NOI '22 P3 - Count

Time limit: 6.0s **Memory limit:** 1G

In contest TL 3s, ML 2048MB.

DMOJ Configuration: 6s, ML 1GB (DMOJ only allows for a 1GB memory limit!)

THIS IS AN INTERACTIVE PROBLEM

Given a quintuple (T, I, S_V, S_E, ι) where:

- T is a rooted tree of n points $T = (V, E)$, where V is the set of points of T and E is the edge set of T. The nodes of the tree are numbered $1, 2, \ldots, n$, where the root node is numbered 1.
- I is a set, and the elements in the set are called information. There are two different special elements: the UNIT element ϵ and the ILLEGAL element $\perp$.

For general information, it has two attributes: VERTEX SET and EDGE SET. For the special case of the identity element, it only has edge set attribute, while for the illegal information, it does not have either of these two attributes.

- For information $o \in I \setminus \{\epsilon, \perp\}$ (the difference $A\ B$ of two sets A, B is defined as $A\ B = \{x \in A \mid x \notin B\}$), the VERTEX SET of o is a size two subset of V, denoted $S_V(o)$. That is, $SV(o) \subseteq V$ and $|SV(o)| = 2$.
- For information $o \in I \setminus \{\perp\}$, the EDGE SET o is a subset of E, denoted $S_E(o)$, such that $S_E(o) \subseteq E$. Define the edge set of the identity element is empty, that is, $S_E(\epsilon) = \emptyset$.
- For any edge $e \in E$ in the tree, denote $e = (u, v)$, there is an information about e, $\iota(e) \in I$, which takes its endpoints its VERTEX SET and the edge itself as its EDGE SET, that is, $S_V(\iota(e)) = u, v$, and $S_E(\iota(e)) = e$.

There are two ways that information get combined. Denote them as R and C. They have the following properties

For all $a, b \in I$, shorthand $r = R(a, b)$, $c = C(a, b)$, such that $r, c \in I$.

- Combining UNIT with any general information gives the other. That is if $a = \epsilon$, then $r = c = b$; If $b = \epsilon$, then $r = c = a$.
- Combining ILLEGAL with ANY information results in illegal information. That is, if $a = \perp$ or If $b = \perp$, then $r = c = \perp$.
- For the remaining cases, if the intersection of the EDGE SET of the two information is non-empty, or the intersection of the POINT SET of the two information has size that's not 1, the combine results in ILLEGAL. That is, if $S_E(a) \cap S_E(b) \neq \emptyset$ or $|S_V(a) \cap S_V(b)| \neq 1$, then $r = c = \perp$.
- Otherwise, the operations are specified as
-   - $S_E(r) = S_E(c) = S_E(a) \cup S_E(b)$,
-   - $S_V(r) = S_V(a)$,
-   - $S_V(c) = S_V(a) \oplus S_V(b)$.

where $\oplus$ represents the symmetric difference operation of sets, that is, $A \oplus B = (A \cup B) \setminus (A \cap B)$.

Define the on-tree distance of two points in T as the number of edges on the tree that a unique simple path traversed by two points as endpoints.

Given the scoring parameter M and q queries, each query consisting a vertex u of the tree and a non-negative integer d. Denote X to be the set of all vertices in T whose distances to u in the tree does not exceed d, and $Y = \{(a, b) \in E \mid a, b \in X\}$ to be the set of edges inside X.

It can be shown that starting from ϵ and all $L(e)$ ($e \in E$), a finite number of R, C calls produces an information o such that $o \neq \perp$ and $S_E(o) = Y$. In particular, if $d = 0$, you the output should be the UNIT element ϵ.

In each set of queries, you need to construct an information o that satisfies this requirement, subject to the limit that the sum of the calls to R and C does not exceed M.

Implementation Details

Files can be found here.

Make sure you #include `count.h` at the beginning of your program. The header file count.h implements the following:

1. Define the data type `info` corresponding to the information;
2. Define the `info` type constant `emptyinfo` corresponding to ϵ, which you can use directly in the program.
3. The following two information merging functions, which you can call directly in the program:

   ```
   info MR(info a, info b);
   info MC(info a, info b);
   ```

   The two functions return the information corresponding to $R(a, b)$ and $C(a, b)$, respectively. You need to ensure that calling $R(a, b)$ or $C(a, b)$ does not result in $\perp$, otherwise the program may behave unexpectedly.
4. A function to determine whether a information is UNIT, you can call it directly in the program:

   ```
   bool isempty(info a);
   ```

   This function returns true if and only if a is an identity element. See the reference interaction library for more implementation details.

You do not need to, and should not, implement main();

You need to implement the following functions:

```
void init(int T, int n, int q, vector<int> fa, vector<info> e, int M);
```

- T is the test point number, n is the number of points in the tree, q is the number of queries, and M is the scoring parameter for that test point.
- Both fa and e have length $n - 1$. For $0 \leq i < n - 1$, fa[i] and i + 2 are the two endpoints of the i-th edge e_i, and e_i is the info type element corresponding to $\iota(e_i)$ mentioned in the title description. The data guarantees that fa_i is

less than $i+2$.

```
info ask(int u, int d);
```

- Give a query, see the question description for the meaning of the parameters. You need to return a information that satisfies the condition of the question at the end of the function.

When testing, at each test case, the interactive library will call the `init` function exactly once, and then call the `ask` function q times. The interactive library will use a special implementation, a single info type variable will constantly consume 12 bytes of memory,

It is guaranteed that under the condition that the number of calls is satisfied and the isempty function is not called, the time required for the interactive library to run in the final test does not exceed 0.6 seconds, and the memory consumed by the interactive library itself does not exceed 16 MiB. It is guaranteed that the final tested interactive library will take no more than 0.25 seconds to run with at most 10^8 isempty function calls.

