A Difficult(y) Choice (books)

Immortal glory goes to those who win a medal at BOI. As you are keen on being one of them, this is your way to go*: training, training, training!

As a first step in your training program, you decided to buy some computer science books. Luckily, your local book store has a special offer with a hefty discount when you buy exactly K books.

Now you are to select the K books to buy from the set of N computer science books (numbered 1 to N) offered in the book store. Your key selection factor is difficulty: Each book i has an individual (and fully objective) difficulty x_i, and the *total difficulty* of a set of books is the sum of their individual difficulties. You don't want the selected books to be too easy (then you wouldn't learn enough to win that precious medal) or too difficult (then you wouldn't understand them before the contest starts). To be precise, you want the total difficulty of the selected books to be at least A, but not more than $2A$.

Judging the actual difficulty of a book requires you to skim through it, but the store owner won't be happy if you read many books without buying them. She allows you to skim through at most S books. Fortunately, she also tells you that the books are sorted by increasing difficulty.

Write a program that assists you in deciding on which books to skim through, and in the end tells you which books to buy.

Communication

This is a communication task. You must implement the function **void** *solve*(**int** N, **int** K, **long long** A, **int** S) where N, K, A and S are as described above. The individual difficulties $x_1 < x_2 < \cdots < x_N$ are initially kept hidden from your program. For each testcase, this function is called exactly once. In the function, you can use the following other functions provided by the grader:

- **long long** *skim*(**int** i) skims through the i-th book and returns its difficulty x_i ($1 \le i \le N$).
- **void** *answer*(**vector**⟨**int**⟩ v) buys your selection of books. This function has to be called with $v = \{i_1, \ldots, i_K\}$ ($1 \le i_1, \ldots, i_K \le N$) where the i_j's are pairwise distinct and satisfy $A \le x_{i_1} + \cdots + x_{i_K} \le 2A$.
- **void** *impossible*() states that it is impossible to buy a set of K books with the desired difficulty.

If there exists a selection of K books with the desired difficulty, you must call *answer* exactly once. Otherwise, you must call *impossible* exactly once. Your program will be automatically terminated after a call to one of these two functions.

If any of your function calls does not match the above format, or if you call *skim* more than S times, your program will be immediately terminated and judged as **Not correct** for the respective testcase. You must not write anything to standard output, otherwise you may receive the verdict **Security violation!**.

If you use C++, you must include the file `books.h` in your source code. To test your program locally, you can link your program with `sample_grader.cpp` which can be found in the attachment for this task in CMS (see below for a description of the sample grader). The attachment also contains a sample implementation with additional explanations as `books_sample.cpp`.

If you use Python, you can find a sample implementation as `books_sample.py` in the attachment which also describes the interface for Python submissions.

* "Is it, really?," says a tiny voice in your head...

Constraints

We always have $K \le N$, $3 \le N$, $S \le 10^5$, $1 \le A$, $x_i \le 10^{17}$, and $3 \le K \le 10$.

Subtask 1 (5 points). $S = N$, $170 \le N \le 1\,000$, $K = 3$

Subtask 2 (15 points). $S = N$, $N \ge 170$

Subtask 3 (10 points). $S \ge 170$, $x_{i+1} - x_i \le A/K$ for all $1 \le i \le N - 1$

Subtask 4 (15 points). $S \ge 170$, $x_{i+1} - x_i \le A$ for all $1 \le i \le N - 1$

Subtask 5 (15 points). $S \ge 170$

Subtask 6 (20 points). $S \ge 40$, $x_{i+1} - x_i \le A$ for all $1 \le i \le N - 1$

Subtask 7 (20 points). $S \ge 40$

Sample Interaction

Consider a testcase with $N = 15$, $K = 3$, $A = 42$, and $S = 8$. At the beginning, the grader calls your function *solve* as *solve*(15, 3, 42, 8). Then, two possible interactions between your program and the grader could look as follows:

Your program	Return value	Explanation
skim(1) *impossible*()	1337	The first (i.e., easiest) book has difficulty 1337. Thanks to your magical intuition, you have decided that there is no valid solution. The solution is correct and is accepted.

Your program	Return value	Explanation
skim(1) *skim*(15) *answer*({11, 15, 7})	7 21	The first book has difficulty 7. The last book has difficulty 21. Since the sequence of difficulties is strictly increasing, the sequence contains all integers from 7 to 21. Any valid set of values from the sequence with sum between 42 and 84 is acceptable. You answer with the book numbers 11, 15, 7, which correspond to difficulties 17, 21, 13. The solution is correct and is accepted.

