COMP SCI 4094/4194/7094 - Distributed Databases and Data Mining
Assignment 1

DUE: 23:30 Monday, 16th April, 2018

Important Notes

- Handins:
  - The deadline for submission of your assignment is **23:30 Monday, 16th Apr, 2018**.
  - You must do this assignment individually and make individual submissions.
  - Your program should be coded in **C++** and pass test runs on the six test files. The sample input and output files are downloadable from the track of Assignment 1 at MyUni (`https://myuni.adelaide.edu.au/courses/36285/assignments/64954`).
  - You need to use `svn` to upload and run your source code in the web submission system following Web-submission instructions stated at the end of this sheet. You should attach your name and student number in your submission.
  - Late submissions will attract a penalty: the maximum mark you can obtain will be reduced by 25% per day (or part thereof) past the due date.
- Marking scheme:
  - 12 marks for testing on 6 standard tests: 2 marks per test.
  - 3 mark for code readability
  - **Note:** If it is found your code did not implement the required computation tasks in this assignment, you will receive zero mark regardless of the correctness of test output.

If you have any questions, please send them to the student discussion forum. This way you can all help each other and everyone gets to see the answers.

The assignment

Suppose you are a database programmer of an outsourcing company. Your supervisor asked you to fragment a students enrolment database of all public universities in South Australia horizontally by an algorithm you learned from the DDDM lecture.

To implement the task, you are give three text files: a student enrolment database, a set of applications/queries, and a set of the simple predicates. The format of these files can be found in the following section. Your code should follow the framework of Algorithm 3.2 on page 90 of the text and produce a set (M) of complete and minimal minterm predicates as the output by which all the fragments are non-empty. That is, if a minterm predicate generates an empty fragment on the given applications, it should be removed from M.

Example

Input

- Student Enrolment Database:

Timestamp	University	Program	TuitionFee	Enrolment
T1	Flinders	PhD	20	13
T2	Flinders	PhD	20	1
T3	UofA	MCS	30	14
T4	Flinders	MSE	32	13
T5	UniSA	BSE	25	16
T6	UniSA	MSE	31	14
T7	Flinders	MCS	29	22
T8	UofA	MSE	31	2
T9	UniSA	MCS	32	17
T10	UofA	BCS	18	10

  Note:

  i. In the database, each line (except the first line) follows the order of Timestamp → University → Program → TuitionFee → Enrolment, split by Tab.

  ii. In the dimension of University, there are 3 keys: UofA, UniSA, Flinders.

  iii. In the dimension of Program, there are 5 keys: PhD, MCS, MSE, BCS, BSE.

  iv. In the dimensions of TuitionFee ($000's) and Enrolment, all the values are positive integers.

  v. In each single test, the auto-marker script will generate a random database with an average of 55 records.

- Set of applications/queries:
  - UofA PROGRAM-ALL TUITIONFEE-? ENROLMENT-ALL
    (Check the tuition fee of UofA)
  - UNIVERSITY-? PROGRAM-ALL TUITIONFEE-<20 ENROLMENT-ALL
    (Find the universities containing programs whose tuition fees are less than $20k)
  - UNIVERSITY-ALL PROGRAM-? TUITIONFEE->=15 ENROLMENT-<10
    (Find the programs whose tuition fees are greater or equal to $15k, enrolments are smaller than $10k)

  Note:

  i. The attributes in an application are split by Tab.

  ii. The applications are listed line by line.

  iii. In each single test, the set of applications are randomly generated.

- Simple predicates set:
  - UNIVERSITY=UofA UNIVERSITY=UniSA UNIVERSITY=Flinders UNIVERSITY=Torrens
  - PROGRAM=PhD PROGRAM=MSE PROGRAM=MCS PROGRAM=BSE PROGRAM=BCS PROGRAM=MA PROGRAM=MCI

  - TUITIONFEE<20 TUITIONFEE>=20 TUITIONFEE>=15 TUITIONFEE<15
  - ENROLMENT>=15 ENROLMENT<15 ENROLMENT<10 ENROLMENT>=10

  Note:

  i. The simple predicates belongs to one attribute are listed in one line.
  ii. The simple predicates follows the order of their attributes order in the database.
  iii. In each line, the simple predicates are split by Tab.

