Homework 6 (due Saturday, November 17, 11:59pm)

Directions: When asked for a short answer (such as a single number), also show and explain your work briefly and clearly. If you do not provide an explanation, there is no chance for partial credit if your final answer is incorrect.

You do not need to reprove anything we have proved in class. For example, if you are dealing with a random variable that is Poisson with parameter $\lambda$, you can use without proof the fact that its variance is $\lambda$.

Your solutions need to be concise and clear. We will take off points for lack of clarity or for excess verbosity. Please see section worksheet solutions (posted on the course website) to gauge the level of detail we are expecting.

Remember to assign your PDF pages to questions when submitting your solutions on Gradescope. Note that one can assign multiple pages to a single question (and you must assign all pages associated with a question to that question) and a single page to multiple questions (if one solves 2 questions on the same page, which we would strongly prefer that you do not do) We will take off points if you do not properly assign PDF pages to questions.

Please put your final answers to questions in boxes. Again, we will take off points if you don’t do this. Note that putting answers in boxes is easy to do if you use the latex template, which is highly recommended.

1. Count-Min Sketch, Coding Section (40 points)
[Assignment courtesy of Greg Valiant, CS 168, Stanford University]

Goal: The goal of this part is to understand the count-min sketch via an implementation, and to explore the benefits of a conservative updates optimization.

Description: You will use a count-min sketch with $\ell = 4$ independent hash tables. Each hash function will be chosen independently at random as described below, and each hash table will have $b = 256$ counters (256 buckets).

Hash Functions: We first fix a prime number $p$ that is greater than the largest element in the data stream. In this assignment, we set $p = 10007$, and this number should NOT be modified. The hash function for table $j$ (for $j = 1, 2, 3, 4$) is chosen as follows:

- Choose an integer $e_j$ such that $1 \leq e_j \leq p - 1$ uniformly at random. Then choose another integer $g_j$ such that $0 \leq g_j \leq p - 1$ uniformly at random. The integers $e_j$ and $g_j$ are chosen independently and randomly for each $j$.

- The hash function for table $j$ is then defined as $h_j(x) = (e_jx + g_j) \mod p \mod b$.

Implementation Requirements:

(a) Implement the count-min sketch using hash functions as described above.

(b) Implement the conservative updates optimization as follows. When updating the counters during an insert, instead of incrementing all 4 counters, only increment the subset of these 4 counters that have the lowest current count (if two or more of them are tied for the minimum current count, then we increment each of these).

(c) You will consider 3 different data streams, each consisting of a list of integers (not necessarily distinct):
(1) Forward: the elements appear in non-decreasing order.
(2) Reverse: the elements appear in non-increasing order.
(3) Random: the elements appear in a random order.

Call an integer in the stream a **heavy hitter** if the number of times it occurs (its frequency) is at least 1% of the total number of stream elements. Each data stream has 87,925 elements.

For each of the three data streams, feed it into a count-min sketch (i.e., successively insert its elements), and compute the values of the following quantities for each of 10 trials. Finally, compute and report the following quantities averaged over the 10 trials:

- The sketch’s estimate for the frequency of element **9050** after seeing all the elements in the stream. In other words, at the end of the trial, extract and output Count(9050).
- The sketch’s estimate for the number of heavy hitters (elements with estimated frequency at least 1% of the stream length). The heavy hitters are to be determined on the fly.

You will write code for this part in **main** function. If the **isConserv** flag is set to true, perform conservative update, otherwise perform regular update.

We’ve provided starter code, download and unzip:

http://courses.cs.washington.edu/courses/cse312/18au/hw/hw6_skeleton_code.zip

**Description of Starter Code:**

- **Data:** Inside the data folder, there are 3 data stream files: **forward_stream.txt**, **backward_stream.txt**, **random_stream.txt**. Each line in a data stream file is an element in the stream. Each of the 3 data stream files has 87,925 elements (integers, not necessarily distinct).

- **Example Data and Output:** Inside the example_data_and_output folder, there is an example stream called **example_stream.txt**. It also contains 87,925 elements. There are two example output files: **example_output_conservative_update.txt** and **example_output_regular_update.txt**. Use these files to verify the correctness of your implementation. You should get the exact same output with **example_stream.txt**

- **HeavyHitters.java:** This is the file you will modify for your implementation. There are 2 parts of this file that you will need to modify: the CountMinSketch class (a nested class in the HeavyHitters class), and the main function of HeavyHitters. A few notes:
  - Do NOT change any constants in the starter code. Changes to these constants will cause your code to be marked incorrect.
  - You must use RAND to randomly choose $e_j$ and $g_j$ for the hash functions. Do NOT change the seed of RAND. The order to choose $e_j$ and $g_j$ is fixed: first choose $e_1$, then $g_1$, then choose $e_2$, then $g_2$, ..., and finally choose $g_4$. Otherwise, you will NOT pass the tests.
  - You should NOT run through all possible elements or scan through the hash tables at the end of a trial. The heavy hitters are to be determined on the fly.
  - Do NOT modify the provided method headers.
  - Please read the comments in the code carefully to make sure you know what to change and what not to change.

