Data Structure and Algorithm, Spring 2023
Homework 4

Red Correction Date: 05/30/2023 11:00

Non-programming part Due: 13:00:00, Tuesday, Jun 13, 2023

Programming part Due: 13:00:00, Friday, Jun 16, 2023

Only two days of late submission is allowed in this homework

TA E-mail: `dsa_ta@csie.ntu.edu.tw`

Rules and Instructions

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

- In Homework 4, the problem set contains a special Problem 0 (see below) and 4 other problems. The set is divided into two parts, the non-programming part (Problems 0, 1, 2) and the programming part (Problems 3, 4).

- For problems in the non-programming part, you should combine your solutions in *one* PDF file. Your file should generally be legible with a white/light background—using white/light text on a dark/black background is prohibited. Your solution must be as simple as possible. At the TAs' discretion, solutions that are too complicated can be penalized or even regarded as incorrect. If you would like to use any theorem which is not mentioned in the classes, please include its proof in your solution. **If you use pseudocode to describe your algorithm, you need to briefly explain how your pseudocode works.**

- The PDF file for the non-programming part should be submitted to Gradescope as instructed, and you should use Gradescope to tag the pages that correspond to each subproblem to facilitate the TAs' grading. Failure to tag the correct pages of the subproblem can cost you a 20% penalty.

- For the programming part, you should have visited the *DSA Judge* (`https://dsa2023.csie.org/`) and familiarized yourself with how to submit your code via the judge system in Homework 0.

- For problems in the programming part, you should write your code in C programming language, and then submit the code via the judge system. Each day, you can submit up to 5 times for each problem. To encourage you to start early, we allow 10 times of submissions per day in the first week **(from 2023/05/26 to 2023/06/01)**. The judge system will compile your code with

```
gcc main.c -static -O2 -std=c11
```

- For debugging in the programming part, we strongly suggest you produce your own testcases with a program. You can also use tools like `gdb` or compile with flag `-fsanitize=address` to help you solve issues like illegal memory usage. For `gdb`, you are strongly encouraged to use compile flag `-g` to preserve symbols after compiling. Note: For students who use IDE (VScode, DevC++...) you can modify the compile command in the settings/preference section. Below are some examples:
  - `gcc main.c -O2 -std=c11 -fsanitize=address # Works on Unix-like OS`
  - `gcc main.c -O2 -std=c11 -g # use it with gdb`
- 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.
- **The score of the part that is submitted after the deadline will get some penalties according to the following rule:**

$$\textit{Late Score} = \max\left(\left(\frac{\mathbf{2} \times 86400 - \textit{Delay Time}(\textit{sec.})}{\mathbf{2} \times 86400}\right) \times \textit{Original Score}, 0\right)$$

  non-programming part and the programming part will be calculated separately.
- If you have questions about the homework, please go to the discord channel and discuss (*strongly preferred*, which will provide everyone with a more interactive learning experience). If you really need an email answer, please follow the rules outlined below to get a fast response:
  - The subject should contain two tags, `"[hw2]"` and `"[Px]"`, specifying the problem where you have questions. For example, `"[hw2][P3] Is k in subproblem 3 an`

`integer"`. Adding these tags allows the TAs to track the status of each email and to provide faster responses to you.

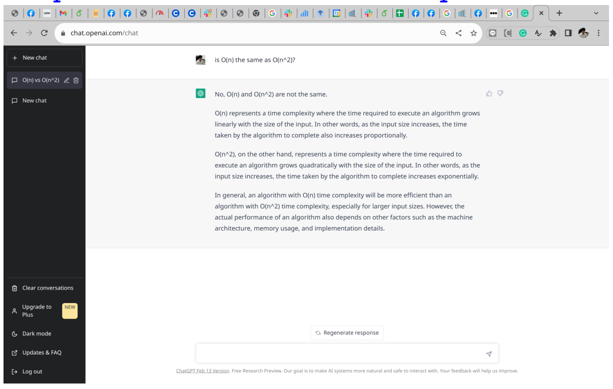
- If you want to provide your code segments to us as part of your question, please upload it to Gist or similar platforms and provide the link. Please remember to protect your uploaded materials with proper permission settings. Screenshots or code segments directly included in the email are discouraged and may not be reviewed.

