Homework #2

RELEASE DATE: 03/24/2015

DUE DATE: 04/07/2015, **16:20** in CSIE R102/R104 and on github

As directed below, you need to submit your code to the designated place on the course website.

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.

This homework set comes with 200 points and 20 bonus points. In general, every homework set of ours would come with a full credit of 200 points.

2.1 More about C++

(1) (10%) The `sub1` below may result in a run-time error. Why?

```
int& sub1(int& a, int& b){
  int c = a - b;
  return c;
}
```

(2) (Bonus 10%) The `sub2` below does not result in a run-time error, but there may be some other problem. What is the problem?

```
int& sub2(int& a, int& b){
  int* pc = new int;
  *pc = a - b;
  return (*pc);
}
```

2.2 Arrays, Linked List, and Recursion

(1) (10%) Do Exercise C-3.4 of the textbook. (The faster the better!)

(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 at the upper triangular part. You need to describe the memory layout and the function for getting/putting values from/to the matrix.

(3) (10%) Do Exercise C-3.22 of the textbook.

(4) (10%) Do Exercise C-3.18 of the textbook using either C/C++ or pseudo code.

(5) (Bonus 10%) Do Exercise C-3.18 of the textbook, but use **one single loop** instead of recursion.

2.3 Asymptotic Complexity

In all the questions below, the notation $f(n) = O(g(n))$ applies to functions $f(n)$ and $g(n)$ from $\mathbb{N}$ to $\mathbb{R}^+ \cup \{0\}$, and $f(n) = O(g(n))$ if and only if there exists $n_0 > 0$ and $c > 0$ such that $\forall n \geq n_0$, $f(n) \leq cg(n)$.

(1) (10%) Do Exercise R-4.24 of the textbook, under the assumption that both $d(n) - e(n) \geq 0$ and $f(n) - g(n) \geq 0$.

(2) (10%) Do Exercise R-4.29 of the textbook.

(3) (10%) Do Exercise R-4.39 of the textbook.

2.4 Playing with Big Data

You are asked to design and implement a data structure to store a very big data set of KDDCup 2012 Track 2. KDDCup is an international data mining competition, and our dear NTU team won the champion of track 2 that year. There are 149639105 lines in the file

`/tmp2/KDDCUP2012/track2/kddcup2012track2.txt`,

which is placed on 217 workstations from linux1 to linux8. The detailed description of the data set is here: `http://www.kddcup2012.org/c/kddcup2012-track2` . In particular, the format of each line is:

(Click)\t(Impression)\t(DisplayURL)\t(AdID)\t(AdvertiserID)\t(Depth)\t(Position)\t
(QueryID)\t(KeywordID)\t(TitleID)\t(DescriptionID)\t(UserID)

The data set is a log file of Tencent Ad system. Each line means that an user (UserID) is shown with an ad (AdID) at location (Postition)-(Depth) upon querying (QueryID); the ad is shown for (Impression) number of times, and clicked for (Click) number of times; the ad is of properties (DisplayURL), (AdvertiserID), (KeywordID), (TitleID), (DescriptionID). You can view the data set as a super big 5D sparse matrix M with M`[UserID][AdID][QueryID][Position][Depth]`=(Click, Impression) along with some side information per (AdID).

Your design should support the following actions:

- `get`(*u, a, q, p, d*): output (Click, Impression) at M`[UserID][AdID][QueryID][Position][Depth]`
- `clicked`(*u*): output all (AdID, QueryID) pairs that user *u* has made at least one click
- `impressed`(*u1, u2*): output the sorted (AdID), line by line, and its associated properties (DisplayURL), (AdvertiserID), (KeywordID), (TitleID), (DescriptionID) that both users *u1* and *u2* has at least one impression on
- `profit`(a, θ): output the sorted (UserID), line by line, whose click-through-rate (total click / total impression) on a is greater than or equal to θ. Note that 0/0 is defined as 0.

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).
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! If you really cannot think of something, the TAs will give you their basic thoughts in the course forum and they welcome discussions.

(1) (30%) 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) (90%) 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. 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.

- if your program processes the first 1000000 lines of data correctly, you get 30 points
- if your program processes the first 10000000 lines of data correctly, you get another 30 points
- if your program processes all lines of data correctly, you get another 30 points

Please be aware that the data set is huge—10 Gegabytes. Thus, if you are not careful with your implementations, your program can easily crash. Also, if you are programming on the R217 machines, **DO NOT copy the data set to your own directory**. Your own directory is on the NFS system and copying it there would only slow down your program (and other users' programs).

Please submit the code (data structure and demo program) as discussed below.

Submission File

Please submit your written part of the homework on all problems together to R102/R104 before the deadline. For the coding part, the TAs will announce how to submit through github later. Please follow the guidelines of the announcement. Note that the TAs will use your `Makefile` to test your code. **Please make sure that your code can be compiled and run with the Makefile on CSIE R217 linux machines. Otherwise your program "fails" its most basic test and can result in ZERO!**