Output

- Minterm predicates set:
  - UNIVERSITY=UofA TUITIONFEE>=15 TUITIONFEE< 20 ENROLMENT>=10
  - UNIVERSITY=UofA TUITIONFEE>=20 ENROLMENT<10
  - UNIVERSITY=UofA TUITIONFEE>=20 ENROLMENT>=10
  - UNIVERSITY!=UofA TUITIONFEE>=20 ENROLMENT<10
  - UNIVERSITY!=UofA TUITIONFEE>=20 ENROLMENT>=10

  Note:

  i. Each minterm predicate/fragment takes one line.
  ii. Attributes in each line should be listed in the order of university, program, tuition fee, enrolment.
  iii. Values of attributes UNIVERSITY and PROGRAM should be listed following the same order as in Note ii and Note iii of Student Enrolment Database (Input).
  iv. Values of attributes TUITIONFEE and ENROLMENT should be listed in increasing order of their numeric values.

- Hint:
  - When constructing the minterm predicates set, you should remove all contradictions by considering the properties of the discrete attributes (only one TRUE value) and continuous attributes (disjoint intervals). E.g., we cannot have a minterm predicate contains both UNIVERSITY=UofA and UNIVERSITY=UniSA; the conjunction of simple predicates TUITIONFEE<15 and TUITIONFEE<20 yields disjoint intervals TUITIONFEE<15, 15 <=TUITIONFEE<20 and TUITIONFEE>=20, rather than four intervals resulted by considering each predicate separately.
  - The minterm predicates set produced by Algorithm 3.2 of the text that does not ensure non-empty fragments on the given input applications would be:
    - UNIVERSITY=UofA TUITIONFEE<15 ENROLMENT<10
    - UNIVERSITY=UofA TUITIONFEE<15 ENROLMENT>=10
    - UNIVERSITY=UofA TUITIONFEE>=15 TUITIONFEE< 20 ENROLMENT<10
    - UNIVERSITY=UofA TUITIONFEE>=15 TUITIONFEE< 20 ENROLMENT>=10
    - UNIVERSITY=UofA TUITIONFEE>=20 ENROLMENT<10
    - UNIVERSITY=UofA TUITIONFEE>=20 ENROLMENT>=10
    - UNIVERSITY!=UofA TUITIONFEE<15 ENROLMENT<10
    - UNIVERSITY!=UofA TUITIONFEE<15 ENROLMENT>=10
    - UNIVERSITY!=UofA TUITIONFEE>=15 TUITIONFEE< 20 ENROLMENT<10
    - UNIVERSITY!=UofA TUITIONFEE>=15 TUITIONFEE< 20 ENROLMENT>=10
    - UNIVERSITY!=UofA TUITIONFEE>=20 ENROLMENT<10
    - UNIVERSITY!=UofA TUITIONFEE>=20 ENROLMENT>=10

Web-submission instructions

- First, type the following command, all on one line (replacing xxxxxxx with your student ID):

  ```
  svn mkdir --parents -m "DDDM"
  https://version-control.adelaide.edu.au/svn/axxxxxxx/2018/s1/dddm/assignment1
  ```

- Then, check out this directory and add your files:

  ```
  svn co https://version-control.adelaide.edu.au/svn/axxxxxxx/2018/s1/dddm/assignment1
  cd assignment1
  svn add PHF.cpp
  svn commit -m "assignment1 solution"
  ```

- Next, go to the web submission system at:
  `https://cs.adelaide.edu.au/services/websubmission/`
  Navigate to 2018, Semester 1, Distributed Databases and Data Mining, Assignment 1. Then, click Tab "Make Submission" for this assignment and indicate that you agree to the declaration. The automark script will then check whether your code compiles. You can make as many resubmissions as you like. If your final solution does not compile you will not get any marks for this solution.

- **Note:**

  i. The auto-marker script compiles and runs the cpp file in the name of "PHF.cpp".

  ii. The auto-marker script will compile your PHF.cpp by the following command:
  `g++ -std=c++11 PHF.cpp -o run`

  iii. Your PHF.cpp should accept three input text files in the order of Database, Queries and Simple predicates then print the required minterm predicates set as the output.

  iv. The file path and the file name in your local machine will not work with our web-submission system.