Non-Programming Part

Problem 0 - Proper References (0 pts)

For each problem below, please specify the references (the Internet URL you consulted with or the classmates/friends you discussed with) in your PDF submitted to Gradescope. If you have used chatGPT or similar tools, please provide a quick snapshot of your interaction with the tools. *You do not need to list the TAs/instructors.* If you finished any problem all by yourself (or just with the help of TAs/instructors), just say "all by myself." While we did not allocate any points to this problem, failure to complete this problem can lead to penalty (i.e. negative points). Examples are

- Problem 1: Alice (B86506002), Bob (B86506054), and
  https://stackoverflow.com/questions/982388/

  is O(n) the same as O(n^2)?

  No, O(n) and O(n^2) are not the same.

  O(n) represents a time complexity where the time required to execute an algorithm grows linearly with the size of the input. In other words, as the input size increases, the time taken by the algorithm to complete also increases proportionally.

  O(n^2), on the other hand, represents a time complexity where the time required to execute an algorithm grows quadratically with the size of the input. In other words, as the input size increases, the time taken by the algorithm to complete increases exponentially.

  In general, an algorithm with O(n) time complexity will be more efficient than an algorithm with O(n^2) time complexity, especially for larger input sizes. However, the actual performance of an algorithm also depends on other factors such as the machine architecture, memory usage, and implementation details.

  ChatGPT Feb 13 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

- Problem 2: all by myself

- ...

Listing the references in this problem does *not* mean you could copy from them. We cannot stress this enough: *you should always write the final solutions alone and understand them fully.*

Problem 1 - Raffle Tickets (100 pts)

"CSIE Night" is an event held annually in the CSIE Kingdom where residents can perform their talents on the stage. During the break, the audience will enter a draw, and mysterious prizes will be given to the winners. As the organizer of the event, Superwoman Lin had to come up with several methods to hash the information of the attendees to some hash tables to organize the raffle tickets. Please help her to complete the tasks.

1. (15 pts) During the organizer interview, Professor Cheng tested Lin with the following problem:

   There are several keys to be inserted into the hash table by $h_1(x)$ and $h_2(x)$. Let

   $$\begin{aligned} h_1(k) &= 8k \mod N \\ h_2(k) &= 1 + ((2k - 2) \mod (N - 1)) \end{aligned}$$

   Given the number of table entries $N = 7$ and the keys given in order: $[72, 23, 51, 3, 18, 40, 21]$, please **specify in a step-by-step manner** how the keys are inserted into the hash table using open addressing with two different methods:

   (a) linear probing (probe next grid) with hash function $h_1(k)$

   (b) double hashing with primary hash function $h_1(k)$ and secondary hash function $h_2(k)$.

   Specify the answers using the format below. Assume each table entry can only hold one key.

keys to be inserted \ index	0	1	2	3	4	5	6
72							
23							
51							
3							
18							
40							
21							

2. (25 pts) In CSIE Kingdom, most names of the residents are the rotation with one another; that is, if there are residents with name **John**, then there are also residents with names **Ohnj**, **Hnjo** and **Njoh**. The first letter of the name is always in uppercase and the rest are in lowercase. Lin decided to hash them with the Rabin-Karp algorithm. Assume that we use ASCII codes to convert characters into numbers. The length of the name is L,

radix-d numeracy system d (i.e., $|\Sigma|$, the size of the character set used in the string[1].), and the modular number Q. Derive an algorithm that can hash a name and all its rotation in $O(L)$-time. (Be careful of the upper and the lower cases)

