Ad-hoc Master II

Translated by Gemini 3

Notice

This is an interactive problem.

Background

After solving Ad-hoc Master, you have set your sights on the following problem.

Problem Description

You are given a positive integer h. Let $n = 2^h - 1$.

The interactor hides a permutation p of $1 \sim n$ and a sequence f of length n, where each element is a **positive integer** not exceeding 10^9.

There exists a **perfect binary tree** G of depth h containing n nodes, rooted at node 1. For any node u satisfying $2 \le u \le n$, its parent is node $\left\lfloor \frac{u}{2} \right\rfloor$.

In each query, you can choose two integers u and d satisfying $1 \le u \le n$ and $1 \le d \le 10^9$. The interactor will return the sum of f_v for all nodes v such that $\mathbf{dis}(p_u, v) = d$. Specifically, if there are no such nodes v, the interactor returns 0.

Here, $\mathbf{dis}(u, v)$ denotes the number of edges on the simple path between node u and node v. Specifically, $\mathbf{dis}(u, u) = 0$.

You need to find the value of $\sum_{i=1}^{n} f_i$ using no more than $5n$ queries. Furthermore, **for each integer u, you may query it at most 5 times**. There is no additional restriction on d. Your score depends on the **number of queries** used per test case.

It is NOT guaranteed that the permutation p and sequence f are fixed; the interactor may be adaptive.

Implementation Details

You should include the header file `tree.h` at the beginning of your code:

```
#include "tree.h"
```

You do not need to implement the `main` function. You need to implement the following function:

```cpp
long long solve(int subtask, int h);
```

- `subtask`: The ID of the current subtask.
- h: The height of the binary tree.
- This function should return the value of $\sum_{i=1}^{n} f_i$.
- **This function may be called multiple times by the interactor for each test point.**

You can interact with the grader by calling the following function:

```cpp
long long ask(int u, int d);
```

- u: The center node of the query. You must guarantee $1 \leq u \leq n$.
- d: The distance constraint. You must guarantee $1 \leq d \leq 10^9$.
- This function returns the sum of f values of all nodes whose distance to p_u is exactly d.

The problem guarantees that within the specified operation limits, the interactor runs in no more than **1** second. The interactor uses a fixed amount of memory, not exceeding **128** MiB.

Local Testing

The `grader.cpp` provided in the problem directory is a reference implementation. The final grader used for evaluation differs from this reference, so your solution should not rely on the specific implementation of the grader.

You can compile an executable in the problem directory using the following command:

```
g++ grader.cpp tree.cpp -o tree -O2 -std=c++14
```

You can also add the `-DDEBUG` option to enable debug mode:

```
g++ grader.cpp tree.cpp -o tree -O2 -std=c++14 -DDEBUG
```

For the compiled executable:

- It reads data from **standard input** in the following format:
  - The first line contains two integers $subtask, T$, representing the subtask ID and the number of test cases.
  - Each test case contains three lines:
    - The first line contains a positive integer h.

    - The second line contains $n = 2^h - 1$ space-separated integers, representing the hidden permutation.
    - The third line contains $n = 2^h - 1$ space-separated integers, representing the values of sequence f.

- For each test case, the grader calls the function `tree` once. After all tests are completed, the grader outputs the following information to the **standard error stream**:
  - The first line contains your score information.
  - The second line contains a description of the test results given by the interactor.
- If debug mode is enabled, the grader will print detailed information about each query to the standard error stream. **Ensure that input test data is small when using debug mode to avoid excessive output.**

Interaction Example

Suppose $h = 2$, $n = 3$, hidden permutation $p = [2, 1, 3]$, node weights $f = [11, 45, 14]$. A valid interaction process is as follows:

Contestant	Interactor	Explanation
	Calls `tree(1, 2)`	Test starts
Calls `ask(1, 1)`	Returns **11**	Node at dist **1** from $p_1 = 2$ is **1**. Sum $= 11$.
Calls `ask(2, 1)`	Returns **59**	Nodes at dist **1** from $p_2 = 1$ are **2, 3**. Sum $= 45 + 14 = 59$.
Calls `ask(3, 1)`	Returns **11**	Node at dist **1** from $p_3 = 3$ is **1**. Sum $= 11$.
Returns **70**	Prints Result	Interaction ends, result correct.

Data Constraints

For all test data: $2 \le h \le 18$, number of test cases $1 \le T \le 10^5$, sum of n over all data $\sum n \le 10^6$.

There are **4** subtasks.

Subtask	Score	Special Properties
1	6	$h = 2$
2	13	$h \in [3, 4]$
3	21	$h \in [5, 8]$
4	60	$h \in [5, 18]$

Scoring

Note:

- Contestants should not attempt to access internal information of the interactor by illegal means. Such behavior will be considered cheating.
- **The final grader is adaptive: without contradicting previous `ask` results, it may dynamically adjust p and f.**

Your program will first be subject to standard limitations (compilation errors, TLE, MLE, etc.).

In each `solve` call, the total number of queries q must satisfy $q \leq 5n$, and the number of queries for a single u is at most **5**. Otherwise, you get **0** points.

If the answer is correct, the score for a subtask is the minimum score across all its test cases. If a program uses q queries and the subtask value is x, the score is $x \cdot f(q)$, where $f(q)$ is the maximum percentage from the table below satisfying the condition:

Condition	Ratio	Condition	Ratio
$q \leq \frac{5n+9}{4}$	100%	$q \leq \frac{7n+11}{4}$	52%
$q \leq \frac{5n+13}{4}$	99%	$q \leq \frac{15n+23}{8}$	40%
$q \leq \frac{5n+17}{4}$	98%	$q \leq 2n+3$	32%
$q \leq \frac{11n+19}{8}$	88%	$q \leq 3n+5$	24%
$q \leq \frac{3n+5}{2}$	76%	$q \leq 4n+5$	16%
$q \leq \frac{13n+21}{8}$	64%	$q \leq 5n$	8%

Time Limit: **3** s (Interactor uses at most **1** s) **Memory Limit:** **1024** MB (Interactor uses at most **128** MB)