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

DUE: 23:59, Sunday, 13 September 2020

Important Notes

- Handins:
  - The deadline for submission of your assignment is **23:59, Sunday, 13 September 2020**.
  - You must do this assignment individually and make individual submissions.
  - Your program should be coded in **C++** and pass test runs on the two test files. The sample input and output files are downloadable in "Assignments" of MyUni (`https://myuni.adelaide.edu.au/courses/54718/assignments`).
  - 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 4 randomly generated tests: 3 marks per test, where 1 mark is for the affinity matrix AA, and 2 marks for the clustered affinity matrix CA.
  - 3 marks for the structure of your code.

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

In this assignment you are required to implement the Bond Energy Algorithm of vertical fragmentation. Your code should contains two separate procedures AA Generator and CA Generator, where AA Generator takes the input of all attributes of a relation, a set of queries and their access frequencies at different sites, and produces the output of an affinity matrix AA, and CA Generator takes input of an affinity matrix AA and produces a clustered affinity matrix CA. For description of the BEA algorithm, definitions of AA and CA, please see lecture slides/textbook.

In this assignment, the Attribute Affinity is measured by the extended Otsuka-Ochiai coefficient (https://en.wikipedia.org/wiki/Yanosuke_Otsuka) instead of the traditional method described in the textbook. The following equations show the details of the computation, where q is the number of queries, and m is the number of sites, A_{ik} is the number of times Attribute A_i is accessed by Query q_k, considering of all sites. For the result of division, you must round it up to the nearest integer.

$$aff(A_i, A_j) = \lceil \frac{\sum_{k=1}^{q} A_{ik} \times A_{jk}}{\sqrt{\sum_{k=1}^{q} A_{ik} \times \sum_{k=1}^{q} A_{jk}}} \rceil,$$

$$A_{ik} = use(q_k, A_i) \times \sum_{j=1}^{m} acc\_matrix(q_k, S_j).$$

Example

For AA Generator:

Input

- The relation, called PROJ, has the following features A_i:

Label	Name
A1	PNO
A2	PNAME
A3	BUDGET
A4	LOC

- Queries (qi):

q1: SELECT BUDGET FROM PROJ WHERE PNO=Value	**A1, A3**
q2: SELECT PNAME, BUDGET FROM PROJ	**A2,A3**
q3: SELECT PNAME FROM PROJ WHERE LOC=Value	**A2,A4**
q4: SELECT SUM(BUDGET) FROM PROJ WHERE LOC=Value	**A3,A4**

- Access frequency matrix ACC, where Si denotes the i-th site:

	S1	S2	S3
q1	15	20	10
q2	5	0	0
q3	25	25	25
q4	5	0	0

Output

- The attribute affinity matrix AA:

	A1	A2	A3	A4
A1	45	0	41	0
A2	0	71	1	71
A3	41	1	38	1
A4	0	71	1	71

For CA Generator:

Input

- The attribute affinity matrix AA:

	A1	A2	A3	A4
A1	45	0	41	0
A2	0	71	1	71
A3	41	1	38	1
A4	0	71	1	71

Output

- The attribute affinity matrix CA:

	A1	A3	A4	A2
A1	45	41	0	0
A3	41	38	1	1
A4	0	1	71	71
A2	0	1	71	71

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/2020/s2/dddm/assignment1
  ```

- Then, check out this directory and add your files:
  ```
  svn co https://version-control.adelaide.edu.au/svn/axxxxxxx/2020/s2/dddm/assignment1
  cd assignment1
  svn add AAGenerator.cpp
  svn add CAGenerator.cpp
  svn commit -m "assignment1 solution"
  ```

- Next, go to the web submission system at:
  `https://cs.adelaide.edu.au/services/websubmission/`
  Navigate to 2020, Semester 2, 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 two cpp files named "AAGenerator.cpp" and "CAGenerator.cpp" one by one.

  ii. The auto-marker script will compile your AAGenerator.cpp and CAGenerator.cpp by the following command:
  ```
  g++ -std=c++11 AAGenerator.cpp -o runAA
  g++ -std=c++11 CAGenerator.cpp -o runCA
  ```

  iii. Your AAGenerator.cpp should accept three input text files in the order of Attributes (att), Queries (query) and Access Frequencies (acc), which are randomly generated by the system, then output and print the required attribute affinity matrix (aa). Your CAGenerator.cpp should accept input affinity matrix (aa) provided by the system rather than reading your AAGenerator's output AA, then output and print the clustered affinity matrix (CA) as the output. In this way of testing AA and CA separately, your marks will be maximized — you will receive marks for your correct CAGenerator coding even if your AAGenerator produces incorrect AA.

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