# The lower bound for evaluating a recursive ternary majority function: an entropy-free proof

Itamar Landau, Asaf Nachmias, Yuval Peres, and Sithparran Vanniasegaram *

May 15, 2006

## 1 Introduction

In this expository note, we discuss the recursive ternary majority function. We begin with a ternary tree T of depth $h$ in which each node is labeled either $+1$ or $-1$ and the label of each non-leaf node agrees with the label of the majority of the node's children. The recursive ternary majority function computes the root label from the labels of the $3^h$ leaves of T. We are interested in the number of leaves queried by a given algorithm. There is a trivial lower bound for any algorithm and any setting of the tree of $2^h$ leaves, and an obvious upper bound of $3^h$. There is a stronger upper bound of $(8/3)^h$ (meaning there exists an algorithm that will do no worse than $(8/3)^h$ on average), which we discuss below, and also a slight improvement on that bound which we do not discuss. Here we give a simplified version of a proof that for any randomized algorithm, the worst-case performance is at least $(7/3)^h$, i.e. for any algorithm there is a particular input of the leaves which forces the algorithm to query at least $(7/3)^h$ leaves, averaging with respect to the internalized randomness of the algorithm. The general approach is motivated by Yao's Minimax Principle [2], which guarantees that the worst-case runtime for any (possibly randomized) algorithm will be no better than the runtime for the best algorithm on any particular distribution of leaves.

The authors in [1] use a clever and subtle coupling argument to prove the following theorem:

*U.C. Berkeley. Research of the first and last authors supported in part by U.C. Berkeley statistics department's NSF-VIGRE grant. Research of second and third authors supported in part by NSF grants #DMS-0244479 and #DMS-0104073

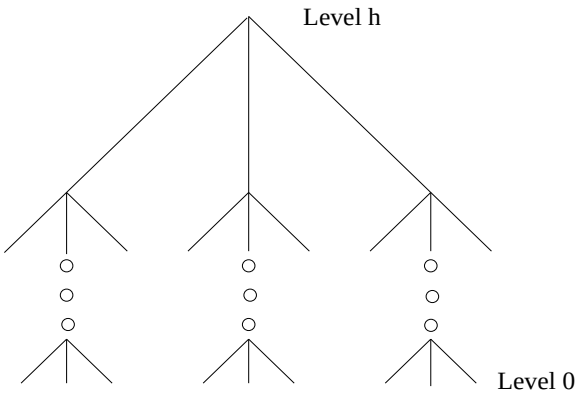


Figure 1: A tree of depth h

**Theorem 1.1.** *Let $T$ be a ternary tree of depth $h$. There exists a distribution on the labels of $T$ such that any (possibly randomized) algorithm which computes the recursive majority function must query at least $(7/3)^h$ leaves on average.*

Given Theorem 4, the following result is trivial:

**Corollary 1.2.** *For any (possibly randomized) algorithm, there exists a particular setting of the labels of $T$ such that the algorithm will read at least $(7/3)^h$ leaves on average, i.e. there is a lower bound for the worst-case performance of any algorithm of $(7/3)^h$.*

The authors in [1] make a complex argument which makes reference to information theory. In our note, we simplify their approach for proving Theorem 4.

## 2 Finding the bounds

Before going into the proof of Theorem 4, we briefly discuss the upper bound of $(8/3)^h$ mentioned above. The algorithm that achieves this result is defined recursively as follows: To evaluate a node $v$, pick two of its children $v_1$ and $v_2$ at random and recursively evaluate them. If the labels of $v_1$ and $v_2$ are the same then this determines the label of $v$ and we are done. With some probability $p$ the nodes have different labels and we query the final child

$v_3$. Thus the expected number of these nodes queried by the algorithm is $2(1-p)+3p = 2+p$. Now consider an arbitrary distribution of the leaves which attempts to maximize this expectation, i.e. to maximize $p$, the probability that two randomly chosen siblings have different labels. It is clear that the worst case distribution will always have one node disagreeing with its siblings, thus yielding $p = 2/3$ and an expectation for the number of leaves read of $8/3$. By induction, we have our upper bound of $(8/3)^h$. The development of this upper bound motivates the construction of a distribution on the leaves of a tree T in which every set of three sibling nodes has one dissenting, or minority, node.

