Homework 1 - Solutions

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

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-trees</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Linear hashing</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>SQL</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Question 1: B-trees.................................[15 points]

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

GRADED BY: Joey Fernau
Consider B-trees of order \( d=2 \) \((2d+1 = 5 = \text{maximum fanout})\). One such tree is in Figure 1.

![Figure 1: B-trees of order \( d=2 \).](image)

(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).

**Solution:** one node, keys 1,2,3,4

*Grading info: no partial credit, for any of the 3 sub-questions*

(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).

**Solution:** 3 nodes, root (3); left leaf (1,2), right leaf(4,5)

(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).

**Solution:** one solution only - one (root==leaf) node, with (11,18,27,31)

Homework 1 continues...
Question 2: Linear hashing  ................. [15 points]

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

**GRADED BY: Tanay Varma**

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

\[ h_0(x) = x \mod 11 \]

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.

**Solution:** 11, 22=11*2, 44=11*2^2

**Grading info:** -1 point, if they give 11*2, 11*4, 11*8

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

(b) 2

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

**Solution:** \( x \mod 88 \ (8*11=88) \), and \( x \mod 176 \ (16*11=176) \).

**Grading info:** 4pts per correct function

Homework 1 continues...
Question 3: SQL ........................................... [70 points]

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

GRADED BY: Mohak Nahta

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 oscars.db
   which should bring you the sqlite> prompt.

Data description: The oscars.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

<table>
<thead>
<tr>
<th>nominationId</th>
<th>year</th>
<th>catId</th>
<th>personId</th>
<th>title</th>
<th>characterName</th>
<th>won</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2009</td>
<td>1</td>
<td>2</td>
<td>Crazy Heart</td>
<td>Bad Blake</td>
<td>t</td>
</tr>
</tbody>
</table>

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. )

**Solution:** Code:

```sql
select P.personId, P.firstName, P.lastName, count(*) as numWins
from nominations as N, people as P
where N.personId = P.personId
  and won = 't'
group by P.personId
having numWins > 2
order by numWins desc, P.lastName, P.firstName;
```

**Grading info:**
- 10 pts for correct code
- full points for all correct alternatives (using 'views' is fine).
- -1 for each small error (wrong ordering, etc)
- no partial credit, if there are serious errors.

**Solution:** Output:

<table>
<thead>
<tr>
<th>personId</th>
<th>firstName</th>
<th>lastName</th>
<th>numWins</th>
</tr>
</thead>
<tbody>
<tr>
<td>534</td>
<td>Katharine</td>
<td>Hepburn</td>
<td>4</td>
</tr>
<tr>
<td>540</td>
<td>Ingrid</td>
<td>Bergman</td>
<td>3</td>
</tr>
<tr>
<td>420</td>
<td>Walter</td>
<td>Brennan</td>
<td>3</td>
</tr>
<tr>
<td>37</td>
<td>Jack</td>
<td>Nicholson</td>
<td>3</td>
</tr>
</tbody>
</table>

**Grading info:**
- 10 pts for correct answer
- -1 if the ordering is wrong
- no penalty if there are no column headers
- -1 if other small errors
- 1pt per correct tuple, if there are serious errors

(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.)

Solution: Code:

```sql
select distinct N1.title, N1.year as firstYear, 
   N2.year as secondYear 
from nominations as N1, nominations as N2 
where N1.title = N2.title 
   and N1.year < N2.year 
order by N1.year, N1.title;
```

Grading info: 15 points for code - all correct alternatives are acceptable

Grading info: -1 for each small error (eg., wrong ordering, duplicates, etc)

Grading info: no partial credit, if there are serious errors.

Solution: Output:

<table>
<thead>
<tr>
<th>title</th>
<th>firstYear</th>
<th>secondYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Letter</td>
<td>1928</td>
<td>1940</td>
</tr>
<tr>
<td>A Star Is Born</td>
<td>1937</td>
<td>1954</td>
</tr>
<tr>
<td>The Hurricane</td>
<td>1937</td>
<td>1999</td>
</tr>
<tr>
<td>Goodbye, Mr. Chips</td>
<td>1939</td>
<td>1969</td>
</tr>
<tr>
<td>Henry V</td>
<td>1946</td>
<td>1989</td>
</tr>
<tr>
<td>Cyrano de Bergerac</td>
<td>1950</td>
<td>1990</td>
</tr>
<tr>
<td>Moulin Rouge</td>
<td>1952</td>
<td>2001</td>
</tr>
<tr>
<td>True Grit</td>
<td>1969</td>
<td>2010</td>
</tr>
</tbody>
</table>

Grading info: 10 points for correct answer

Grading info: -1 for each small error (wrong ordering, etc)

Grading info: 1pt for each correct tuple, if there are serious other problems

(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.

Question 3 continues...
(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.’)

Solution: Code:

```sql
create view competitors as
    select N1.personId as pid1, N2.personId as pid2,
            count(*) as num_clashes
    from nominations as N1, nominations as N2
    where N1.personId <> N2.personId
        and N1.catId = N2.catId
        and N1.year = N2.year
    group by N1.personId, N2.personId;

select P1.lastName, P1.firstName,
        P2.lastName, P2.firstName, C.num_clashes
from people as P1, people as P2, competitors as C
where P1.personId = C.pid1
    and P2.personId = C.pid2
    and P1.lastName < P2.lastName
    and C.num_clashes > 2
order by C.num_clashes desc, P1.lastName, P1.firstName,
        P2.lastName, P2.firstName;
```

Grading info:
15 pts for correct answer - again, all correct alternatives, are fine.
-1 for small errors (eg., self pairs, mirror pairs, wrong ordering, etc)
no partial credit otherwise.

Solution: Output:

<table>
<thead>
<tr>
<th>lastName</th>
<th>firstName</th>
<th>lastName</th>
<th>firstName</th>
<th>num_clashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Davis</td>
<td>Bette</td>
<td>Garson</td>
<td>Greer</td>
<td>4</td>
</tr>
<tr>
<td>Davis</td>
<td>Bette</td>
<td>Hepburn</td>
<td>Katharine</td>
<td>4</td>
</tr>
<tr>
<td>Nicholson</td>
<td>Jack</td>
<td>Pacino</td>
<td>Al</td>
<td>4</td>
</tr>
<tr>
<td>Bergman</td>
<td>Ingrid</td>
<td>Garson</td>
<td>Greer</td>
<td>3</td>
</tr>
<tr>
<td>Boyer</td>
<td>Charles</td>
<td>Tracy</td>
<td>Spencer</td>
<td>3</td>
</tr>
<tr>
<td>Colbert</td>
<td>Claudette</td>
<td>Davis</td>
<td>Bette</td>
<td>3</td>
</tr>
<tr>
<td>Kerr</td>
<td>Deborah</td>
<td>Taylor</td>
<td>Elizabeth</td>
<td>3</td>
</tr>
<tr>
<td>Lemmon</td>
<td>Jack</td>
<td>O’Toole</td>
<td>Peter</td>
<td>3</td>
</tr>
<tr>
<td>Newman</td>
<td>Paul</td>
<td>Tracy</td>
<td>Spencer</td>
<td>3</td>
</tr>
<tr>
<td>Olivier</td>
<td>Laurence</td>
<td>Stewart</td>
<td>James</td>
<td>3</td>
</tr>
</tbody>
</table>

Question 3 continues...
Grading info: 10 pts for correct answer.

Grading info: -1 pt for each small error (ordering, duplicates etc).

Grading info: +0.5 pt for each correct tuple, if there are serious errors

End of Homework 1