3. (30 pts) During the CSIE night event, the prize-drawing uncle (抽獎阿伯) was eager to win the grand prize. There are 30 prizes and 840 audiences in the event, with each audience holding exactly one ticket. Every prize is determined by randomly drawing one ticket out of the 840 tickets and returning the ticket back to the pool after each draw. The prize-drawing uncle was very lucky to get two prizes and went on stage to claim his prize twice. At this point, some of the audience shouted out,

   "It's unfair! The probability of anyone in the audiences being drawn twice is too low, only $\frac{2}{840^2}$, so the prize-drawing uncle must have put in many tickets."

   Lin, who got an A+ in the DSA course, realized that these statements were strange, but was too tired from organizing the CSIE night event to argue back. Can you help her point out what's wrong with these audience's arguments?

4. (30 pts) Lin is busy finishing her homework after CSIE night. There are two hashing problems from the IDSA (Introduction of Data Structure and Algorithm) course:

   (a) Given a string key and a set S of n strings $S = \{s_1, s_2, \ldots, s_n\}$. Determine if there exist any $s_i \in S$ equal to key.

   (b) Given a set S of n strings $S = \{s_1, s_2, \ldots, s_n\}$. Determine if there exist two strings s_i, s_j such that $s_i = s_j$ and $s_i, s_j \in S$.

   Lin uses a hashing function such that the hashing value is uniformly distributed among $[0, P-1]$. Assume $n = 10^5$. Lin chooses $P = 10^9 + 7$ (a prime number), and she got AC (Accepted) for the first problem and WA (Wrong Answer) for the second problem. Can you explain why by analyzing how likely spurious hits happen?

[1]Mentioned in page 9 of the string matching slide deck.

Problem 2 - README (100 pts)

Hi, this is your TA Souffle speaking. Listen carefully. Designing Students' Assignments (DSA) is so difficult. We try to strike a balance between fun story-like problem descriptions and the length of it. Moreover, it is hard to meet everyone's requirements, such as the difficulty of problems, the grading policy, the hint for homework, etc. We try to maintain the balance to the best of our ability. As a result, we need a self-balancing data structure to meet these requirements. :P

1. (15 pts) Souffle prefers to store problems in an RB tree. However, there could be bad problems that are too hard. These problems are marked black in the RB tree. On the other hand, good problems in the RB tree are marked red. Ignoring the black NIL (i.e., the leaves, or the external node), what is the maximum ratio of good problems to bad ones? Construct an RB tree with 15 nodes that reach the maximum ratio of good problems to bad ones.

For the following problems, it is not necessary to provide pseudo code as those introduced in the class materials can be re-used in this problem. However, you should clearly explain how you modify these algorithms to support the features described in the problems.

2. (25 pts) Now, Souffle has designed n problems and stored them in an RB-tree. Each problem p_i has a difficulty level d_i, $1 \le i \le n$. Assume that the RB-tree uses d_i as the key of the stored problems. In order to find a problem of appropriate difficulty level, the given RB tree needs to support a new operation which finds the k-th easiest problem. Note that for this problem, you are not allowed to create a completely new data structure, but you can add additional information to the RB-tree, stored in the tree node. Note that in this problem, the RB-tree will not have any new insertion or deletion.

   Complete the "update" of the RB-tree (i.e., to add additional information) in $O(n)$-time. Then, any operation to query the k-th easist problem should be completed in $O(\log n)$-time.

3. (30 pts) Now, we need to further develop the existing problem-holding RB-tree. Let n be the number of the problems which have been designed by Souffle and not yet rejected by the professor. Assume we start with $n = 0$. In addition to be able to find the k-th easiest problem at any time in $O(\log n)$-time, as in the previous problem, the updated RB-tree should also be able to perform insertion of a new problem and deletion of an existing problem, both in $O(\log n)$-time. Explain clearly how you can revise the existing operations of the RB-tree to achieve each of these tree operations.