We now define a distribution on the labels of T. The root receives $+1$ or $-1$ with probability $1/2$ each, and we proceed in the following recursive manner: when a non-leaf node $v$ receives label $x \in \{1, -1\}$, we draw one of its children at random and give it label $-x$, while giving its two unset siblings the label $x$.

The distribution on T defines naturally the *minority path* which is the path from the root to one of the leaves where each node has a different label from its father. We will use the following notation:

- $M \equiv$ the set of nodes in the minority path.
- $C(v) \equiv$ the children of a node $v$.
- $Z(v) \equiv$ the set of leaves of the subtree beneath a node $v$.
- $R_A \equiv$ the set of leaves queried by an algorithm $A$.
- $\mathrm{T(m)} \equiv$ the set of nodes at level $m$ (counting from bottom up) of a tree T.
- $J(h,\ell) \equiv \inf_A \sum_{v \in \mathrm{T}(\ell)} \mathbb{E}\left[ |Z(v) \cap R_A| \,\middle|\, v \in M \right]$, where the infimum is taken over all randomized algorithms $A$.

To be clear, for a tree of depth $h$, the set $\mathrm{T}(h)$ consists of the root and $\mathrm{T}(0)$ consists of the leaves of the tree. In particular, the amount we wish to lower bound is $J(h,h) = \inf_A \mathbb{E}\left[ |R_A| \right]$.

**Lemma 2.1.** *Let $0 \le \ell \le h$. For a node $v$ at level $\ell$ of a tree $T$ of depth $h$, let $v_1$, $v_2$, $v_3$ be the children of $v$. Then given any algorithm $A_0$,*

$$\sum_{v \in \mathrm{T}(\ell)} \sum_{j=1}^{3} \mathbb{E}\left[ |Z(v_j) \cap R_{A_0}| \,\middle|\, v_j \notin M \right] \ge 3 \cdot J(h-1, \ell-1)\,.$$

*Proof.* Fix an algorithm $A_0$ for a tree with depth $h$. Given a tree $\mathrm{T}_1$ of depth $h-1$ we will create an algorithm by coupling our tree with a tree $\mathrm{T}_2$ of depth $h$ and using our algorithm $A_0$. We construct the labels of $\mathrm{T}_2$ in the following way: For the top $h-\ell$ levels, $\mathrm{T}_2$ will have exactly the same labels as $\mathrm{T}_1$. At level $\ell$ of $\mathrm{T}_2$ we couple each node with the corresponding node at level $\ell-1$ of $\mathrm{T}_1$. Let $u$ denote a node at level $\ell-1$ of $\mathrm{T}_1$, and let $v$ denote the corresponding node of $\mathrm{T}_2$, with children $v_1$, $v_2$, and $v_3$. Now let $w$ denote a uniform random choice from among the children of $v$. Assign $w$ the same label as $u$ and assign all of its descendants the labels of the descendants of $u$, i.e. copy the subtree below $u$ to the subtree below $w$. This process is illustrated in Figure 2. Of the remaining children of $v$, pick one randomly and label it $+1$. Label the remaining child $-1$. For each of these children, label its subtree following the recursive definition of the leaves' distribution, i.e. pick a child at random to be the minority and give the other two children the label of the parent. Observe that $w$ is a majority node by construction. We follow this procedure for each node $v$ in level $\ell$ of $\mathrm{T}_2$. Note that the leaves of $\mathrm{T}_2$ have the required distribution and it is coupled with $\mathrm{T}_1$ in three important senses: (1) the root label is the same for both trees and (2) the leaves below each node $u$ in level $\ell-1$ of $\mathrm{T}_1$ are the same as the leaves below a randomly chosen child $w$ of the corresponding node $v$ of $\mathrm{T}_2$.

