Homework #2

As directed below, you need to submit your code to the designated place by the TAs.

Any form of cheating, lying or plagiarism will not be tolerated. Students can get zero scores and/or get negative scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

Both English and Traditional Chinese are allowed for writing any part of your homework (if the compiler recognizes Traditional Chinese, of course). We do not accept any other languages. As for coding, either C or C++ or a mixture of them is allowed.

2.1 More about Arrays

(1) (10%) Do Exercise R-3.3 of the textbook, using any pseudo code that the TAs can understand.

(2) (10%) A lower triangular matrix A is a matrix with $A[i][j] = 0$ whenever $i < j$. Describe how you can store A with a dense one-dimensional array without wasting space on the entries with value 0. You need to describe the memory layout and the function for getting/putting.

2.2 Sparse Matrix Processing

You are asked to design and implement a data structure to store a very big data set of KDDCUP2012 Track1. There are 73209277 lines in the file `/tmp2/KDDCUP2012/track1/rec_log_train.txt`, which is placed on every 217 workstation from linux1 to linux13. The format of each line is:

`(UserId)\t(ItemId)\t(Result)\t(Unix-timestamp)`

The data set is a log file of a recommendation system. Each line means that the system recommends (gives) an item `(ItemId)` to a user `(UserId)` at time `(Unix-timestamp)`. `(Result)` will be 1 if the user accepts the recommendation, and -1 otherwise. You can view the data set as a super big 3D sparse matrix M with M[UserId][ItemId][Unix-timestamp] = Result.

The purpose of the homework is to help you understand that designing data structures for *large data* is a non-trivial problem. We understand that you do not know many tools (yet). So please just try your best to come up with *something*. We also encourage you to be creative!

Your design should support the following actions:

- accept(*u,i,t*): outputs the `(Result)` when user u is given item i at time t
- items(*u1, u2*): outputs the sorted `(ItemId)`, line by line, that are recommended to both user *u1* and user *u2*
- users(*i1,i2,t1,t2*): outputs the sorted `(UserId)`, line by line, which corresponds to users who are given both item *i1* and item *i2* between the time interval [*t1,t2*]
- ratio(*i,threshold*): outputs the acceptance rate of item i among the users who have been given items for more than *threshold* times. The format of your output is (#accept)/(#total) .

  For instance, ratio(10, 78) = 14/101 means there are 101 users who are given items by the system for more than 78 times. Among them, 14 users (remove duplication) have accepted item 10.
- findtime_item(*i, Us*): outputs the sorted list of timestamps (remove duplication), line by line, that correspond to when item i is given to any member from a group of users *Us*. You can choose your implementation of *Us*.

The TAs will provide the desired input/output format online. You need to follow the formats so the TA can test your program with their own input files. You are allowed to use any standard libraries that you know how to use (for the questions below).

(1) (10%) Describe your design of the data structure. Emphasize on why you think the data structure would be (time-wise) efficient for the four desired actions.

(2) (10%) Compute how much time it takes to convert the raw data to your data structure. Be sure to describe the platform you are using. (The faster the better!)

(3) (60%) Implement the data structure you designed and write a demo program (with the input/output format) to show how your data structure performs the four desired actions. Then, briefly ($\leq$ 20 lines) state how you test whether your implementation is correct. Furthermore, write a Makefile to compile your codes (data structure and demo program). TAs will use *make* to compile your source code and use *make run* to run your demo program on 217 workstation.

Homework that cannot be compiled or run correctly on the 217 workstation will not be graded.