Data Structure and Algorithm, Spring 2024
Homework 1

RED CORRECTION: 06:00:00, March 7, 2024

Due: 13:00:00, Tuesday, March 26, 2024

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.

- 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 (including chatGPT) 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.

- In Homework 1, 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 texts on a dark/black background is prohibited. Your solution must be as simple as possible. At the TAs' discretion, solutions which 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.

- 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 tagging the correct pages of the subproblem can cost you a 20% penalty.

- For problems in the programming part, you should write your code in C programming language, and then submit the code via the *DSA Judge*:

  https://dsa2024.csie.org/.

  Each day, you can submit up to 5 times for each problem. 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 test-cases with programs and/or 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, but possibly not M1/M2`
  - `gcc main.c -O2 -std=c11 -g # use it with gdb`

- The score of the part that is submitted after the deadline will get some penalties according to the following rule:

$$Late\ Score = \max\left(\left(\frac{5 \times 86400 - Delay\ Time(sec.)}{5 \times 86400}\right) \times Original\ Score, 0\right)$$

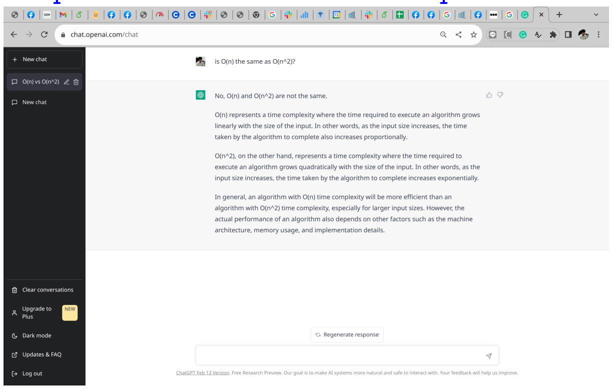
- If you have questions about HW1, please go to the Discord channel and discuss (*strongly preferred*, which will provide everyone with a more interactive learning experience). If you need an email answer, please follow the rules outlined below to get a fast response:

  - Please include the following tag in the subject of your email: `"[HW1.Px]"`, specifying the problem where you have questions. For example, `"[HW1.P2] Is k in subproblem 5 an integer?"`. Adding the tag 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.

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 on 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 - Analysis Tools (100 pts)

Suppose that f and g are asymptotically non-negative functions defined on $\mathbb{N}$. Recall the following asymptotic notations:

- $f(n) = O(g(n))$ if and only if there exist positive numbers c and n_0 such that

$$f(n) \leq cg(n) \text{ for all } n \geq n_0.$$

- $f(n) = \Omega(g(n))$ if and only if there exist positive numbers c and n_0 such that

$$f(n) \geq cg(n) \text{ for all } n \geq n_0.$$

- $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Read the following pseudo code carefully. Write down the time complexity for each of the following functions using the Θ notation along with an expression of n. Two requirements:

- Your answer should be as simple as possible. For instance, $\Theta(\frac{1}{n} + \lg(n^{543}) + 8763n^2)$ should be written as $\Theta(n^2)$.
- Your answer should also come with a *brief* explanation. The word *brief* means "very short." An example of a *brief* explanation is as follows.

Sample problem: Write down the time complexity for this function using the Θ notation along with an expression of n.

```
HELLO-WORLD($n$)
1  for $i = 1..2n$
2      for $j = 1..2n$
3          print("Hello world")
```

Sample solution: The iteration counts of the two loops are $2n$ and $2n$ respectively, so there are $4n^2$ iterations. Hence the time complexity is $\Theta(n^2)$. $\square$

1. (15 pts)

```
CEILING-SQRT($n$)
1  for $i = 1..n$
2      if $i \times i \geq n$
3          return $i$
```

2. (15 pts)

```
TRIPLE-SUMMATION($n$)
1  $sum = 0$
2  for $i = 1 \,.\,.\, n$
3      for $j = i \,.\,.\, n$
4          for $k = i \,.\,.\, j$
5              $sum = sum + 1$
```

3. (15 pts)

```
DOUBLE-TWODOWN($n$)
1  if $n > 0$
2      DOUBLE-TWODOWN($n - 2$)
3      DOUBLE-TWODOWN($n - 2$)
4      DOUBLE-TWODOWN($0$)
```

Next, prove or disprove each of the claims below. You should provide a *formal proof.* That is, you need to specify all constants like c and n_0 or use any theorems that have been proved in class if you believe the claim to be true, or provide a counterexample if you believe the claim to be false.

