Homework #2
RELEASE DATE: 03/25/2020
DUE DATE: 04/14/2020, **noon** on Gradescope/CEIBA

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 Array, Linked List, and Recursion

(1) (10%) Do Exercise C-3.3 of the textbook. (The faster the better!)

(2) (10%) The size-N Pascal triangle contains all the values of C_k^n such that $0 \leq n \leq N$ and $0 \leq k \leq n$. Describe how you can store the triangle with a dense one-dimensional array with $\frac{(N+1)(N+2)}{2}$ elements. You need to describe the memory layout and the function for getting C_k^n from the array.
(*No, you do not need to consider how to compute C_k^n. We just ask you to think about the storage.*)

(3) (Bonus 10%) From the previous question, can you come up with another way of storing the triangle that take about half of the space based on the fact that $C_k^n = C_{n-k}^n$? You need to describe the memory layout and the function for getting C_k^n from the array.

(4) (10%) Do Exercise C-3.23 of the textbook.

(5) (10%) Do Exercise C-3.18 of the textbook using either C/C++ or pseudo code.

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

2.2 Analysis Tools

In this problem, you can use any theorems in the textbook and any theorems on the class slides as the foundation of your proof. You **cannot** use any other theorems unless you prove them first.

(1) (10%) Do Exercise R-4.24 of the textbook.

(2) (10%) Do Exercise R-4.26 of the textbook.

(3) (10%) Do Exercise C-4.8 of the textbook. (Note: need "proof"!)

(4) (10%) Do Exercise C-4.22 of the textbook.

2.3 Playing with Big Data

You are asked to design and implement a data structure to store a very big data set of Criteo Sponsored Search Conversion Log. The data set is available here along with descriptions:

`https://ailab.criteo.com/criteo-sponsored-search-conversion-log-dataset/`

The data set is a log file of Criteo Ad system. Each line means that an user (user_id) has clicked a product (product_id) at time (click_timestamp) that eventually results in purchase (Sale) or not, along with some side information. You can view the data set as a super big 3D sparse matrix M with M`[user_id][product_id][click_timestamp]` = Sale along with some side information.

Your design should support the following actions:

- `get`(u, p, t): output (Sale) at M`[`u`][`p`][`t`]`
- `purchased`(u): output the sorted (product_id, click_timestamp) and its associated properties (product_price, product_age_group, product_gender) that user u has made a purchase eventually
- `clicked`($p1$, $p2$): output the sorted (user_id), line by line, that have clicked both products $p1$ and $p2$
- `profit`(t, θ): output first-10 sorted (user_id), line by line, whose average sales amount per click (total number of sales / total click) on clicks after time t 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 on 217 workstation. Please check the spec in detail for what you need to do.

- if your program processes the first 100000 lines of data correctly, you get 30 points
- if your program processes the first 1000000 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—6.4 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**. The TAs will put the 6.4G raw file at

`/tmp2/dsahw2/CriteoSearchData`

on our R217 linux machines. The `/tmp2/` directory provides faster local storage. 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 to Gradescope before the deadline. Also, you need to upload your coding part as a single ZIP compressed file to CEIBA before the deadline. The zip file should contain no directories and only the following items:

- any source code needed
- `Makefile`
- an optional `README`, anything you want the TAs to read before grading your code

The TAs will use the `Makefile` to compile 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!**