**Running the program:**
To run the program, first compile it with:
then, execute it with:

```$ java HeavyHitters <filepath> <conserv_flag>
```

<filepath> is path to a data stream file. <conserv_flag> is either 0 or 1 to indicate whether you want to use conservative update or regular update.

For example, to execute your program with conservative update:

```$ java HeavyHitters data/random_stream.txt 1
```

To execute your program without conservative update:

```$ java HeavyHitters data/backward_stream.txt 0```

Submitting your work:

There is a Heavy Hitters coding question on Gradescope where you will submit your program. You only need to submit your finished HeavyHitters.java.

Some notes and advice:

- Think about when you should choose $e_j$ and $g_j$ for table $j$, and what data structure you should use to store them.
- Make sure you write all your code in HeavyHitters.java.
- Do not use integer division when calculating your averaged counts.
- Remember to remove any debug statements that you are printing to the output.
- If you use Eclipse, remove all package statements before you turn in your source code.
- Needless to say, you should practice what you’ve learned in other courses: document your program, use good variable names, keep your code clean and straightforward, etc. Include comments outlining what your program does and how. We will not spend time trying to decipher obscure, contorted code.

2. Count-Min Sketch, Discussion Section (17 points)

(a) [5 Points] Does the order of the stream affect the estimated count of 9050 and the estimated count of heavy hitters in your implementation? In either case, explain why. Your answers should address both standard updates and conservative updates:

   (a) Without conservative updates, did order affect the estimated count of heavy hitters? Did order affect the estimated count of 9050?
   (b) With conservative updates, did order affect the estimated count of heavy hitters? Did order affect the estimated count of 9050?

(b) [5 Points] Explain why, even with conservative updates, the count-min sketch never underestimates the count of a value.

(c) [5 Points]

   - What does Markov’s Inequality guarantee about the probability that after seeing $t$ elements $Count(x) \geq f_t^x + 4n/b$? [Recall the notation from class: $f_t^x$ is the number of times element $x$ has occurred among the first $t$ elements of the stream. $n$ is the number of stream elements, $b$ is the size of each hash table and $\ell$ is the number of hash tables used.]
• In the coding section, we had $\ell = 4$ hash tables and the size of each table was $b = 256$. Suppose that element $x$ has frequency $0.005626n$. Use Markov's inequality to bound the probability that your implementation outputs element $x$? In other words, give an upper bound on the probability that at the end

$$\text{Count}(x) \geq 0.005626n + (0.01 - 0.005626)n$$

(d) [2 Points] Briefly describe 2 real world applications of the Count-Min Sketch.

3. Statistics Books (35 points)
Alice is going shopping for statistics books for $H$ hours, where $H$ is a random variable, equally likely to be 1, 2 or 3. The number of books $B$ she buys is random and depends on how long she is in the store for. We are told that

$$\Pr(B = b \mid H = h) = \frac{c}{h}, \quad \text{for } b = 1, \ldots, h,$$

for some constant $c$.

(a) [4 Points] Compute $c$.

(b) [8 Points] Find the joint distribution of $B$ and $H$.

(c) [5 Points] Find the marginal distribution of $B$.

(d) [8 Points] Find the conditional distribution of $H$ given that $B = 1$ (i.e., $\Pr(H = h \mid B = 1)$ for each possible $h$ in 1, 2, 3.

(e) [6 Points] Suppose that we are told that Alice bought either 1 or 2 books. Find the expected number of hours she shopped conditioned on this event.

(f) [4 Points] The cost of each book is a random variable with mean 3. What is the expected amount of money Alice spends?

4. Coal Mining (15 points)
A miner is trapped in a mine containing 4 doors, and each door is equally likely to be chosen. The first door leads to a tunnel that will take him to safety after a number of hours which is Poisson with parameter 2. The second door leads to a tunnel that will take him to safety after a number of hours which is Geometric with parameter $\frac{1}{5}$. The third door leads to a tunnel that will take him to safety after a number of hours which is binomial with parameters $n = 100$ and $p = 1/20$. The fourth door leads to a tunnel which brings him back to where he started after 2 hours. What is the expected number of hours until the miner reaches safety?

5. Duck Hunt (20 points)
Ten hunters are waiting for ducks to fly by. When a flock of ducks flies overhead and hunters fire at the same time, but each chooses his target at random, independently of the others. If each hunter independently hits his target with probability 0.6, compute the expected number of ducks that are hit. Assume that the number of ducks in a flock is a Poisson random variable with mean 6.

6. r-nomial? (15 points)
Consider the following generalization of the binomial rv. Instead of two outcomes, we have $r$ outcomes. Say we roll an $r$-sided die $n$ times with $p_i = \Pr(\text{side } i \text{ appears})$. Let $X_1, \ldots, X_r$ be rv's, where $X_i$ is the number of times side $i$ appeared. What is the joint PMF $p_{X_1,\ldots,X_r}(k_1, \ldots, k_r)$, where $k_1 + \ldots + k_r = n$?