IOITC 2022 TST 2

Inversions

There is a hidden vector F, which initially has n values, $F_0, F_1, \ldots F_{n-1}$, which are a permutation of $0, 1, 2, \ldots, n-1$. You want to find the number of inversions in F, that is, the number of pairs (i, j) with $0 \leq i < j \leq n-1$ with $F_i > F_j$.

By calling some functions defined later, you can append some values to the end of F. Say, currently $F = [F_0, F_1, \ldots F_{k-1}]$. **At any instant, any value in F must be in the range $[0, 10^9]$, otherwise you get a wrong answer verdict.** You can call the following functions, each of which appends some value to F, and returns k (the index at which this new value is stored):

1. `int put(int i)`
    - This function appends i to the end of F.
2. `int compose(int i)`
    - This function appends F_{F_i} to the end of F
    - To call this function, $0 \leq i < k$ and $0 \leq F_i < k$ are necessary conditions.
3. `int add(int i, int j)`
    - This function appends $F_i + F_j$ to the end of F
    - To call this function, $0 \leq i, j < k$ is necessary.
4. `int subtract(int i, int j)`
    - This function appends $F_i - F_j$ to the end of F
    - To call this function, $0 \leq i, j < k$ is necessary.
5. `int multiply(int i, int j)`
    - This function appends $F_i \times F_j$ to the end of F
    - To call this function, $0 \leq i, j < k$ is necessary.
6. `int exclusive_or(int i, int j)`
    - This function appends $F_i \oplus F_j$ to the end of F, where $\oplus$ denotes the bitwise xor operator
    - To call this function, $0 \leq i, j < k$ is necessary.
7. `int compare(int i, int j)`
    - This function appends 1 to the end of F, if $F_i < F_j$, else it appends 0.
    - To call this function, $0 \leq i, j < k$ is necessary.

All these functions are implemented in `dummy_grader.cpp` , and you can check them for more clarity.

You need to implement the function `void solve(int n)`, which makes a total of at most Q calls to these 7 functions, such that in the end, the last value in F contains the number of inversions in the original permutation. (Note that the values appended by calling these functions don't contribute to the number of inversions).

Test Data

In all test data, $n = 3000$

Subtask 1 (25 Points):

- $Q = 4 \times 10^5$.
- There exist at max two indices $0 \leq i < n$ with $F_i \neq i$. That, is the original permutation can be sorted by selecting two indices and swapping them.

Subtask 2 (75 Points):

Here, there are no constraints on the original permutation. Also, $Q = 10^7$. Let q be the number of calls made by you. Then you get a score of $75\sqrt{\frac{4 \times 10^5}{\max(4 \times 10^5, q)}}$

Local testing

You are provided with a dummy grader with the name `dummy_grader.cpp`. You should compile your solution (assumed to be in the file `solution.cpp`) as:

```
g++ solution.cpp dummy_grader.cpp -o grader
```

Then you can run `./grader`, and give input of the form as given in the sample input:

- The first line contains n and Q
- Then next lines $F_0, F_1, \ldots F_{n-1}$.

Do NOT read anything from stdin or write something to stdout/stderr.

Limits

Time: 2 seconds

Memory: 256 MB