AND/OR Tree Evaluation

1 AND/OR Trees

In a **game tree**, a node is either a min or a max, alternating per level. The values of the leaves are given explicitly. The value of a min node is the minimum of the values of its children, and vice versa for max nodes. In the special case where each leaf is either a 0 or a 1, the game tree can be represented as an AND/OR tree, where ANDs correspond to mins and ORs correspond to maxes.

Our goal is to evaluate the root of the tree while querying as few leaves as possible. Deterministically, we can be forced to query all leaves to get the value of a subtree. So, we'd like to randomize.

Good case for AND evaluation: 0,0

Medium case for AND evaluation: 0,1 or 1,0

Bad case for AND evaluation: 1,1

Good case for OR evaluation: 1,1

Medium case for OR evaluation: 0,1 or 1,0

Bad case for OR evaluation: 0,0

Note that the bad case for the AND is the good case for the OR, and vice versa. Thus, when randomizing, it is impossible for us to keep running into the worst-case scenario, and we'll use this fact in our analysis.

2 Snir's Algorithm

Algorithm 1 Snir's Algorithm

```
Pick either subtree uniformly at random
Evaluate that subtree recursively
if the value of the current node is now determined by the above cases then
    Return the value
else
    Evaluate the other subtree
end if
```

Snir's algorithm is essentially a randomized DFS that will give us better results in our analysis. Let $X^{0/1}_{\text{AND/OR}}(h)$ be the cost of evaluating a tree of height h whose root is AND/OR and evaluates to 0/1. We have the following system of equations by the above cases:

$$\mathbb{E}[X^0_{\text{AND}}(h)] \leq 2\mathbb{E}[X^1_{\text{OR}}(h-1)]$$

$$\mathbb{E}[X^0_{\text{AND}}(h)] \leq \frac{1}{2}(\mathbb{E}[X^0_{\text{OR}}(h-1)] + \frac{1}{2}\mathbb{E}[X^1_{\text{OR}}(h-1) + X^0_{\text{OR}}(h-1)])$$

$$\mathbb{E}[X^0_{\mathrm{OR}}(h)] \leq 2\mathbb{E}[X^0_{\mathrm{AND}}(h-1)]$$

$$\mathbb{E}[X^1_{\mathrm{OR}}(h)] \leq \frac{1}{2}(\mathbb{E}[X^1_{\mathrm{AND}}(h-1)] + \frac{1}{2}\mathbb{E}[X^1_{\mathrm{AND}}(h-1) + X^0_{\mathrm{AND}}(h-1)])$$

To make the analysis a little simpler, we'll define $Y_{\mathrm{AND/OR}}(h)$ to be the maximum cost for evaluating an AND/OR node of height h:

$$Y_{\mathrm{AND}}(h) = \max(\mathbb{E}[X^0_{\mathrm{AND}}(h)], \mathbb{E}[X^1_{\mathrm{AND}}(h)])$$

$$Y_{\mathrm{OR}}(h) = \max(\mathbb{E}[X^0_{\mathrm{OR}}(h)], \mathbb{E}[X^1_{\mathrm{OR}}(h)])$$

To get values for these new variables, we'll express the cost of each node in terms of the cost of the same type of node two levels down in the tree.

$$\mathbb{E}[X^0_{\mathrm{OR}}(h)] \leq \mathbb{E}[X^0_{\mathrm{OR}}(h-2)] + (\mathbb{E}[X^1_{\mathrm{OR}}(h-2)] + \mathbb{E}[X^0_{\mathrm{OR}}(h-2)])$$

$$\leq 3Y_{\mathrm{OR}}(h-2)$$

$$\mathbb{E}[X^1_{\mathrm{OR}}(h)] \leq \mathbb{E}[X^1_{\mathrm{OR}}(h-2)] + (\mathbb{E}[X^1_{\mathrm{OR}}(h-2)] + (\frac{1}{2}\mathbb{E}[X^0_{\mathrm{OR}}(h-2)] + \frac{1}{4}\mathbb{E}[X^1_{\mathrm{OR}}(h-2)]))$$

$$\leq \frac{11}{4}Y_{\mathrm{OR}}(h-2)$$

Thus:

$$Y_{\mathrm{OR}}(h) \leq 3Y_{\mathrm{OR}}(h-2)$$

Analogously:

$$Y_{\mathrm{AND}}(h) \leq 3Y_{\mathrm{AND}}(h-2)$$

By induction on h:

$$Y_{\mathrm{OR}}(h) \leq 3^{\frac{h}{2}}, Y_{\mathrm{AND}}(h) \leq 3^{\frac{h}{2}}$$

If the tree has n leaves, then $h = \log(n)$, so the number of evaluated leaves L results in expectation as:

$$\mathbb{E}[L] \leq 3^{\frac{\log(n)}{2}} = n^{\frac{\log(3)}{2}} \leq n^{0.793}$$

3 A Lower Bound on Randomized AND/OR Tree Evaluation Algorithms

Now that we have an upper bound on the number of leaves we need to check, we can use Yao's Principle to obtain a lower bound on all randomized AND/OR tree evaluation algorithms. We want to design an input distribution (i.e., the truth values of the leaves) such that each possible deterministic algorithm needs to query a relatively large amount of leaves.

The simplest distribution would be to make every leaf 1 with probability p. But, we'd like p to be such that, at all layers from the bottom, the nodes are true with probability p. So, $\frac{1}{2}$ doesn't work.

Note that an AND/OR tree of even height and (without loss of generality) an AND at the root is equivalent to a NOR tree. We know that each NOR node is true with probability $(1-p)^2$, which we want equal to p.

$$(1-p)^2 = p$$

$$p^2 - 3p + 1 = 0$$

$$p = \frac{3-\sqrt{5}}{2}$$

Now, we need the best deterministic algorithm versus this input. This algorithm is DFS: at any node, recursively evaluate its left subtree, and if needed, evaluate the right subtree. The proof of optimality requires some care so we'll skip it here. Let $T(h)$ be the expected number of steps to evaluate a tree of height h. Then, using this algorithm:

$$T(h) = T(h-1) + (1-p)T(h-1)$$

$$= (2-p)T(h-1)$$

By induction,

$$= (2-p)^h$$

$$= n^{\log(\frac{1+\sqrt{5}}{2})}$$

$$= n^{\log \phi}$$

$$\approx n^{0.694}$$

So by Yao's Principle, every randomized algorithm must evaluate $\geq n^{0.694}$ leaves in expectation. Snir's Algorithm is optimal; this lower bound is not. To improve it, we have to correlate the leaf distributions top-down to ensure that no OR node ever has two children evaluate to 1 and no AND node ever has two children evaluate to 0. The resulting correlations make it more difficult to prove that DFS is the optimal deterministic algorithm, but in the end we'd get the desired bound of $n^{0.793}$.