4. (30 pts) As a TA, Souffle has to reply to emails. However, as some emails are impolite, Souffle can only take so much each day. Thus, the main objective for Souffle is to reply as many emails as possible each day.

   Each email has an impoliteness score of z_i, which is an non-negative integer, where i is the index of the email. Souffle starts the day with an emotional health score of E. Responding to the i-th email results in deduction of z_i from E. Once the emotional health score is equal or smaller than zero, Souffle will not be able to respond to any email for the day.

   Assume we start with no email. Modify the RB-tree in the previous problem, such that we can determine the maximum number x of emails Souffle can respond to, given the emotional health score E when the day starts. This should be performed in $O(\log n)$-time. You can just briefly describe how to find the x. If you really want to write pesudo code, please return x as your answer. Moreover, the RB-tree should be able to hold any email which has not yet been responded to. Thus, insertion of a new email and deletion of a processed email should both be performed in $O(\log n)$-time. Explain clearly how you can revise the existing operations of the RB-tree to achieve each of these tree operations.

Programming Part

Problem 3 - Alexander Loves Sweet Dumplings 2 (100 pts)

Problem Description

There are N types of sweet dumplings for sale in DSA (Deluxe Supermarket Alliance). On any given day, each type of the dumplings have two packs available for purchase. However, these two packs of the same type have different prices.

At the end of each day, Alexander and Bomboo both want to purchase all types of dumplings, each with one pack. Both of them want to make the purchase with minimum costs. As a result, they reached an agreement to buy the sweet dumplings with these rules:

1. On any day, they will each take turn to buy one pack of sweet dumpling until they complete the purchases for the day. Alexandar always goes first at the beginning of any day. Both of them are genius and they will use optimal strategy to make the purchases, in order to save money.
2. For each of the M days, each of them will complete the purchase for all N types of dumplings, each with one pack.
3. At the beginning of each day, each of the N types of sweet dumplings are restocked to two packs.
4. At the end of each day, the prices of two packs of one type of dumpling will be changed.

Your goal is to help Alexandar to determine the cost to make the purchase for each of the M days.

Input

The first line contains two integers, N and M, representing the number of types of the sweet dumplings and the number of the days to make the purchases.

The next N lines contain two integers each, a_i and b_i, representing the price of two packs of the i-th type of sweet dumplings.

In the following $M-1$ lines, the i-th line contains five integers t_i, c_i, d_i, e_i, f_i. Indicating that the price of t_i-th sweet dumplings will change to $(c_i \times P + d_i \mod 1000000007)$ and $(e_i \times P + f_i$

mod 1000000007) on day $i + 1$, where P is the minimum cost for Alexander on the previous day.

(This problem has nothing to do with the mathematical properties of the price-updating formula.)

Output

The output should consist of M lines. The i-th line should contain a single integer representing the minimum cost for Alexander to buy all types of sweet dumplings, each with one pack, on the i-th day.

Constraints

- $1 \le N \le 10^5$
- $1 \le M \le 10^5$
- $1 \le a_i, b_i \le 10^9$
- $0 \le c_i, d_i, e_i, f_i \le 10^9$
- $1 \le t_i \le N$

Subtask 1 (5 pts)

- $M = 1$

Subtask 2 (10 pts)

- $1 \le N, M \le 10^3$

Subtask 3 (15 pts)

- a_i, b_i are generated randomly.

Subtask 4 (70 pts)

- No other constraints.

