Re-Tree-Val

Project
Data Structures
Algorithms
Due date: 7 May, 2020

Problem Statement: Nostalgic huh...!
Disclaimer: Keep your head together while reading this problem.
Lets play the game of DARE and TRUTH. For each test case, OJ guesses a number X and you have to DARE the OJ in order to get hints about the number X. Once you find that X, you have to return TRUTH value of X to OJ. TRUTH value of X = phi(X), where phi is Euler's Totient Function, it returns the number of numbers between 1 and X, both inclusive, which are relatively prime with X. You have to answer Q such test cases (All test cases are independent).
OJ randomly picks X from a perfect binary tree of height I, and the nodes are 1 indexed. The root is 1 and the node with number R has two children 2*R and 2*R+1. You can ask special queries to OJ about your guess i and it will say $X = i$ or if not, then whether X lies on the left subtree or right subtree. For each test case, you will be allowed only I such special queries.
Simple right?
It does the following computations to answer your special query.
First, OJ will select V based on following condition:

$$V = \begin{cases} 1 & if \text{ X = i} \\ rand(2, 2^{I-1}) & \text{if X is on proper left subtree of i, both inclusive} \\ rand(2^{I-1}+1, 2^{I}-1) & \text{if X is on proper right subtree of i, both inclusive} \\ 0 & \text{otherwise} \end{cases}$$

Second, it picks a random number D such that $D > V$ and $gcd(V, D) = 1$. Note that D is not necessarily prime.
Third, it finds U such that $UV = 1$ modulo D.
Forth, it picks a random value P, such that $1 \leq P \leq 82024$.
Fifth, it computes T = $(U * P)\%(10^9 + 7)$.
Sixth, it computes S = $(\prod_{i=0}^{q} T_i)\%(10^9 + 7)$, q is the current special query number and T_i is the answer it gave you for the i^{th} special query. T_0 will be given at the beginning of each test case.
Finally, it returns three variables: S, P_i, D, where is S is the product as computed above, P_i is the P^{th} odd prime and D as mentioned above. Once you find the X which OJ has guessed, return the phi(X).

Note
BE EXTRA SURE OF THE INTERACTION. IT IS CASE SENSITIVE AND SPACE SENSITIVE. DO NOT PRINT ANY EXTRA LINES AND DO NOT PRINT ANY OTHER DEBUGGING STATEMENTS.
phi(1) is 1.

Input
First fixed line will contain Q, the number of times you have to play the guessing game. For each turn of play, 2 numbers will be given, I and T_0, where I is the height of the tree and T_0 is useful in computing the value of S.

Interaction
Query "DARE i"(without quotes), and the OJ will return three space separated integers S P_i D. Note that you can directly read them as an integer. NOTE: AFTER USING cout, ALWAYS USE fflush(NULL); statement in C++. If you don't use that, there will be issues with input.

Output
Once you find the correct PIN, output "TRUTH phi(X)"(without quotes).

Constraints
$1 \leq Q \leq 10000$
$1 \leq I \leq 20$
special Dare queries $\leq I$
$0 \leq V \leq 2^I - 1$, as mentioned above
$V < D \leq 10^9$
$gcd(V, D) = 1$
$V < D$
$1 \leq P \leq 82024$ that means $3 \leq P_i \leq 1048573$
$1 \leq T < 10^9 + 7$

Time Limit: 10 sec
Memory Limit: 256 MB

Sample Test Case

Guessed X for fixed height I = 3	1 2 7
Interaction Sample 1(on the above arrays)	
3 3 7485 498635730 836189 16809 3 489574 413076712 456223 984943663 708786788 692581 470211274 3 1 441722805 330037 458777938 805928263 644141 115438176 24854108 698051 74243044	DARE 1 TRUTH 1 DARE 1 DARE 2 TRUTH 1 DARE 1 DARE 3 DARE 7 TRUTH 6

Explanation
On decoding the first query of the first round, you get V = 1, implying that your guess is correct.
On decoding the first query of the second round, you get V = 4, implying that correct X is on proper left subtree. On decoding the second query, you get V = 1, implying that correct X is 2.
On decoding the first query of the second round, you get V = 7, implying that correct X is on proper right subtree. On decoding the second query, you get V = 5, implying that correct X is again on proper right subtree. On decoding the third query, you get V = 1 implying that correct X is 7.
The output on the left side is what you might not get, since the numbers are randomly generated, but if you plugin this inputs in your correct code, it should return the output correctly.

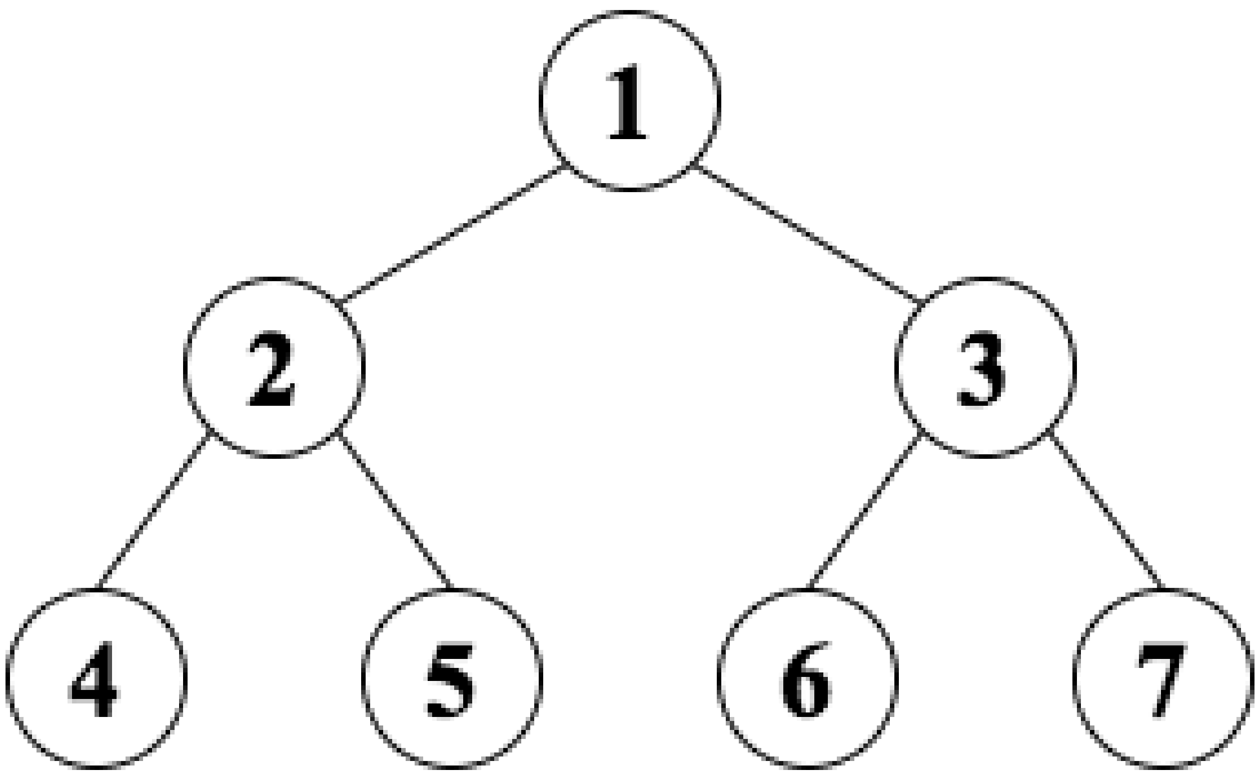

Figure 1: A perfect binary tree with height 3.