DNA dictionary

Time limit:	1 second
Memory limit:	256 megabytes

To help in your research, Bakhyt has given you a DNA dictionary - a data structure that stores N DNA-strings. DNA-strings are strings of length K consisting only of uppercase letters 'A', 'C', 'G', and 'T'. Unfortunately, this dictionary has a weird interface, and you need to find a way to use it properly.

This dictionary allows you to access its data using templates. A template is a string of length K that can have three types of characters - uppercase letters ACGT, question marks and underscores. For any given template the dictionary will check all of its DNA-strings whether they match the template and tell the information about the strings that do. A string S matches the template T if there is a match at every position i ($1 \le i \le K$) except the positions when T_i is an underscore. Match at a position is determined according to the following rules:

- `'A', 'C', 'G', 'T'` — if T_i is one of the letters, this position is a match only if S_i is equal to T_i
- `'?'` — if T_i is a question mark, this position is a match regardless of S

The dictionary has only one function

- *get_min_max(T)* — Find the lexicographically minimal and maximal strings that match the given template T

An underscore in a template means that this position will be **completely ignored** during the check. The strings returned by *get_min_max(T)* will also have underscores at the same positions that T does.

You need to find a way to determine for any given DNA-string the string from the dictionary that would stay right after it in alphabetical order (or that is equal to it).

Interaction Protocol

YOUR SUBMISSIONS MUST NOT READ FROM THE STANDARD INPUT, PRINT TO THE STANDARD OUTPUT OR INTERACT WITH ANY OTHER FILE.

Your task is to implement the following function:

```
string find_next(string P)
```

- P is a DNA-string of length K
- The function must return the lexicographically minimal string from the dictionary that is lexicographically greater or equal to P.
- If no such string exists the function must return an empty string.

You can call the following function

```
pair <string, string> get_min_max(string T):
```

- T is a template string of length K
- The function will return the pair of lexicographically minimal and maximal strings in the dictionary that match the template T, with underscores at the same positions as in T.
- If no strings match the template, the function will return a pair of empty strings.

You can make no more than 2500 function calls in total in each test case. If any of the above conditions are violated, your program will get the **Wrong Answer** verdict. Otherwise, your program will get the **Accepted** verdict and your score will be calculated based on the total number of calls of the functions *get_min_max* and the total number of questions marks in the parameters of the function across all calls (Refer to the section "Scoring").

Scoring

In this task, the grader is NOT adaptive. It means that the content of the dictionary is fixed at the beginning of the run and does not depend on calls from your program.

This task consists of two subtasks:

1. (10 points) $K = 4$.
2. (90 points) $K = 256, N \leq 2 \times K$ In this subtask, your score is calculated in the following manner. Let q be the total number of calls of the function *get_min_max* and let m be the total number of question marks used in the parameters of *get_min_max* across all calls. Then your score is calculated in the following manner.
   - If $q + m \geq 2500$, your score is 0.
   - If $q + m \leq 550$, your score is 90.
   - If $550 < q + m < 2500$, your score is calculated according to the formula $\frac{2500-(q+m)}{2500-550} \cdot 90$.

   Note that your score for this subtask is equal tp the minimum score among the scores on all of its test cases.

Note

The sample grader reads the input in the following format:

- Line 1: N, K
- Line 2: $S_1, S_2, ..., S_n$

YOU CAN DOWNLOAD `dnadict.zip` in the system that contains examples for languages Java and C++11.

All the examples of calling the functions can be found above.

`dnadict.zip` contains examples of solutions for each language.

For the solutions in `Java` language, file and class name have to be named as `Dnadict.java` and `Dnadict` respectively.

KazHackStan

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Aisultan and Batyr decided to participate in KazHackStan conference. They took a computer network problem. There are given N different computers and $N-1$ bidirectional connections, two connected computers called neighbours. Computers numbered from 1 to N. It's guaranteed that there's a way to send data via connections from any computer to any other.

Sometimes some computers can be infected with viruses, that transfers to neighbour computers. Each virus in one second will infect all neighbour computers. There are given Q queries. Each of the queries can be one of the two types:

1. Appearance of new virus: will be given computer number v and time t when the virus appears, the ID of the new virus will be the minimum positive integer number that's not used before as virus ID.
2. For given l, r, t they need to calculate the number of different computers that will be infected with all viruses with ID in a range from l to r (inclusively) by time t. It's guaranteed that all viruses from l up to r exists.

In all queries t **not necessarily sorted**. For better understanding see the explanation of the sample test.

Aisultan and Batyr couldn't solve this problem. Help them with this problem.