A file named count.cpp is included in the issued files: you may use it.

Testing

The reference materials provides of two interactive libraries `grader.o` and `checker.o` are provided in this topic directory, which are linkable files generated by compiling two different interactive libraries. The implementation of the interaction library used in the final test is different from this implementation, so the solution of the contestant should not depend on the specific implementation of the interaction library, nor should it depend on the specific implementation of the info type in count.h.

You need to modify the distributed `count.h` to help with linking. Specifically, when linking the source code `count.cpp` with the program `grader.o`, you need to comment out the 5th line of the count.h code and keep the 4th line of code. Linking the `checker.o` method is similar, you need to comment out the 4th line of the `count.h` code and keep the 5th line of code. Players can modify the implementation of `count.h` by themselves to compile different programs.

After modification, the contestant can use the following command to compile the executable program in the directory of this question:

```
g++ count.cpp -c -O2 -std=c++14 -lm && g++ count.o grader.o -o count
g++ count.cpp -c -O2 -std=c++14 -lm && g++ count.o checker.o -o count
```

The first command line will compile the current count.cpp and link it with `grader.o` to generate the executable file count. The second line command will compile the current `count.cpp` and link it with `checker.o` to generate the executable file count.

The executable file count obtained by compiling the above method runs as follows:

- The executable will read data in the following format from standard input:

- The first line contains four integers id, n, q, M, which represent the test point number, the number of points in the tree, the number of queries, and the scoring parameter;
- In the second line, n - 1 integers $p_2, p_3, \ldots, p_n$ represent the parent node numbers from 2 to n, respectively. When debugging locally, you need to ensure that $\forall i \in [2, n]$, $p_i < i$;
- The next q lines contain two integers u, d each, describing a query.

After being read in, the interactive library is tested. If your program does not meet the interactive library limit, the output will return the corresponding error. Otherwise, for the linked executable, the output is as follows:

- A total of three integers $C1, C2, C3$ on one line, where: $*$ $C1$ *represents the total number of times the program calls the interactive library function in the init function;* $*$ $C2$ represents the total number of times the program calls interactive library functions during running; $*$ * $C3$ represents the maximum number of times the program calls the interactive library function in q times of ask functions.
- For the above three statistics, we only count the number of calls of the MR and MC functions, but not the number of calls of the isempty function.
- When linking different files, the checks they can perform are also different, specifically:
- `grader.o` : It does not check whether the information returned by the ask function is correct at runtime, but it can help players determine whether the interaction meets the requirements. The running time of this program is closest to the interactive library at the time of evaluation, so players can use this program to test the running speed, but the correctness of the program is not guaranteed.
- `checker.o` : It will check whether the information returned by the ask function is correct at runtime, and can also help players determine whether the interactive operation meets the requirements. At the same time, it will check whether the information returned by the ask function is correct. This program can check the correctness of the answers.

Samples

See count{1,2,3,4}.{in, ans}

Samples 2 & 3 satisfy special properties A and B (see below) respectively

NOTE:

- An uninitialized variable of type `info` is not guaranteed to be `emptyinfo`.
- Please do not try to access or modify member variables of type `info`, otherwise it will be regarded as attacking the interactive library.
- Please do not call the `MR` and `MC` functions before the init function call, otherwise undefined behavior may occur.
- You can only access the variables defined by yourself and the `info` type variables returned by the interactive library. Trying to access other spaces may result in compilation errors or runtime errors.

This question will first be subject to the same restrictions as traditional questions. For example, compilation errors will result in 0 points for the entire question, and runtime errors, exceeding the time limit, exceeding the space limit, etc. will result in 0 points for the corresponding test points, etc. In addition to the above conditions, a test point will receive a score of 0 if the program executes an illegal function call or gives an incorrect answer in a query operation. Otherwise, denote $C1$ and $C3$ as the number of times your program calls the interactive library function in the init function, and the maximum number of times your program calls the interactive library function in all q ask functions. If $C1 \leq 3 \cdot 10^7$ and $C3$ does not exceed the scoring parameter M for that test point, you will get a score for that test point, otherwise you can't get a score for that test point. Note: When calculating $C1$ and $C3$, only the number of calls of the MR and MC functions will be counted, but not the number of calls of the isempty function.

Constraints

For all test cases, $1 \le n \le 2 \times 10^5$, $1 \le q \le 10^6$, for each query, $1 \le u \le n$, $1 \le d \le n-1$.

TestCase	$n =$	$q =$	Property	$M =$
1	1000	10^4		500
2	2000	10^4		500
$3, 4$	10^5	10^6	A	5
$5, 6$	6×10^4	6×10^4	B	50
7	6×10^4	6×10^4	B	5
8	10^5	10^5	B	5
9	7500	5×10^4	C	500
10	10^4	5×10^4		500
11	1.5×10^4	5×10^4		500
12	2×10^4	5×10^4		50
13	2.5×10^4	5×10^4		5
14	3×10^4	10^5		5
15	6×10^4	10^6	D	5
16	6×10^4	10^6		5
17	8×10^4	10^6		5
18	10^5	10^6		5
19	1.5×10^5	10^6		5
20	2×10^5	10^6		1

- Property A: For all $i = 1, 2, \ldots, n-1$, the parent of $(i+1)$ is i.
- Property B: For all queries, $u = 1$.
- Property C: For all queries, $d <= 100$.
- Property D: For all queries, $d >= 1000$.