Now we use the algorithm $A_0$ on the coupled tree $\mathrm{T}_2$ to find the root label of $\mathrm{T}_1$. This completes the description of a randomized algorithm, which we call $A_1$, for a tree of size $h-1$. We now analyze this algorithm's performance. For each node $u \in \mathrm{T}_1(\ell-1)$ with corresponding node $v \in \mathrm{T}_2(\ell)$, and $w$ the uniform random choice from the children of $v$, we have $|Z(w) \cap R_{A_0}| = |Z(u) \cap R_{A_1}|$ by construction of the algorithm $A_1$. Thus

$$\mathbb{E}\left[|Z(w) \cap R_{A_0}| \,\middle|\, w \notin M\right] = \mathbb{E}\left[|Z(u) \cap R_{A_1}| \,\middle|\, u \in M\right].$$

Now since $w$ was chosen at random from $v_1$, $v_2$, $v_3$ we have that

$$\sum_{j=1}^{3} \frac{1}{3} \cdot \mathbb{E}\left[|Z(v_j) \cap R_{A_0}| \,\middle|\, v_j \notin M\right] = \mathbb{E}\left[|Z(u) \cap R_{A_1}| \,\middle|\, u \in M\right].$$

Multiplying both sides by three and then taking the sum over all nodes $v$ in level $\ell$ of $\mathrm{T}_2$, which are coupled to the nodes $u$ in level $\ell-1$ of $\mathrm{T}_1$ we get

$$\sum_{v \in T_2(\ell)} \sum_{j=1}^{3} \mathbb{E}\left[|Z(v_j) \cap R_{A_0}| \,\middle|\, v_j \notin M\right] = 3 \cdot \sum_{u \in T(\ell-1)} \mathbb{E}\left[|Z(u) \cap R_{A_1}| \,\middle|\, u \in M\right].$$

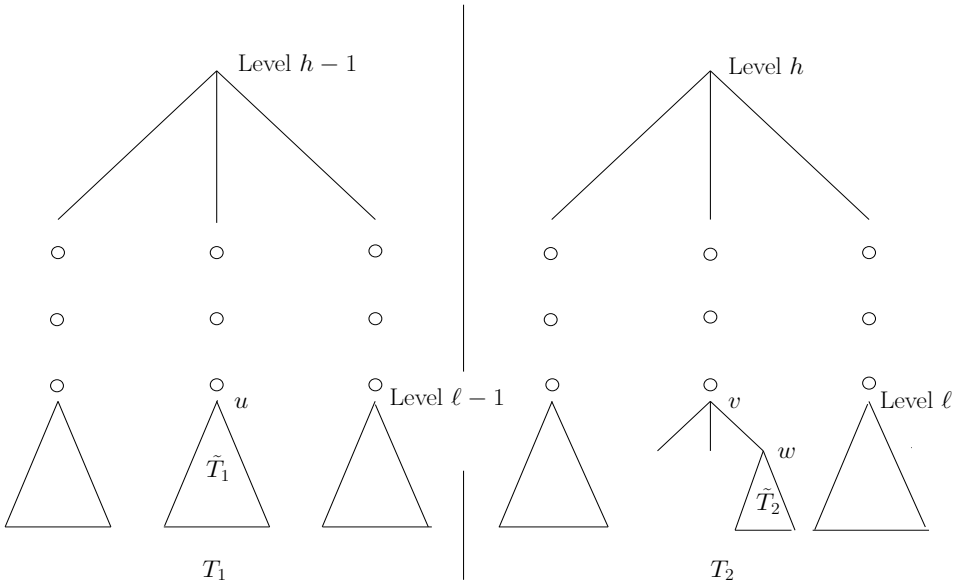


Figure 2: In addition to the top $h-\ell$ levels being similar in both trees, the labels of the descendants of $w$ ($\tilde{T}_2$) match the labels of the descendants of $u$ ($\tilde{T}_1$.