Input

First line of the input contains two positive integer numbers N, Q ($1 \le N, Q \le 10^5$) — the number of computers in network and the number of queries. Each of the next $N-1$ lines contains two integers u_i, v_i ($1 \le u_i, v_i \le N$, $u_i \ne v_i$) — i-th connection. In the next Q lines given description of queries. Query starts with integer *type*. If $type = 1$, then given two integers v, t ($1 \le v \le n$, $1 \le t \le 10^9$) — computer ID that will infect by new virus and the time when this appears, the ID of the new virus will be the minimum positive integer number that's not used before as virus ID. If $type = 2$, then given three integers l, r, t ($1 \le l \le r$, $1 \le t \le 10^9$) — the start and end range of virus IDs and the time point at which the number of infected computers must be calculated. It's guaranteed that IDs from l to r exist. Also guaranteed that there're at least one query of type 2.

Output

For each query of type 2 print one integer — answer to the problem.

Scoring

This problem contains 8 subtasks:

1. $1 \le N, Q \le 100$. Worth 8 points.
2. $1 \le N, Q \le 1000$. Worth 7 points.
3. $1 \le N, Q \le 10^4$, $t_i \le t_{i+1}$ for all $1 \le i \le Q-1$ — all times comes in non-decreasing order. Worth 9 points.
4. $u_i = 1$, $v_i = i+1$ for all $1 \le i \le N-1$. Worth 11 points.
5. $u_i = i$, $v_i = i+1$ for all $1 \le i \le N-1$. Worth 20 points.
6. All viruses will apear in computer 1. Worth 10 points.
7. $1 \le N, Q \le 10^4$. Worth 15 points.
8. Constraints from statement. Worth 20 points.

Example

standard input	standard output
5 5 1 2 2 3 2 4 4 5 1 1 5 2 1 1 7 1 4 5 2 1 2 6 2 1 2 1	4 1 0

Note

In first query new virus comes in computer 1 on 5th second. Minimum positive integer number that's not used in virus IDs equals to 1. First virus will infect:

- Computer 1 will infect on 5-th second
- Computer 2 will infect on 6-th second
- Computer 3 will infect on 7-th second
- Computer 4 will infect on 7-th second
- Computer 5 will infect on 8-th second

Second query: we need to calculate number of computers which are infected by virus 1 on 7th second. Total such different computers 4 with numbers: 1, 2, 3, 4.

In third query new virus comes in computer 4 on 5th second. Minimum positive integer number that's not used in virus IDs equals to 2. Second virus will infect:

- Computer 4 will infect on 5-th second
- Computer 5 will infect on 6-th second
- Computer 2 will infect on 6-th second
- Computer 1 will infect on 7-th second
- Computer 3 will infect on 7-th second

Fourth query: we need to calculate number of computers which are infected by virus 1 and virus 2 on 6th second. Total such different computers 1 with number: 2.

Fifth query: we need to calculate number of computers which are infected by virus 1 and virus 2 on 1st second. No computer infected on 1st second.

Queries

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

You're given a sequence of numbers A of length N and Q queries. Each query is described by two numbers L and R. Before each query, all elements that *don't* satisfy the condition $L \leq A_i \leq R$ are erased. If X is the remaining sequence, then you need to calculate the value: $|X_1 - X_2| + |X_2 - X_3| + \cdots + |X_{k-1} - X_k|$, where k is the number of elements in X. After processing the query, all removed elements are restored back. Note that when $k \leq 1$, answer is 0.

Input

The first line of input contains two positive integers N and Q ($1 \leq N, Q \leq 200000$) — the number of elements in the sequence and the number of queries.

The next line contains N integers $A_1, A_2, \ldots, A_N$ ($1 \leq A_i \leq 10^9$) — description of sequence A.

The next Q lines each contain two integers L and R ($1 \leq L, R \leq 10^9$) — descriptions of the queries.

Output

For each query print the answer — sum of the absolute difference between each neighbor pair in the remaining subsequence.

Scoring

This problem is made up of 8 subtasks, that meet the following constraints:

1. $N, Q \leq 1000$. Worth 7 points.
2. $a_i \leq a_{i+1}$. Worth 9 points.
3. $a_i \leq 300$. Worth 10 points.
4. $l_i \leq l_{i+1}, r_i \leq r_{i+1}$. Worth 13 points.
5. $N, Q \leq 50000$. Worth 10 points.
6. $N, Q \leq 75000$. Worth 11 points.
7. $N, Q \leq 100000$. Worth 11 points.
8. Original problem constraints. Worth 29 points.

Example

standard input	standard output
10 10 2 4 7 10 6 8 6 2 6 4 3 4 2 6 3 6 3 4 8 8 1 9 6 10 3 5 4 4 3 8	0 14 4 0 0 20 11 0 0 10