Proofs for Lecture 8

Ariel Procaccia

Assumption 1. Game trees are binary and the values of all leaves are in $\{0, 1\}$.

Theorem 2. *Given a game tree with n leaves and an algorithm that evaluates the game, there exists an input such that α-β pruning must query all the leaves.*

Proof sketch. We prove the claim by induction on the number of leaves n, using an adversary argument. The base case ($n = 1$) is trivial.

Given a tree with n leaves, assume that the claim holds for all trees with $n - 1$ leaves. We can eliminate all nodes v with only one child by replacing v with its child; note that this does not change the number of leaves. Hence we can assume without loss of generality that each node has exactly two children (i.e., the binary tree is full).

Now, the algorithm queries some leaf first. If the leaf's parent is a max node, the adversary answers that the value is 0; and if the leaf's parent is a min node, the adversary responds with 1. Based on these answers, it is not possible to decide the value of the leaf's parent without querying the leaf's sibling. In fact, the parent's value depends only on the leaf's sibling, so we can replace the parent with the sibling, thus obtaining a tree with $n - 1$ unqueried leaves. By the induction assumption, all the leaves of this new tree must be queried. □

Consider a randomized version of α-β pruning that switches the order of children with probability 1/2.

Lemma 3. *Consider a game tree whose root is a max node, with two children that are min nodes, each of which has two children that are leaves. The expected number of leaves randomized α-β pruning needs to query given this tree is at most* 3*.*

Proof. We consider two cases.

Case 1: the value of the tree is 1. It follows that the value of at least one of the min nodes is 1; without loss of generality the value of the left min node is 1. With probability at least 1/2 the algorithm first computes the value of the left min node. If this happens, the value of the second min node would not be calculated. Therefore, the expected running time is at most

$$\frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 4 = 3.$$

Case 2: the value of the tree is 0. In this setting, the value of both min nodes is 0. For each one, with probability 1/2 the algorithm would first query the value of a leaf with utility 0, and its sibling would not be queried. The expected running time is at most

$$2\left(\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 2\right) = 3.$$

□

Next, consider a special game tree that we refer to as an *alternating* game tree; it is a binary tree of height $2h$, where each internal node has exactly two children, the nodes at even depth are max nodes, and the nodes at odd depth are min nodes.

Theorem 4. *The expected number of leaves randomized α-β pruning needs to query given an alternating game tree with n leaves is at most $n^{\log_4 3}$.*

Proof. Using Lemma 3 we see that the algorithm needs to evaluate at most three of its subtrees at height $2(h-1)$ in expectation. That is, we have that $T(h) \leq 3T(h-1)$. In addition, note that $T(0) = 1$ (because a tree of height 0 is a leaf). Therefore, $T(h) \leq 3^h$. Since we are dealing with a full binary tree of height $2h$, we have that $n = 2^{2h} = 4^h$. It follows that

$$T(h) \leq 3^h = 4^{(\log_4 3)h} = (4^h)^{\log_4 3} = n^{\log_4 3}.$$

□

Lemma 5 (Yao's minmax principle [1])**.** *A lower bound on the performance of the best randomized algorithm on the worst deterministic input is given by the performance of the best deterministic algorithm on some probabilistic input.*

Theorem 6. *For any randomized algorithm there exists a game tree with n leaves such that the algorithm needs to query at least $n/2$ leaves in expectation.*

Proof sketch. We consider trees of height $2h$ where the nodes at depth $0, \ldots, h$ are max nodes, and the nodes at depth $h+1, \ldots, 2h$ are min nodes. That is, player 1 makes h consecutive moves and then player 2 makes h consecutive moves. In other words, for each of the 2^h possible strategies of player 1, player 2 has 2^h possible responses. We can represent the game as a matrix M with 2^h rows and 2^h columns; when player 1 plays strategy i, and player 2 responds with strategy j, the utility (to player 1) is M_{ij}. The value of the tree is 1 if and only if there is a row that only contains ones.

Consider the uniform distribution over matrices that have exactly one zero in each row. In order to give a correct answer, a deterministic algorithm has to find all the zeros in the matrix. To find a zero in a given row, the algorithm has to query half the cells in expectation. The information about one row does not help pinpoint the zero in a different row, and therefore the algorithm must query half the cells in the matrix. The theorem now follows from Lemma 5. □

References

[1] A. C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proceedings of the 17th Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.