Notice that the sum on the right hand side is by definition bigger than $3 \cdot J(h-1, \ell-1)$, so our lemma is established. □

**Lemma 2.2.** *For all $h$ and $\ell \leq h$, $J(h,\ell) \geq \frac{1}{3} \cdot J(h,\ell-1) + 2 \cdot J(h-1,\ell-1)$.*

*Proof.* Fix an algorithm $A_0$ for a tree of depth $h$, and select a vertex $v \in T(\ell)$. Denote $v$'s children by $v_1$, $v_2$, and $v_3$. Note that $Z(v) = \coprod_{j=1}^{3} Z(v_j)$. As a result,

$$\mathbb{E}\Big[|Z(v) \cap R_{A_0}| \;\Big|\; v \in M\Big] = \sum_{j=1}^{3} \mathbb{E}\Big[|Z(v_j) \cap R_{A_0}| \;\Big|\; v \in M\Big]. \tag{1}$$

For each $j = 1, 2, 3$ we have that $P\Big(v_j \in M \mid v \in M\Big) = 1/3$. Furthermore, by symmetry the number of leaves read below a child in the majority does not depend on which of its siblings is in fact the minority, i.e. for $i, j \in \{1,2,3\}, i \neq j$,

$$\mathbb{E}\Big[|Z(v_i) \cap R_{A_0}| \;\Big|\; v_i \notin M\Big] = \mathbb{E}\Big[|Z(v_i) \cap R_{A_0}| \;\Big|\; v_j \in M\Big]$$

Thus we have

$$\begin{aligned}\mathbb{E}\left[|Z(v_j)\cap R_{A_0}|\,\middle|\, v\in M\right] &= \frac{1}{3}\cdot\mathbb{E}\left[|Z(v_j)\cap R_{A_0}|\,\middle|\, v_j\in M\right]\\ &+ \frac{2}{3}\cdot\mathbb{E}\left[|Z(v_j)\cap R_{A_0}|\,\middle|\, v_j\notin M\right].\end{aligned}$$

Therefore, substituting into (1) above we get,

$$\begin{aligned}\mathbb{E}\left[|Z(v)\cap R_{A_0}|\,\middle|\, v\in M\right] &= \frac{1}{3}\cdot\sum_{j=1}^{3}\mathbb{E}\left[|Z(v_j)\cap R_{A_0}|\,\middle|\, v_j\in M\right]\\ &+ \frac{2}{3}\cdot\sum_{j=1}^{3}\mathbb{E}\left[|Z(v_j)\cap R_{A_0}|\,\middle|\, v_j\notin M\right].\end{aligned}$$

Now, summing over all nodes $v$ in level $\ell$ and applying Lemma 2.2 to the second sum on the right, we have

$$\begin{aligned}\sum_{v\in\mathrm{T}(\ell)}\mathbb{E}\left[|Z(v)\cap R_{A_0}|\,\middle|\, v\in M\right] &\geq \frac{1}{3}\cdot\sum_{v\in\mathrm{T}(\ell)}\sum_{j=1}^{3}\mathbb{E}\left[|Z(v_j)\cap R_{A_0}|\,\middle|\, v_j\in M\right]\\ &+ 2\cdot J(h-1,\ell-1).\end{aligned}$$

Taking the infimum, the left hand side is by definition $J(h,\ell)$ and the first sum on the right is by definition $J(h,\ell-1)$ and we have our result. □

**Lemma 2.3.** $J(h,\ell)\geq\left(\frac{7}{3}\right)^{\ell}$

*Proof.* For fixed $h\geq 0$, we will prove this by induction on $\ell$.
For $\ell=0$, $J(h,0)\geq 1$ since any algorithm has to read at least one of the $3^h$ leaves. Now assume the result holds for $J(h,\ell-1)$ and apply Lemma 2.2:

$$\begin{aligned}J(h,\ell) &\geq \frac{1}{3}\cdot\left(\frac{7}{3}\right)^{\ell-1}+2\cdot\left(\frac{7}{3}\right)^{\ell-1}\\ &= \frac{7^{\ell-1}+6\cdot 7^{\ell-1}}{3^{\ell}}\\ &= \left(\frac{7}{3}\right)^{\ell}.\ \square\end{aligned}$$