Sample problem: Prove that $an + b = \Theta(n)$, where a and b are positive constants.
Sample solution 1: Observe that $an \leq an + b \leq 2an$ for $n \geq \lceil \frac{2b}{a} \rceil$. Let $n_0 = \lceil \frac{2b}{a} \rceil > 0, c_1 = 1 > 0, c_2 = 2 > 0$ in the definition of Θ, we see that $an + b = \Theta(n)$. $\square$
Sample solution 2: Rewrite $an + b = n(a + bn^{-1})$. We see that $\lim_{n\to\infty} \frac{an+b}{n} = a > 0$. Using the proof in class that justifies how the limit definition leads to the formal definition, we establish that $an+b = \Theta(n)$. $\square$

4. (15 pts) (Prove or disprove) $n! = O(2^n)$.

5. (20 pts) (Prove or disprove) For any $p \in \mathbb{N}$, $q > 1$, $n^p = O(q^n)$. You can use the fact that $\ln n \leq n - 1$ for $n \geq 1$ if needed.

6. (20 pts) (Prove or disprove) Let $\{f_k(n)\}_{k=1}^{\infty}$ be a sequence of real-valued functions defined on $\mathbb{R}^+$ such that $f_k(n) = O(n)$ for $k \in \mathbb{N}$. Then, $\sum_{k=1}^{n} f_k(n) = O(n^2)$.

Problem 2 - Quacking Ducks (100 pts)

For all sub-problems listed below, unless stated otherwise, please ensure that your answers consist of the following components:

- Please describe your algorithm using pseudocode and provide a **brief explanation** of why it works. Your pseudocode for each sub-problem should be concise and easy to read, **consisting of no more than 30 lines**. Please note that overly complex or lengthy pseudocode may result in some penalties.
- Analyze the time complexity and extra-space complexity of your algorithm. Please note that if your algorithm is not efficient in terms of time or space, you may not receive full credit for your solution.

DSA (Duck Silencing Airplane) is a popular course at NTU (No Teacher University). In 2023 Spring, there were about 400 people enrolled in this course. Literally, in the first half of the semester, the students need to learn to use paper airplanes to make the ducks outside the classroom quiet (but cannot harm them). While it may seem straightforward, it is important to note that only students whose final grades exceed 95 can earn an A+ in 2023 Spring.

1. (25 pts) To get a higher grade in this course, a student bought n ducks with $n \geq 6$ to practice the technique taught in the course. The student numbered those ducks from 1 to $n-3$ and arranged them in increasing order by their number in an ordered array. Because of the "duck"-hole principle, some numbers must be repeated. Ze believed that three of the numbers were repeated exactly once. Can you design an algorithm to find out the three repeated numbers in the ordered array in $O(\log n)$ time and $O(1)$ extra space?

2. (25 pts) One DSA TA also bought m ducks with $m \geq 2$ for the class, and ze numbered those ducks from 1 to $m-1$. Because the TA was very busy, ze did *not* arrange the ducks in an ordered manner. Ze just placed the ducks in an array with m elements. Again, because of the "duck"-hole principle, some number(s) must be repeated. Ze believes that one of the numbers is repeated exactly once. Can you design an algorithm to find out the repeated number in the array in $O(m)$ time and $O(1)$ extra space?

