Carnegie Mellon University
Department of Computer Science
15-826 Multimedia Databases and Data Mining
C. Faloutsos, Spring 2016

# Homework 1 - Solutions

Due: **hard copy, in class, at 3:00pm, on 1/25/2016**

**VERY IMPORTANT**:

- This homework is on prerequisite, elementary material. People who score *80 or lower*, will be encouraged to drop the course or switch 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* : 2016/02/02 18:15

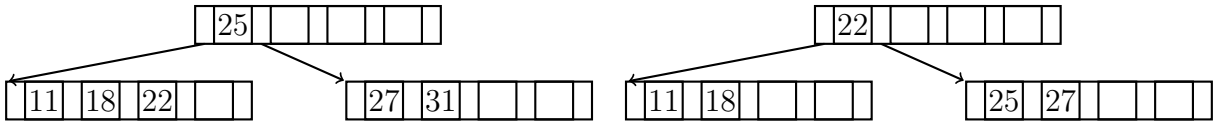
| Question | Points | Score |
|---|---|---|
| B-trees | 15 | |
| Linear Hashing | 15 | |
| SQL | 70 | |
| Total: | 100 | |

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

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

*Grading info: graded by Varshaa Naganathan*

Consider the B-trees of order $d$=2 ($2^*d+1 = 5$ = maximum fanout), as shown in Figure 1. We want to delete key '31'.




(a) original (b) your answer

Figure 1: B-tree example

(a) **[3 points]** Before the deletion, the B-tree has 3 nodes - how many will it have after the deletion of key '31'?

(a) **3**

(b) **[4 points]** Which keys (if any) will be in the same node as key '11'?

(b) **key '18'**

*Grading info: 2/4 if 11,18 are mentioned*

(c) **[8 points]** Draw the tree, after the deletion - use Figure 1(b); cross out ('×') any nodes that you don't need.

*Grading info: 5/8 if correct tree with multiple guesses(?)*

*Grading info: 2/8 if explanation attempted*

*Grading info: 6/8 if one key is off*

## Question 2: Linear Hashing.....[15 points]

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

*Grading info: graded by Varshaa Naganathan*

Consider a hash table that operates under linear hashing. When it started, the initial hashing function was $h(x) = x \bmod 4$; the hash table had $B$=4 buckets (0,1,2,3), and the split pointer was $s$=0. After some other splits, the table has $N$=64 buckets (numbered 0,1,..., 63)

(a) **[3 points]** Where is the split pointer $s$? Give the bucket number it points to (integer in the range 0 ... 63)

(a) **0**

(b) **[2 points]** How many hashing functions are active?

(b) **1**

(c) **[2 points]** Which one(s)?

**Solution:** $h(x) = x \bmod 64$

*Grading info: if partially correct ($h(x) = \bmod 64, h(x) = h(x) \bmod 64$ etc.), -1 pt*

(d) **[8 points]** Suppose the next step is a contraction - which cell will be merged with which one?

**Solution:** cell 63 will merge into cell 31

*Grading info: if they give one wrong cell, -4 pt*

*Grading info: if they give the correct pair of cells, but not which one will be the destination, then -2 pt*

## Question 3: SQL ..... [70 points]

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

*Grading info: graded by Wan-Shue Lai*

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 3rd entry in `nominations` is shown in Table 1

| nominationId | year | catId | personId | title | characterName | won |
|---|---|---|---|---|---|---|
| 3 | 2010 | 1 | 3 | The Social Network | Mark Zuckerberg | f |

Table 1: 3rd entry of the `nominations` table

It is the result of the query `select * from nominations where nominationId=3`, and it corresponds to the 2010 nomination of actor Jesse Eisenberg (with `personID` =3), for the movie 'The Social Network', for leading actor (`catId`=1); he played the character 'Mark Zuckerberg', and he did not win ('won'='f').

Question 3 continues...

**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. Use `.headers on` for prettier output.

(a) **[20 points] Super-stars:** List all the people that have been nominated 9 or more times. Specifically, give the `personId`, first and last name, and count of nominations. Order most-nominations-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:**

```
select P.personId, P.firstName, P.lastName,
    count(*) as numNominations
from nominations as N, people as P
where N.personId = P.personId
group by P.personId
    having numNominations > 8
order by numNominations desc, P.lastName, P.firstName;
```

**Solution:**

```
personId    firstName   lastName    numNominations
----------  ----------  ----------  --------------
450         Meryl       Streep      16
534         Katharine   Hepburn     12
37          Jack        Nicholson   12
578         Bette       Davis       11
62          Paul        Newman      9
104         Laurence    Olivier     9
129         Spencer     Tracy       9
```

(b) **[25 points] Competitors of 'Jesse Eisenberg'**: Find the people who competed against him, that is, they got nominated in the same year(s) and the same categorie(s) that he got nominated. For each such actor, print (`lastName, firstName, title, year, catId`), and sort by `lastName` (ascending), and then by `firstName`).

*(FYI - Relationship to data mining: Such queries are useful in link prediction and product recommendation, like, say Amazon: 'people who bought product-X, tend to also buy product-Y'.)*

**Solution:**

```
select P2.lastName, P2.firstName ,
    N2.title, N2.year, N2.catId
```

```
from nominations N1, nominations N2, people P1, people P2
where N1.catId = N2.catId
    and N1.year = N2.year
    and N1.personId = P1.personId
    and N2.personId = P2.personId
    and P1.firstName = 'Jesse'
    and P1.lastName = 'Eisenberg'
    and P1.personId <> P2.personId
order by P2.lastName ;
```

**Solution:**

| lastName | firstName | title | year | catId |
|---|---|---|---|---|
| Bardem | Javier | Biutiful | 2010 | 1 |
| Bridges | Jeff | True Grit | 2010 | 1 |
| Firth | Colin | The King's | 2010 | 1 |
| Franco | James | 127 Hours | 2010 | 1 |

(c) **[25 points] Duplicate detection**: Usually, there is only one winner for each year, and each category. Are there violations to that conjecture? List the exceptions, if any - for each exception, print `year`, `catId` and the count of winners (which should be $> 1$).

*(FYI - Relationship to data mining: Spotting exceptions and rule-violations are typical tasks of data cleaning, which is usually the most time consuming step of data mining. )*

**Solution:**

```
select N.year, N.catId, count(*) as winnerCount
from nominations N
where N.won = 't'
group by N.year, N.catId
    having winnerCount >1;
```

**Solution:**

| year | catId | winnerCount |
|---|---|---|
| 1931 | 1 | 2 |
| 1968 | 3 | 2 |