**Theorem 2.4.** $J(h,h)\geq(7/3)^h$.

*Proof.* Simply plug in $\ell=h$ in Lemma 2.3. □

## 3 Finding Upper Bounds for $b$-ary Trees

In this section we give an algorithm that for any setting of the leaves of a $b$-ary tree will read on average $(b - \frac{b-1}{b+3})^h$ leaves. The following algorithm that we define recursively is a generalization of the algorithm given for the ternary tree: first pick $\frac{b+1}{2}$ children of a node $v$ uniformly at random. If these $\frac{b+1}{2}$ have the same label, then $v$ also has that same label. Otherwise, continue picking children at random until the label of $v$ can be determined (i.e. until there are $\frac{b+1}{2}$ children which have the same label). To find the expected number of leaves read (for simplicity $L$ will denote the number of leaves read), we will use the tail sum formula for expectation:

$$E[L] = \sum_{i=1}^{b} \mathrm{P}(L \geq i)$$

Since at least $\frac{b+1}{2}$ leaves must be read, the last equation simplifies to

$$E[L] = \frac{b+1}{2} + \sum_{i=\frac{b+3}{2}}^{b} \mathrm{P}(L \geq i) \tag{2}$$

The probability that at least $i$ leaves will be read is the same as the probability that reading $i-1$ is not sufficient. Therefore,

$$\begin{aligned}
P(L \geq i) &= P(i-1 \text{ leaves is not sufficient}) \\
&= 1 - P(i-1 \text{ leaves is sufficient}) \\
&= 1 - P(\text{all } n-i+1 \text{ unexamined leaves have different label from parent}) \\
&= 1 - \frac{(b-1)/2}{b} \cdot \frac{(b-1)/2-1}{b-1} \cdots \frac{i-(b-1)/2-1}{i} \\
&= 1 - \frac{((b-1)/2)_{b-i+1}}{(b)_{b-i+1}}
\end{aligned}$$

The third and fourth equalities follow from the fact that the parent has exactly $\frac{b+1}{2}$ children that have the same label as it and that $\frac{b-1}{2}$ children have a different label. Plugging the last equality into (2) yields

$$\begin{aligned}
E[L] &= \frac{b+1}{2} + \sum_{i=\frac{b+3}{2}}^{b} 1 - \frac{((b-1)/2)_{b-i+1}}{(b)_{b-i+1}} \\
&= b - \sum_{i=\frac{b+3}{2}}^{b} \frac{((b-1)/2)_{b-i+1}}{(b)_{b-i+1}} \\
&= b - \sum_{i=1}^{\frac{b-1}{2}} \frac{((b-1)/2)_i}{(b)_i}
\end{aligned}$$

All that is left to show is that

$$\sum_{i=1}^{\frac{b-1}{2}} \frac{((b-1)/2)_i}{(b)_i} = \frac{b-1}{b+3}. \tag{3}$$

This can be proven using a combinatorial argument that for any $k$ and $n \geq k$,

$$\sum_{i=1}^{k} \frac{(k)_i}{(n)_i} = \frac{k}{n-k+1}. \tag{4}$$

First, note that multiplying each side of the previous equation by $\frac{n!\cdot(n-k+1)}{k}$ gives the following equation:

$$\sum_{i=1}^{k} (k-1)_{i-1}(n-k+1)(n-1)! = n!. \tag{5}$$

Therefore, to prove (4), it suffices to prove (5). Fix $n \geq k$. Given a permutation $\pi$ on $n$ elements, let $I_k(\pi)$ be the smallest $i$ such that $\pi(i) \geq k$, and let $A_i$ be the set of permutations with $I_k(\pi) = i$. Clearly, $|A_i| = (k-1)_{i-1}(n-k+1)(n-i)!$ since you first have to choose $\pi(1), ..., \pi(i-1)$ from $1, ..., k-1$ then choose $\pi(i)$ from $k, ..., n$ and finally choose the remaining $n-i$ images $\pi(i+1), ..., \pi(n)$ from the remaining $n-i$ possibilities (the complement of $\pi(1), ..., \pi(i)$). Using the identity that $\sum_{i=1}^{k} |A_i| = n!$ gives (5) and so (3) follows.