After the midterm exam, the course intended to instruct students on silencing ducks *on actual airplanes*. Unfortunately, budget constraints prevent purchasing airplane tickets for all students. Consequently, during the last half of the semester, only theoretical lessons are provided, and there is no need to keep actual ducks around. Thus, the ducks are released into the Drunken Moon Lake post-midterm. Yes, that is why there are so many ducks near the lake. ;-)

3. (20 pts) Hsin-Mu thought that there were too many ducks on campus, and decided to ask the TAs to bring n ducks to the *other* DSA (Data Structures and Algorithms) course for its exciting earth game. It is so *coincidental* that the other course is also abbreviated as DSA. ;-) Anyway, the n ducks were so energetic that the TAs found it challenging to escort them to the classroom. Consequently, the TAs devised a game for the ducks. Initially, the n ducks were arranged in a random order as a list. Then, one TA would shout out a number k ($1 \leq k \leq n$), prompting the last k ducks in the list to reverse their order, and then cut in front of the other ducks in the list.

Example 1

- Original list:

$$1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 9$$

- After shouting out $k = 1$

$$\mathbf{9} \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 7$$

Example 2

- Original list:

$$1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 9$$

- After shouting out $k = 3$

$$\mathbf{9} \rightarrow \mathbf{7} \rightarrow \mathbf{5} \rightarrow 1 \rightarrow 3$$

Assume that the list of ducks is represented as a *doubly linked list* within the computer, can you design an algorithm to simulate the game in $O(k)$ time and $O(1)$ extra space per shouting?

Note: with the requirement on $O(1)$ extra-space complexity, you cannot afford to create new nodes to solve this sub-problem, and need to re-use the nodes that have been allocated for the original linked list.

4. (30 pts) The ducks did a great job in the earth game. Hsuan-Tien thought they were so smart and exhibited some Aviation Intelligence (AI). So he brought them to his lab and trained them to learn some matrix operations. The n ducks, each representing a number, line up as a list to represent a k by m matrix with k rows (with $n = k \times m$). For instance, when $k = 3$ and $m = 2$, the following ducks

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6,$$

represent a 3×2 matrix.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}.$$

A fundamental operation on the matrix is *transposing*, i.e., changing the matrix to

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}.$$

or equivalently in the list representation

$$1 \rightarrow 3 \rightarrow 5 \rightarrow 2 \rightarrow 4 \rightarrow 6.$$

Assume that the list of n ducks (k by m matrix) is represented as a *singly linked list* within the computer, can you design an algorithm to modify the linked list such that the corresponding matrix is transposed in $O(n)$ time and $O(k)$ extra space?

Programming Part

Problem 3 - Skip List (100 pts)

Time Limit: 1 s

Memory Limit: 16384 KB

Problem Description

When using linked lists to store ordered elements, a big disadvantage (over ordered arrays) is that searching for an element cannot be done efficiently with binary search because it is difficult to efficiently access the "mid-point" of a linked list. The skip list is here for rescue! In this problem, we ask you to implement the skip list (which includes an ordered linked list in its bottom layer). The following description of the skip list is modified from Wikipedia (http://en.wikipedia.org/wiki/Skip_list):

A skip list is built in layers. The bottom (zero-th) layer is an ordinary ordered linked list. Each higher layer acts as an "express lane" for the lists below, where a node in layer i appears in layer $i+1$ with probability $\frac{1}{2}$.

A search for a target node begins at the head in the top list, and proceeds horizontally until the current node is greater than or equal to the target. If the current node is equal to the target, it has been found. If the current node is greater than the target, or the search reaches the end of the linked list, the procedure is repeated after returning to the previous node and dropping down vertically to the next lower list.

Note that the nodes in the upper layers act like access points to the "mid-point" in some sense, therefore facilitating rapid search.

