

CARNEGIE MELLON UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
15-826 MULTIMEDIA DATABASES AND DATA MINING
C. FALOUTSOS, SPRING 2017

Homework 1

Due: hard copy, in class, at 3:00pm, on 1/30/2017

VERY IMPORTANT:

- This homework is on prerequisite, elementary material. People who score *80 or lower*, should consider dropping the course or switching to audit.
- Deposit **hard copy** of your answers, in class. For ease of grading, please
 1. **Separate** your answers, on different page(s) for each question (staple additional pages, if needed).
 2. **Type** the full info on **each** page: your **name**, **Andrew ID**, **course#**, **Homework#**, **Question#** on each of the pages.

Reminders:

- *Plagiarism*: Homework is to be completed *individually*.
- *Typeset* all of your answers whenever possible. Illegible handwriting may get zero points, at the discretion of the graders.
- *Late homeworks*: in that case, please email it
 - to all TAs
 - with the subject line exactly 15-826 Homework Submission (HW 1)
 - and the count of slip-days you are using.

For your information:

- Graded out of **100** points; **3** questions total
- Rough time estimate: *2-6 hours*

Revision : 2017/02/06 14:22

Question	Points	Score
B-trees	15	
Linear hashing	15	
SQL	70	
Total:	100	

Question 1: B-trees.....[15 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

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

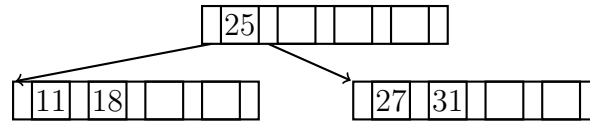


Figure 1: B-trees of order $d=2$.

- (a) **[5 points]** In an initially empty B-tree of order 2, insert the first 4 integers: 1,2,3,4; draw the resulting tree (hand-drawing is acceptable).

.....

.....

.....

.....

.....

- (b) **[5 points]** In an initially empty B-tree of order 2, insert the first 5 integers: 1,2,3,4,5; draw the resulting tree (again, hand-drawing is acceptable).

.....

.....

.....

.....

.....

- (c) **[5 points]** In the B-tree of Figure 1, delete key '25', and draw the resulting tree. If more than one solutions exist, draw them *all*. (Hand-drawing is acceptable).

.....

.....

.....

.....

.....

Question 2: Linear hashing [15 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

A hash table using linear hashing (with the traditional 1-to-2 split), started with the following hash function;

$$h_0(x) = x \text{ mod } 11$$

The buckets were numbered 0, 1, ..., 10.

- (a) **[6 points]** The table may grow and shrink. Give the 3 smallest table-sizes m ($m > 1$) that the split pointer is at bucket number 0.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

- (b) **[1 point]** How many hashing functions are active, when the hash table has $m=11^2=121$ buckets (numbered 0, 1, ..., 120)?

(b) _____

- (c) **[8 points]** List the active hashing function(s), for the above case ($m=121$ buckets).

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Question 3: SQL [70 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

For this part, we will use `sqlite3` (version 3.7.17), which is available on the andrew unix machines (`ssh unix.andrew.cmu.edu`).

Set up

1. Download the SQL database from
`http://www.cs.cmu.edu/~christos/courses/826-resources/DATA-SETS-HOMEWORKS/oscars-data/oscars.db`
2. and operate on it with the command
`sqlite3 oscar.db`
which should bring you the `sqlite>` prompt.

Data description: The `oscar.db` database has 3 tables, with information about movie academy awards ('Oscar'), including both nominations as well as wins. The tables and the meaning of their attributes are as follows:

- `people` (`personID`, `firstName`, `lastName`). The `personID` is a unique identifier for each actor/actress; the rest are self-explanatory.
- `nominations` (`nominationId`, `year`, `catId`, `personId`, `title`, `characterName`, `won`) Each row corresponds to a nomination of an actor/actress (`personID`), for a specific movie (`title`), for a given year.
 - `nominationId` is a unique identifier (1, 2, ...).
 - `won` is a boolean 't'/'f', depending on whether the nominated person won the award or not.
 - `characterName` is the name of the character in the movie.
- `categories` (`catId`, `category`) . This table has only 4 rows: '1' for 'Leading Actor', '2' for 'Supporting Actor', etc.

For example the 6th entry in `nominations` is shown in Table 1

<code>nominationId</code>	<code>year</code>	<code>catId</code>	<code>personId</code>	<code>title</code>	<code>characterName</code>	<code>won</code>
6	2009	1	2	Crazy Heart	Bad Blake	t

Table 1: 6th entry of the `nominations` table

It is the result of the query `select * from nominations where nominationId=6`, and it corresponds to the 2009 nomination of actor 'Jeff Bridges' (with `personID=2`), for the movie 'Crazy Heart', for leading actor (`catId=1`); he played the character 'Bad Blake', and he won (`'won'='t'`).

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

- hand-in both the SQL **code** of your answer, as well as the **output** of your code.
 - You may use **views**.
 - Please use **.headers** on for prettier output.
- (a) **[20 points] Top-winners:** List all the people that have won 3 or more times. Specifically, give the **personId**, first and last name, and count of wins. Order most-wins-first, and break ties by last name (ascending), and then by first name (also ascending).
- (FYI - Relationship to data mining: Grouping, sorting, and spotting of 'heavy hitters' are powerful, for data mining tasks like information summarization, and anomaly detection.)*
-
-

- (b) **[25 points] Duplicate detection:** Most movie titles are unique, except for a few re-makes. Find the re-makes - specifically, list the (common) title, the year of the first movie (**firstYear**), and the year of the second movie (**secondYear**). Sort by **firstYear**, and then by **secondYear**, both ascending.
- (FYI - Relationship to data mining: Spotting duplicates, exceptions, and rule-violations are typical tasks of data cleaning, which is usually the most time consuming step of data mining.)*
-
-

- (c) **[25 points] Top competitors :** Find pairs of people who compete too often against each other (and thus seem similar). Specifically, list the pairs of names that clashed 3 or more times, and the count **num_clashes** of times they clashed.
- We have a 'clash' when person-A and person-B are nominated in the same year, for the same category (for the same, or different movie). For each such pairs,
- print (**lastName1, firstName1, lastName2, firstName2, num_clashes**)
 - and sort by **num_clashes desc**, and then **lastName1** (ascending), and then by **firstName1**).
 - Within each pair, make sure that **lastName1 < lastName2**.

(FYI - Relationship to data mining: Such queries are useful in finding similar items, like similar actors here; similar genes/proteins in bioinformatics, near-duplicate tweets (possibly indicating plagiarism/fraud). Also, they are useful in link prediction and product recommendation, like, say Amazon: 'many people who bought product-X, also bought product-Y'.)

.....

.....