Grader

The sample grader expects on standard input two lines. The first line should contain the four integers N, K, A and S. The second line should contain a list of N integers, the sequence of difficulties $x_1\ x_2\ \ldots\ x_N$ which has to be strictly increasing. Then, the grader writes to standard output a protocol of all grader functions called by your program. At the end, the grader writes one of the following messages to standard output:

Invalid input. The input to the grader via standard input was not of the above format.

Invalid skim. The function *skim* was called with invalid parameters.

Out of books to skim. The function *skim* was called more than *S* times.

Invalid answer. The function *answer* was called with invalid parameters.

Wrong answer. The function *answer* was called with a wrong selection of books.

No answer. The function *solve* terminated without calling *answer* or *impossible*.

Impossible (not checked): *s* book(s) skimmed. None of the above cases occurred, the function *skim* was called *s* times and the function *impossible* was called. It is not checked wether this is the correct answer.

Correct: *s* book(s) skimmed. None of the above cases occurred and the function *skim* was called *s* times.

In contrast, the grader which is used for the evaluation of your submission will only output **Not correct** (for any of the above errors) or **Correct**. Both the sample grader and the grader used for evaluation will terminate your program automatically whenever one of the above errors occurs or if your program calls *answer* or *impossible*.

Limits

Time: 1 s
Memory: 512 MiB

Inside information (servers)

Why did nobody tell you that computer science books can be so *boring*?! You really do not make progress with them, and your BOI medal is getting out of reach. But wait—who says you have to win it fair and square?*

BOI officials already mentioned that task statements are kept in a secret vault that can only be opened by the chair of the Scientific Committee. So the statements are beyond your reach. However, getting access to test data beforehand sounds manageable and should be enough to give you an advantage.

Unfortunately for you, the Scientific Committee has taken measures against possible unfairness. They split up the test data into N chunks and distributed them among N servers numbered from 1 to N. Initially, server i holds data chunk i. The N servers are connected by $N - 1$ wires in such a way that any two servers are connected to each other either directly or indirectly. From time to time, two servers *share* their data, meaning that afterwards both store the same chunks of data, namely precisely those chunks that were stored by at least one of them before. *Any two servers directly connected to each other, and only these servers, share their data precisely once.*

You are given the whole sequence of share operations. To coordinate your hacking attempts, you want to know at several points, in between two share operations, how test data is currently distributed among the servers. More precisely, you query *whether a given server currently stores a given data chunk* or *how many servers currently store a given data chunk*.

Write a program that allows you to answer such questions given (a) the sequence of share operations and (b) when each query is made.

Input

The input describes the sequence of share operations and queries. The first line of input contains two integers N and K. Each of the following $N + K - 1$ lines can have one of the following forms and meanings:

S *a b* means servers a and b **S**hare all their data.

Q *a d* means you **Q**uery whether server a currently stores data chunk d.

C *d* means you query the **C**ount (number) of servers that currently store data chunk d.

$N - 1$ lines start with an S, and K lines start with either Q or C.

Output

For each query Q a d in the input, your program should output a single line containing the string yes if the a-th server stored the d-th piece of data at the time of the query, or no otherwise. For each query C d your program should output a single line containing a single integer: the number of servers storing data chunk d at the time of the query. Of course, the answer to each query only depends on the share operations that happened before the query.

Constraints

We always have $1 \le N, K \le 120\,000$.

* Well, *we* do, and your team leaders do too, but never mind.

Subtask 1 (5 points). $N \leq 4\,000$

Subtask 2 (5 points). Server 1 is directly connected to servers $2, 3, \ldots, N$.

Subtask 3 (10 points). Servers A and B are directly connected to each other if and only if $|A - B| = 1$.

Subtask 4 (20 points). Servers A and B, with $A < B$, are directly connected to each other if and only if $2A = B$ or $2A + 1 = B$.

Subtask 5 (25 points). Any server is directly connected to at most 5 other servers.

Subtask 6 (35 points). No further constraints.

Moreover, the following holds: In each subtask you receive 50% of the points awarded for the respective subtask if you solve all testcases in which you never query how many servers store a certain data chunk (i.e. no input line starts with C). Inside CMS this is shown as "Group 1" of the corresponding subtask.

Samples

Input	Output
6 9 S 1 2 S 1 3 S 3 4 Q 5 1 S 4 5 S 1 6 Q 5 1 Q 1 5 C 1 C 2 C 3 C 4 C 5 C 6	no yes no 6 6 5 3 2 2
4 4 S 1 2 S 1 3 S 3 4 Q 2 1 Q 2 2 Q 2 3 Q 2 4	yes yes no no

Limits

Time: 2 s
Memory: 512 MiB

From Hacks to Snitches (watchmen)

As one can't make a living from the prizes awarded at computer science competitions, you have decided to get into the art business—or, more precisely, into a certain museum you have chosen for your debut as an art thief. Unfortunately, this museum is guarded quite well: there are K watchmen patrolling the building, each touring on his own simple closed path through the museum.