In this problem, we ask you to implement the *reversely* ordered skip list. That is, the skip list is ordered from the largest number to the smallest one. Here are the $4+1$ routines that we ask you to implement

```
SlowGet($L, data$)
1  $node = L[0].head$ // the bottom layer, which is simply a reversely ordered linked list
2  while $node.next \neq$ NIL and $data \leq node.next.value$
3      $node = node.next$
4  if $node.value = data$
5      return $node$
6  else
7      return NIL
```

FastGet($L, data$)

```
1  $node = L[L.FastLayers].head$ // the top layer
2  while $node.below \neq$ NIL
3      while $node.next \neq$ NIL and $data \leq node.next.value$
4          $node = node.next$
5      $node = node.below$
6  while $node.next \neq$ NIL and $data \leq node.next.value$
7      $node = node.next$
8  if $node.value = data$
9      return $node$
10 else
11     return NIL
```

Insert($L, data$)

Insertion can be done by first taking FastGet while storing the "previous" node of each layer. Then, one can insert the new node after the "previous" node of each layer probabilistically. Note that if a node is inserted into the top layer, then an additional coin flipping is needed to see if one more layer should be added.

To make insertion deterministic (which facilitates the judge system to do grading), please use the following CoinFlip routine to determine whether a number k should be inserted into the i-th layer for $i > 0$.

$$\text{CoinFlip}(k, i) = \text{IsOdd}(\lfloor k/2^{i-1} \rfloor).$$

Remove($L, data$)

Removal can be done by first taking FastGet and removing the target nodes (in all layers) once they have been found. Note that if all nodes in the top layer are removed, remember to remove the top layer by updating $L.FastLayers$.

Input

Assuming that the skip list is empty initially. The first line includes one integer, M, representing the number of operations. The next M lines include 2 integers, t and k. The first integer, t, represents the operation type:

- Type 1 is for the SlowGet operation. You need to find the smallest number that is greater than or equal to k by running the SlowGet function described above.

- Type 2 is for the FastGet operation. You need to find the smallest number that is greater than or equal to k by running the FastGet function described above.
- Type 3 is for the Insert operation. You are required to insert k into the skip list.
- Type 4 is for the Remove operation. You are required to remove k from the skip list. If k is not in the skip list, simply ignore it and proceed to the next operation.

Output

For each SlowGet and FastGet operation, if there is some number that is greater than or equal to k in the skip list, please print the values of **all** nodes that have been accessed (line 2.5 of SlowGet, line 3.5 and line 6.5 of FastGet) during the searching process in one line. Otherwise, please output `-1` in one line.

Constraints

- $1 \le M \le 10^5$
- $0 \le k \le 10^{18}$
- $t \in \{1, 2, 3, 4\}$
- It is ensured that there are no duplicate numbers in the skip list at any given time, but a number may be re-inserted after being removed.

Subtasks

Subtask 1 (20 pts)

- $t \in \{1, 3, 4\}$
- $1 \le M \le 3 \times 10^3$
- The two constraints mean that the task can be solved with the bottom layer of a skip list, a typical linked list, or even an array.

Subtask 2 (40 pts)

- $t \in \{2, 3\}$, i.e., skip list without Remove

Subtask 3 (40 pts)

- $t \in \{2, 3, 4\}$, i.e., a fully functional skip list

Sample Test Cases

Sample Input 1

```
6
3 7
3 14
3 49
1 10
1 7
1 50
```

Sample Output 1

```
49 14
49 14 7
-1
```

Sample Input 2

```
8
3 1
3 2
3 3
3 4
3 5
1 1
2 4
2 1
```

Sample Output 2

```
5 4 3 2 1
5 5 4
3 3 1 1
```

Sample Input 3

```
9
3 7
3 14
3 49
2 7
4 7
2 7
4 14
4 49
2 7
```

Sample Output 3

```
7 7 7 7
49 49 14
-1
```

Hints

In Sample 2, after completing all insertions, the skip list depicted in Figure 1 will be obtained.

