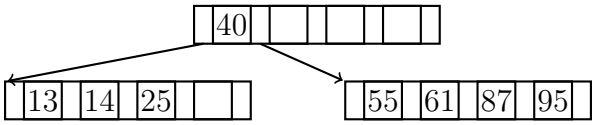

Figure 1: A B-tree of order d=2, with n=3 nodes, and height h=2.

NO NEED to justify your answers.

(a) **[4 points]** In a B-tree of order d=2 and height h=1 (ie., root-node only), what is (i) the *minimum*, and (ii) the *maximum* number of keys it can hold?

(a) _____

(b) **[6 points]** In a B-tree of order d=2 and height h=2 (i.e., like the one in Figure 1), what is the minimum number of keys that it can hold?

(b) _____

Question 2: SQL . . . . . [90 points]

For this part, we will use `sqlite3` (version 3.45.1), which is available on the andrew unix machines (needs vpn; then: `ssh unix.andrew.cmu.edu`). Feel free to use a different version of sqlite3 on some other machine, as long as your queries work correctly on `unix.andrew`.

Set up

1. Download the database file with the patent citation graph from `https://www.cs.cmu.edu/~christos/courses/826-resources/DATA-SETS-HOMEWORKS/phonecall-synthetic/phonecalls.db`
2. At the unix/linux prompt, open the database with the following command:

   ```
   sqlite3 phonecalls.db
   ```

   which should bring you the `sqlite>` prompt.

Optional: sanity checks

1. the command

   ```
   sqlite> .schema PhoneCalls
   ```

   should give:

```
CREATE TABLE PhoneCalls(
    source TEXT,
    destination TEXT,
    duration INTEGER);
```

2. Check the count of rows - the command:

   ```
   select count(*) from PhoneCalls;
   ```

   should give

   ```
   12
   ```

   (= total number of rows)

Data description: The `phonecalls.db` database has one table `PhoneCalls`, listing who calls whom, and for how long. For example the following row in the table means that customer 'Alice' called customer 'Bob' for 10 minutes.

```
source   destination  duration
-------  -----------  ----------
Alice    Bob          10
```

SQL Hints:

- Hint #1: Use `.headers on` and `.mode column` for easier debugging.
- Hint #2: For duplicate elimination: use `distinct`.
- Hint #3: `limit 5` will give the first 5 rows of the response.

FYI: Rationale: The queries in this exercise will be very useful for the upcoming project (analysis of a million-scale who-calls-whom network). In general, grouping, sorting, and spotting of 'heavy hitters' are vital for several data mining tasks like information summarization and anomaly detection.

Queries, and what to hand in: For all the queries below, hand in

- on canvas: e-copy (ascii) of all your sql code, in a single file `queries.txt`, ready to run (`sqlite3 phonecall.db < queries.txt`)
- on canvas: e-copy of your answers (output of your `queries.txt`).
- hard copies of the two above items.

(a) [**10 points**] **Warm-up: Alice's out-calls:** Find how many phonecalls 'Alice' has initiated.

.....

.....

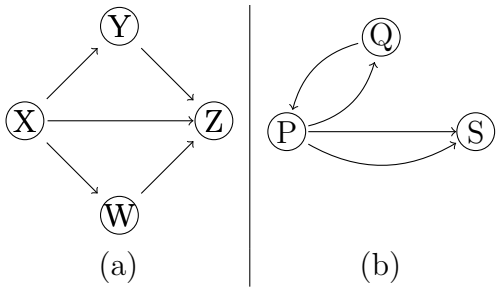

Figure 2: Two tiny examples of who-calls-whom.

(b) **[10 points] Heavy-hitters: Count of out-calls:** For each customer, print the name and the count of phone calls he/she has done, sorted in decreasing count order. In case of tie, sort by increasing alpha order of the source (i.e., "(Peter, 3) (Zoe,3)"). Show only the first k=2 such customers (use the keyword `limit`).

(FYI - Relationship to project & data mining: High-activity callers are important: either they are the best-paying customers of the phone company, or they are some sort of fraudsters - either way, worth listing them. Similarly, in social networks, high-degree nodes are interesting ('hubs', or 'super-spreaders'))

(c) **[10 points] Most-talkative: Out-duration:** As in the previous question, but using the total phonecall duration. That is, for each customer, print the name and total out-call duration; again, sort by duration (highest first); break ties (if any) by source in alpha order. Show all results (ie., no `limit`).

(FYI - Relationship to project & data mining: High-duration customers in a real phone network, are also worth inspecting, especially if they have very small or very high out-degree.)

(d) **[20 points] Unique out-friends:** For each customer, print the name and the count of **distinct** destinations; as before, sort (highest count first); break ties by source in alpha order. Again, report all customers (no `limit`). Consider the keyword `distinct`.

(FYI - Relationship to project & data mining: This measure could help us spot anomalies, like telemarketers and denial-of-service (DoS) attacks: if a high-activity node 'X' has too few out-friends, it might be sign of DoS; conversely, if too many out-friends, it could be a sign of a telemarketer or scammer.)

(e) **[40 points] Influencers: 2-step out-friends:** For each customer, print the name and the count of unique, 2-forward-step-away customers. Eliminate self-loops (eg., ('Alice', 'Alice')). Sort in the usual way (highest count first, break ties by source in alpha order).

There are some subtle issues with this query - thus we give a few examples:

Example #1: In Figure 2(a), there is only one qualifying pair: (X, Z), and thus the answer should be (X,1).

- Notice that X can reach Z through multiple paths - but we eliminate duplicates.
- Also notice that X can reach Z directly - in this exercise, this is fine; in a real setting, we may want to eliminate the one-step-away neighbors.

Example #2: In Figure 2(a), the only qualifying pair is (Q,S), thus the answer should be (Q,1).

- Notice that we ignore the path (P-Q-P), since it leads to a self-loop.
- Also notice that there *exist multi-edges*: P calls S twice, and thus Q can reach S through two different paths - but, again, we eliminate duplicates.

(FYI - Relationship to project & data mining: The count of 2-step-away neighbors is also important in phonecall and general network analysis: high such number means that the source node could quickly spread news / rumors / trends / viruses.)