2.2 Game tree evaluation: randomized algorithms

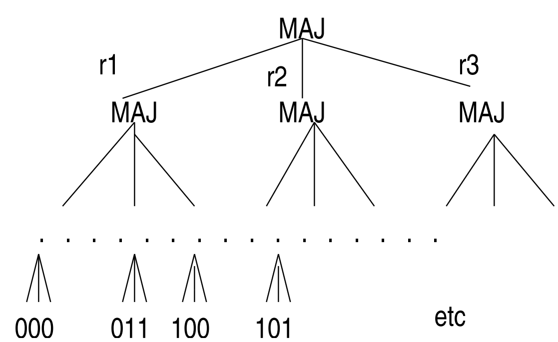
Let us now focus on game trees with a uniform structure. The most canonical example is alternating levels of binary AND and OR gates. This can be simplified with the identity $\mathrm{OR}(x,y) = \neg\mathrm{AND}(\neg x, \neg y)$. So we can reduce to the problem of evaluating a complete binary tree of NAND gates. (In the conversion, it may be necessary to flip the input gates or the output gate or both; but this does not affect the query complexity of the problem.)

NAND	0	1
0	1	1
1	1	0

The key to savings in the randomized case is that when either input to a NAND gate is found to be 0, the output is bound to be 1 and therefore we don't need to look at the other input. This suggests a *Random DFS* evaluation–randomly, uniformly pick which child to recurse on–should be efficient. Of course, the potential savings make sense for deterministic evaluation too, but if we go in a prescribed order, the adversary will always make us see a 1 first. (This is exactly what we exploited in the lower bound in the previous section.) A randomized algorithm has the advantage that if either child equals 0, we have half probability of looking at that child first and avoiding the other evaluation.

Simpler question: MAJ3 trees of depth n

There's a small technical challenge in analyzing the NAND tree so let's look at an easier question: The complete ternary tree of depth n in which every non-leaf vertex is a Majority gate. (A singleton node is depth 0.) Let S_n = worst-case expected number of leaves evaluated by Random DFS.



Whatever the values of the three children of a node, we only have to evaluate the last child if the first two disagreed. At least one of the three pairs is always in agreement, so the probability we first find a disagreeing pair is at most 2/3. So

$$S_n \leq (2/3) \cdot 3 \cdot S_{n-1} + (1/3) \cdot 2 \cdot S_{n-1} = (8/3) S_{n-1}$$

Hence (keeping in mind $S_0 = 1$), $S_n \leq (8/3)^n = N^{\log_3 8/3} \cong N^{0.893}$ where $N = 3^n$ is the number of leaves.

What we have here is very much like a branching process. Every node has either two or three children (until level n at which the branching process is cut off), and the savings in the randomized algorithm comes from showing that, in the notation of the next section, $\bar{\mu} \leq 8/3$.

Digression: branching processes

Let μ be a probability distribution on the nonnegative integers. The *branching process* or *Galton-Watson process* with distribution μ is the following tree-valued random variable T:

T has a root. Each vertex v of T gets some N_v children, for N_v independently distributed according to μ. Let $\bar{\mu} = E(N_v)$ (possibly infinite).

Theorem 6. *TFAE:*

1. *T is a.s. finite. (a.s. = Almost Surely = With Probability 1)*
2. *$\bar{\mu} \leq 1$ and $\mu_1 < 1$.*

The intuition is this. In the subcritical regime, i.e., $\bar{\mu} < 1$, each parent has less than 1 child on average—so no wonder the generations die out with probability 1. In the supercritical regime, $\bar{\mu} > 1$, things are not so definite—it could happen for example that the root has no children at all—but, the number of vertices at a level is generally drifting upwards, and as it grows, the likelihood of population collapse decreases drastically, so overall, the probability of the tree being infinite is positive. The critical case $\bar{\mu} = 1$ is (as always in these kinds of problems) hardest to determine and here we have two cases. One is that $\mu_1 = 1$ in which case the process is deterministic, T is infinite and there is nothing more to say. The other is that $\mu_1 < 1$ which implies that $\mu_0 > 0$. Now the number of children at each level of the tree is just drifting without bias up or down. However, there is an absorbing boundary at 0: extinction is forever. This process is not a random walk with bounded step size, such as we have studied in "gambler's ruin", but intuitively it behaves similarly, and it goes extinct with probability 1 for basically the same reason.

Here we prove formally only a weaker version of (2) $\Rightarrow$ (1); after that we'll give some idea of the other direction but point to Grimmett and Stirzaker §5.4 Thm (5) for a full proof [18].

If $\bar{\mu} < 1$ then T is a.s. finite.

Proof. We will use the shorthand $\mu_{\geq i} = \sum_{j \geq i} \mu_j$.

When a vertex has N children, we list them in an arbitrary "birth order" as children $1, \ldots, N$. The "address" of a vertex of the tree is a finite string of positive integers $(X_1 \ldots X_\ell)$: the root is represented by the empty string and the address of a vertex is its parent's address followed by its place in the birth order.

For a string $(X_1 \ldots X_\ell)$, let $[\![X_1 \ldots X_\ell]\!]$ be the (indicator rv of) the event that this address exists in the tree. An equivalent characterization of this event is that

1. The root has at least X_1 children, *and*
2. The vertex (X_1) has at least X_2 children, *and* ...
3. The vertex $(X_1 \ldots X_{\ell-1})$ has at least X_ℓ children.

Note that $\bar{\mu} = \sum_{i \geq 0} i\mu_i = \sum_{i \geq 1} \mu_{\geq i}$.

We have the following:

$$\Pr([\![X_1 \ldots X_\ell]\!]) = \prod_{j=1}^{\ell} \mu_{\geq X_j}$$

The event that T is infinite is equivalent to the event that $\sum_{\vec{X}} [\![\vec{X}]\!] = \infty$. Let's calculate the expectation of the LHS. Note that in the following calculation all products are of finitely many terms.

$$
\begin{aligned}
E(\sum_{\vec{X}} [\![\vec{X}]\!]) = \sum_{\vec{X}} \Pr([\![\vec{X}]\!]) &= \sum_{\ell \geq 0} \sum_{\vec{X}:|\vec{X}|=\ell} \prod_{j=1}^{\ell} \mu_{\geq X_j} \\
&= \sum_{\ell \geq 0} \prod_{j=1}^{\ell} \sum_{X \geq 1} \mu_{\geq X} \\
&= \sum_{\ell \geq 0} \prod_{j=1}^{\ell} \bar{\mu} \\
&= \sum_{\ell \geq 0} \bar{\mu}^{\ell} \\
&= \frac{1}{1-\bar{\mu}} < \infty \quad \text{here we finally use } \bar{\mu} < 1
\end{aligned}
$$

Now let's recall the first Borel-Cantelli lemma (**??**) *(Reminder: Let B_i be countably many events s.t. $\sum_{i \geq 1} \Pr(B_i) < \infty$. Then $\Pr(\limsup B) = 0$.)* It follows that T is almost surely finite. □