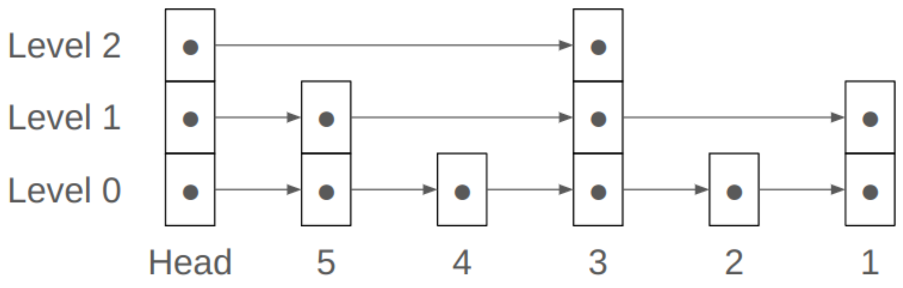


Figure 1

When we perform the FastGet operation for number 4, we traverse through the header in both levels 2 and 1, then through 5 in both levels 1 and 0, and finally through 4 in level 0. As a result, you should output 5 5 4. The search path is shown in Figure 2.

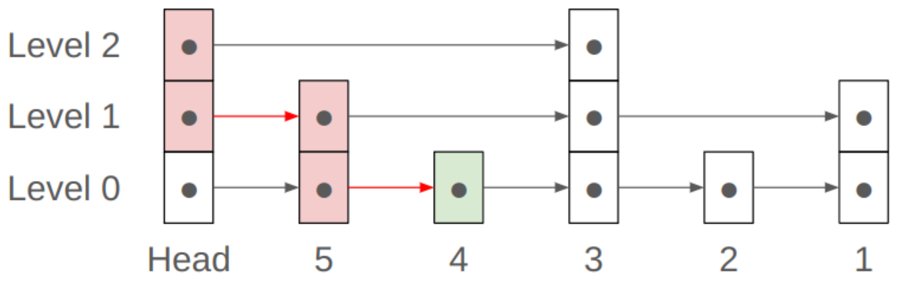


Figure 2

Similarly, when we perform the FastGet operation for number 1, we traverse through the header in level 2, then through 3 in both levels 2 and 1, and finally through 1 in both levels 1 and 0. As a result, you should output 3 3 1 1. The search path is shown in Figure 3.

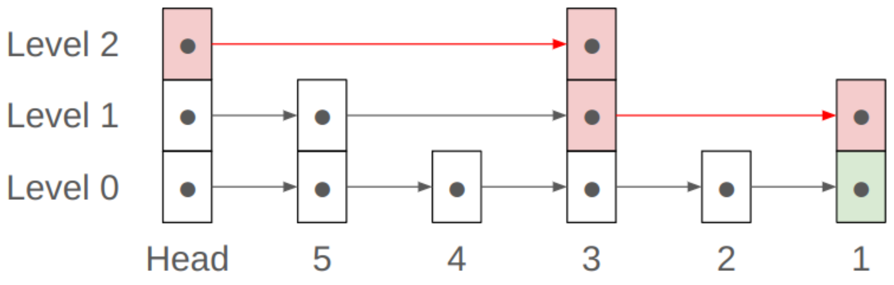


Figure 3

Last but not least, while we illustrate the skip list with singly-linked lists, there is nothing that prevents you from implementing it with doubly-linked lists (if that is easier for you and you can control the memory usage to be within the limit).

Problem 4 - Dragon's Song: Awakening (100 pts + 20 bonus pts)

Time Limit: 3 s
Memory Limit: 524288 KB

Problem Description

N classmates of Little Shimeming are playing Dragon's Song: Awakening (DSA), a mobile game notorious for its inflationary powers. The classmates are labeled from 1 to N based on their initial ranks in the game. Each of them initially possesses a power denoted as $p_1, p_2, \cdots, p_N$, where $p_1 \geq p_2 \geq \cdots \geq p_N$.

The game proceeds with T incidents happening sequentially. The i-th incident is one of the following:

1. The classmate labeled a_i at rank j attacks! That is, ze increases zir power to be the same as the classmate that is right in front of zir (i.e. the classmate at rank $j-1$). Then the two classmates swap their ranks.
   - If the classmate labeled a_i is already at the first place, nothing happens and **the attack does not count**.
   - If the two classmates are of the same power, there is no power increase but the swapping should still happen. That is, **the attack still counts**.
2. The game rewards each classmate according to zir rank. The classmate at rank 1 increases zir power by $(N-1)$ points, the classmate at rank 2 increases zir power by $(N-2)$ points, and so on.
3. Little Shimeming can query the last rank that is of power $\geq q_i$, and report the label of the classmate at that rank.
4. Little Shimeming can query the total power increase of classmate labeled b_i from the last m_i attacks, where m_i is no more than some known constant M.