To coordinate your operation, you use a map of the area which describes the museum as a set of M corridors connecting N corners, numbered from 1 to N. You start at corner 1, whereas your target—a valuable exhibit—is located at corner N. Of course, one can reach any corner of the museum from any other corner, but you are not sure whether this is possible *without getting noticed*, that is, without ever being at the same corner as a watchman and without passing a watchman in a corridor.

Fortunately, you got your hands on the guard roster. So you know for each watchman where he is located at the beginning and which route he is taking. Each minute, a watchman moves from his current position to the next corner on his route, and you can either stay at your current position or move to an adjacent corner. You observed that *no two of the watchmen's routes intersect* and that neither your starting position nor your target is contained in any of them.

Write a program that uses this information to either calculate the minimum amount of time in minutes you need to safely reach your target* without being noticed† or to decide that this is impossible.

Input

The first line of input contains the integers N and M described above. The following M lines each contain two integers u and v ($1 \le u, v \le N$, $u \ne v$), meaning that there is a corridor directly connecting corners u and v. It is guaranteed that at most one corridor directly connects any two given corners.

The next line contains the single integer K described above. Then K lines follow; the i^{th} of these lines contains a sequence of integers, describing the route of the i^{th} watchman as follows: The first integer ℓ_i gives the number of distinct corners on the route. Then ℓ_i pairwise distinct integers $v_1, \ldots, v_{\ell_i}$ specify the sequence of corners the watchman passes on his route. More precisely, the watchman starts at corner v_1, in one minute he will be at v_2, and so on; after ℓ_i minutes he will be at v_1 again.

Output

Your program should output a single line. This should consist of an integer, the minimum amount of time in minutes you need to safely reach your target, or the string `impossible` if there is no way to achieve this.

Constraints

We always have $1 \le N \le 250\,000$, $1 \le M \le 3\,000\,000$, $3 \le \ell_i \le 1\,500$, and $\ell_1 + \cdots + \ell_K \le 2\,750$.

Subtask 1 (5 points). $N, M \le 100\,000$, $K = 1$, $\ell_1 \le 125$

---

* Once you reach the exhibit, you will open a window and leave the museum by means of the fancy wingsuit you won in your national computer science competition, so there is no need to plan a safe route back.

† *Of course*, you are a gentleman thief who would never lay a finger on the watchmen!

Subtask 2 (10 points). $N, M \leq 100\,000$, $\ell_1 + \cdots + \ell_K \leq 125$ and no corridor connects the routes of two distinct watchmen.

Subtask 3 (10 points). $\ell_i \leq 200$, $\ell_1 + \cdots + \ell_K \leq 350$ and no corridor connects the routes of two distinct watchmen.

Subtask 4 (10 points). No corridor connects the routes of two distinct watchmen.

Subtask 5 (25 points). $\ell_1 + \cdots + \ell_K \leq 125$

Subtask 6 (20 points). $\ell_i \leq 200$, $\ell_1 + \cdots + \ell_K \leq 350$

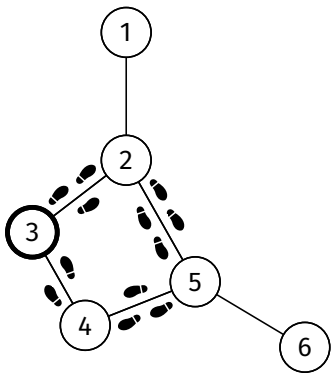
Subtask 7 (20 points). No further constraints.

Samples

Input	Output
6 6 1 2 2 3 3 4 4 5 5 2 5 6 1 4 3 2 5 4	4
6 6 1 2 2 3 3 4 4 5 5 2 5 6 1 4 4 5 2 3	5

Input	Output
11 13 1 2 2 3 3 4 4 2 3 5 5 6 6 7 7 5 6 8 8 9 9 10 10 8 9 11 3 3 4 2 3 3 7 6 5 3 10 8 9	impossible

The following picture corresponds to the situation of the first sample case above:



Here, the corner where the watchman is located at the beginning is shown in bold and his route is indicated by shoe prints. An optimal route for you would be the following: you wait at your starting location (corner 1) for one minute, and then go to corners 2, 5, and finally 6 without further delay.

The second sample has the same museum layout, but the starting location and direction of the watchman differ. A possible optimal route: go from 1 to 2, 3, 4, 5, and finally 6.

Limits

Time: 4 s
Memory: 512 MiB