Sample Testcases

Sample Input 1

```
4 6
2 4
```

```
5 7
1 7
2 1
4 0 5 0 2
1 0 6 0 2
4 0 4 0 3
2 0 1 0 3
3 0 6 0 6
```

Sample Output 1

```
12
15
16
16
12
14
```

Sample Input 2

```
5 4
1 1
2 3
4 6
7 10
11 15
1 0 1 0 6
3 1 4 1 5
5 4 3 2 1
```

Sample Output 2

```
29
31
61
174
```

Hints

Explanation of Sample 1:

- First day, Alexander can buy each type with price $4, 5, 1, 2$ respectively to minimize his cost.
- Second day, the price of type 4 is changes to $12 \times 0 + 5$ and $12 \times 0 + 2$. Hence the price of each type is $(2, 4), (5, 7), (1, 7), (5, 2)$ respectively. And Alexander can buy each type with price $2, 7, 1, 5$ respectively to minimize his cost.
- Third day, the price of type 1 is changes to $15 \times 0 + 6$ and $15 \times 0 + 2$. Hence the price of each type is $(6, 2), (5, 7), (1, 7), (5, 2)$ respectively. And Alexander can buy each type with price $6, 7, 1, 2$ respectively to minimize his cost.
- and so on.

Problem 4 - Rotating... Again?! (100 pts)

Problem Description

Do you recall the game developer, Little Cucumber, who recently launched his game? He invited a magical fairy to give it a try, but to his surprise, the fairy cast a bizarre rotating spell on the game... (See more)

Long story short, you have to maintain a multiset of strings S. At the beginning, S has N strings. Let t_i denote the i-th string in S, $1 \le i \le N$. Then, Q operations will be performed on S, where each operation can be one of the following two:

1. `insert`(t_j): insert the string t_j into S.
2. `remove`(t_j): remove one instance of the string t_j from S, it is guaranteed that at least one t_j is in S.

Consider two strings, A and B, of the same length L. A and B are *rotationally identical* if and only if there exists $k, 1 \le k \le L$ such that

$$A_1 A_2 \ldots A_L = B_k B_{k+1} \ldots B_L B_1 \ldots B_{k-1} \quad [2]$$

, i.e., if B is rotated by k characters, then A and B are identical.

You are asked to output *the number of string pairs* in S which are *rotationally identical* before the first operation and after each operation.

Input

The first line contains two integers N and Q, the number of strings inside S initially and the number of operations, respectively. Then, the next N lines have t_i in order, $1 \le i \le N$, where t_i is a string in S initially. Each of the next Q lines contains an integer P and a string t_j.

- If $P = 1$, it represents an `insert`(t_j) operation.
- If $P = 2$, it represents a `remove`(t_j) operation.

Output

The output should consist of $Q + 1$ lines. The first line should contain an integer representing the number of rotationally identical string pairs in S initially. Each of the next Q lines should also contain an integer representing the number of rotationally identical string pairs in S after each operation.

[2] $k = 1$ is a special case representing no rotation of B.

Constraints

- $2 \leq N \leq 10^6$
- $0 \leq Q \leq 10^6$
- $1 \leq |t_i| = |t_j| = M \leq 10^6$
- t_i and t_j consists of only lowercase Latin letters.
- $(N + Q)M \leq 10^6$ (the total string length does not exceed 10^6.)

Subtask 1 (10 pts)

- $N, Q, M \leq 50$

Subtask 2 20 pts)

- $Q = 0$

Subtask 3 (20 pts)

- $M = 3$

Subtask 4 (50 pts)

- No other constraints.

Hints

Consider the algorithm you developed for problem 1.2, which calculates the Rabin-Karp hashes for all rotations of a string. Think about how to compare between the hashes of all rotations of two strings.

Sample Testcases

Sample Input 1

```
6 0
ntu
tun
unt
sad
dsa
qqq
```

Sample Output 1

```
4
```

Sample Input 2

```
3 7
bananana
nananaba
cucumber
1 nanabana
1 nabanana
1 anananab
2 cucumber
1 cumbercu
1 bercucum
2 nanabana
```

Sample Output 2

```
1
3
6
10
10
10
11
7
```

Sample Input 3

```
2 3
walnut
walnut
1 walnut
1 walnut
2 walnut
```

Sample Output 3

```
1
3
6
3
```