Input

The first line contains three space-separated integers N, T, and M. The second line contains N integers p_1, p_2, $\cdots$, p_N. The i-th line of the following T lines contains one, two, or three integers depending on the incident:

- 1 a_i, indicating incident 1 happens.
- 2, indicating incident 2 happens.
- 3 q_i, indicating incident 3 happens.
- 4 b_i m_i, indicating incident 4 happens.

Output

The output consists of two parts separated by an empty line. The first part consists of the answers to incidents 3 and 4.

- For each incident 3, output 2 integers in a line, representing the last rank that is of power $\geq q_i$, and report the label of the classmate at that rank. If there is no such classmate, please output two numbers `0 0` as a single line.
- For each incident 4, output 1 integer in a line, the total power increase of classmate labeled b_i from the last m_i attacks. If the number of attacks from b_i is less than m_i, output the total power increase of classmate labeled b_i from all attacks.

The second part consists of the game record. It contains N lines, and the j-th line is the record for the classmate labeled j. The first integer k_j of the line denotes the number of attacks that the classmate has executed. Then, the next k_j numbers denote the power gained from (i.e. the difference) the first attack, the second attack, etc. Please separate the integers by space.

Constraints

- $1 \leq N \leq 10^6$
- $1 \leq T \leq 5 \cdot 10^5$
- $1 \leq M \leq 5 \cdot 10^5$
- $0 \leq p_N \leq p_{N-1} \leq \cdots \leq p_1 \leq 10^9$
- $1 \leq a_i, b_i \leq N$
- $0 \leq q_i \leq 10^{18}$
- $1 \leq m_i \leq M$

Subtasks

Subtask 1 (10 pt)

- $1 \leq N \leq 10^3$
- $1 \leq T \leq 10^3$
- $m_i = M$

Subtask 2 (10 pts)

- Only incidents 1 and 3 occur.

Subtask 3 (30 pts)

- $m_i = M = 1$

Subtask 4 (50 pts)

- $m_i = M$

Subtask 5 (20 bonus pts)

- No other constraints, i.e., it is possible that $m_i < M$.

Sample Test Cases

Sample Input 1 (basic)

```
5 6 10
20 15 10 10 0
1 1
1 4
1 4
2
3 16
4 4 10
```

Sample Output 1

```
3 2
5

0
0
0
2 0 5
0
```

Explanation for Sample 1

The initial list of number(power), ordered by ranks, is: 1(20) 2(15) 3(10) 4(10) 5(0). After each incident, the list becomes:

1. No changes occur because the classmate labeled 1 is already at the first place.
2. 1(20) 2(15) 4(10) 3(10) 5(0)
3. 1(20) 4(15) 2(15) 3(10) 5(0)
4. 1(24) 4(18) 2(17) 3(11) 5(0)
5. No changes occur.
6. No changes occur.

Sample Input 2 (basic)

```
8 11 2
52 43 41 41 26 20 20 18
3 41
2
3 100
1 5
1 6
2
1 6
2
4 6 2
1 6
4 6 2
```

Sample Output 2

```
4 4
0 0
24
4

0
0
0
0
1 16
3 23 1 3
0
0
```

Sample Input 3 (bonus)

```
3 7 20
487 6 3
4 3 19
1 3
4 3 1
1 3
4 3 1
1 3
4 3 8
```

Sample Output 3

```
0
3
481
484

0
0
2 3 481
```

Hints

Please be aware that the bonus subtask is intended as an optional extra, meaning that solving it is not obligatory. However, if you are keen on tackling it, the data structure introduced in Problem 3 could serve as one potential tool, alongside others that you will learn later in this semester.