4 Digital Circuit

Written by: **Prabowo Djonatan** (Indonesia), Garena Singapore
Prepared by: Prabowo Djonatan
Solutions, review, and other problem preparations by: Jonathan Irvin Gunawan, Muhammad Ayaz Dzulfikar
Analysis author: Prabowo Djonatan

For ease of discussion, we will represent the circuit as a tree, where the gates are the nodes. Furthermore, the tree is rooted at 0, and the array P represents the parent of each node. We will also refer threshold gates as internal nodes, and source gates as leaves.

4.1 Subtask 1

In this subtask, $N = 1$, $M \leq 1000$, $Q \leq 5$.

The tree is star-shaped. For a node with c children, if the state of exactly k children is 1, then there are exactly k threshold parameters that can be assigned such that the node has state 1, which are $1, 2, \ldots k$. Since the root has all the leaves as its children, it is sufficient to output the sum of all A after each update.

Time complexity: $O(QM)$

4.2 Subtask 2

In this subtask, $N, M \leq 1000$, $Q \leq 5$, and the tree is a full binary tree.

Define $\texttt{dp}_0(u)$ and $\texttt{dp}_1(u)$ as the number of ways to make node u has value 0 and 1 respectively, by assigning parameters to the subtree of u.

We have the following transitions:

- $\texttt{dp}_0(u) = \texttt{dp}_0(l)\texttt{dp}_1(r) + \texttt{dp}_1(l)\texttt{dp}_0(r) + 2\texttt{dp}_0(l)\texttt{dp}_0(r)$
- $\texttt{dp}_1(u) = \texttt{dp}_0(l)\texttt{dp}_1(r) + \texttt{dp}_1(l)\texttt{dp}_0(r) + 2\texttt{dp}_1(l)\texttt{dp}_1(r)$

where l and r are the children of u.

After each update, recompute the whole `dp`.

Time complexity: $O(Q(N + M))$

4.3 Subtask 3

In this subtask, $N, M \leq 1000$, $Q \leq 5$.

Use the same definition of `dp` from subtask 2, and modify the transition to a knapsack-like DP to find the number of ways such that exactly k of its children have state 1. Alternatively, we can also use Left-Child Right-Sibling tree DP technique.

Time complexity: $O(Q(N + M)^2)$

4.4 Subtask 4

In this subtask, the tree is a perfect binary tree, and each update toggles only a single leaf.

A perfect binary tree has a height of $O(\log(N+M))$. Using the same DP formulation from subtask 2, we notice that the only DP values that are to be updated are those in the path from that leaf to the root.

Time complexity: $O(N+M+Q\log(N+M))$

4.5 Subtask 5

In this subtask, the tree is a perfect binary tree.

Suppose all leaves contained in the subtree of node u are updated. In that case, we can swap the value of $\texttt{dp}_0(u)$ and $\texttt{dp}_1(u)$, then lazily update its subtree. Thus, we can treat the whole perfect binary tree as a segment tree, and do the updates accordingly.

Time complexity: $O(N+M+Q\log(N+M))$

4.6 Subtask 6

In this subtask, the tree is a full binary tree (each node has exactly two children).

For each node v, note that the value of $\texttt{dp}_0(v)+\texttt{dp}_1(v)$ is fixed regardless of the values of A, because it covers all possible states of the node v. In fact, the constant equals to $2^{\text{size}(v)}$ (where size(v) is the size of the subtree of v) because that is all the possible number of ways to assign parameters to the subtree of v.

Let's rearrange the `dp` formula from subtask 2:

$$\begin{aligned}\texttt{dp}_1(u) &= \texttt{dp}_0(l)\texttt{dp}_1(r)+\texttt{dp}_1(l)\texttt{dp}_0(r)+2\texttt{dp}_1(l)\texttt{dp}_1(r)\\ &= \texttt{dp}_1(l)(\texttt{dp}_0(r)+\texttt{dp}_1(r))+\texttt{dp}_1(r)(\texttt{dp}_0(l)+\texttt{dp}_1(l))\\ &= \texttt{dp}_1(l)(2^{\text{size}(r)})+\texttt{dp}_1(r)(2^{\text{size}(l)})\end{aligned}$$

Note that similar rearrangement can be done for $\texttt{dp}_0(u)$ as well.

Another way to interpret the formula rearrangement is that: the number of ways to assign a parameter to a threshold gate is isomorphic to a multiplexer (i.e. the threshold gate can be thought as an operator that takes the value from one of its two children). Using this interpretation, we can count, for each leaf with state 1, how many ways for it to reach the root.

The number of ways for a leaf v to reach the node is $2^{N-\text{depth}(v)}$. That means when the state is 0, the leaf will contribute 0 towards the answer, and $2^{N-\text{depth}(v)}$ otherwise. The range update can be done using a segment tree that is similar to subtask 5.

Time complexity: $O(N+M+Q\log M)$

4.7 Subtask 7

We can extend the interpretation from subtask 6 to this subtask. That is, the threshold gate can be seen as an operator that takes a value from one of its children.

Therefore, for each leaf, we can mark all the paths from it to the root, and then compute the product of the number of children of all unmarked internal nodes in $O(N + M)$. The product will be the contribution value of this leaf. The update can again be performed using the help of a segment tree.

Time complexity: $O(M(N + M) + Q \log M)$

4.8 Subtask 8

Continuing from subtask 7, we can optimize the computation of the contribution value of each node. First, for each node v, we compute the product of the number of children for all the internal nodes in the subtree of v. Let the values be $subtree(v)$. This can be done in $O(N)$.

Next, we can compute the contribution of all leaves in a single DFS. When traversing to a child of a node, we need to multiply all the *subtree* values from all the other children to the final product. To do this quickly, we can order the children $c_1, c_2, \ldots$ and for each child c, we calculate the prefix product and suffix product of $subtree(c)$. To traverse to child c_i, we multiply the product of the prefix product of $subtree(c_{i-1})$ and the suffix product of $subtree(c_{i+1})$.

The update can be done similarly as subtask 7.

Time complexity: $O(N + M + Q \log M)$