# 4 Finding lower Bounds for $b$-ary trees

We now extend Theorem to the following theorem for a $b$-ary tree of depth $h$:

**Theorem 4.1.** *Let $T$ be a $b$-ary tree of depth $h$. There exists a distribution on the labels of $T$ such that any (possibly randomized) algorithm which computes the recursive majority function must query at least $(\frac{b+1}{2} + \frac{b-1}{2b})^h$ leaves on average.*

This extension follows from the following extensions of Lemmas 2.1, 2.2, and 2.3:

**Lemma 4.2.** *Let $0 \leq \ell \leq h$. For a node $v$ at level $\ell$ of a tree $T$ of depth $h$, let $v_1$, $v_2$, ..., $v_b$ be the children of $v$. Then given any algorithm $A_0$,*

$$\sum_{v \in \mathrm{T}(\ell)} \sum_{j=1}^{b} \mathbb{E}\left[ |Z(v_j) \cap R_{A_0}| \,\middle|\, v_j \notin M \right] \geq b \cdot J(h-1, \ell-1)\,.$$

**Lemma 4.3.** *For all $h$ and $\ell \leq h$, $J(h,\ell) \geq \frac{b-1}{2b} \cdot J(h,\ell-1) + \frac{b+1}{2} \cdot J(h-1,\ell-1)$.*

**Lemma 4.4.** $J(h,l) \geq (\frac{b+1}{2} + \frac{b-1}{2b})^l$

To prove these statements, we define a distribution on a $b$-ary tree $T$ (which will be similar to the distribution defined for the ternary tree). The root of $T$ receives $+1$ or $-1$ with probability $1/2$ each, and we proceed in the following recursive manner: when a non-leaf node $v$ receives label $x \in \{1,-1\}$, we draw $\frac{b-1}{2}$ of its children at random and give it label $-x$, while giving its $\frac{b+1}{2}$ unset siblings the label $x$. Instead of a minority path, the distribution defines naturally a *minority tree*. The minority tree consists of the root, the $\frac{b-1}{2}$ children which have a different label, the $(\frac{b-1}{2})^2$ grandchildren who have the same label and a different label from the parents, etc.

The proof of Lemma 4.2 mimics that of Lemma 2.1 except for the fact that $v$ has $n$ children and so after a child is chosen randomly to have the same label as $u$ and have descendants whose labels match the labels of $u$'s descendants, $\frac{b-1}{2}$ children will be picked randomly to have the label $+1$. Then, the remaining $\frac{b-1}{2}$ children will be given the label $-1$. For the $n$-ary tree, all the equations proved in Lemma 4 (with $n$ replacing 3 and $M$ now denoting the minority tree) will hold, and therefore Lemma 4.2 follows. With Lemma 4.2 in hand, Lemma 4.3 can be established by using essentially the same proof that was used to prove Lemma 2.2. (If a vertex is in the minority tree, the probability a randomly chosen child of his will be in the minority tree is $\frac{b-1}{2b}$.) Lemma 4.4 follows from Lemma 5 using the same inductive argument that was used in proving Lemma 4.4. Finally, substituting $h$ for $l$ yields Theorem 4.1.

# 5 Further Discussion

As mentioned in the introduction, the approach of this proof is motivated by Yao's minimax principle, which stems from the work of John Von Neumann in his development of game theory. To illuminate the connection, we imagine a game in which one player, who we call Luis the Labeler, arranges the labels of the leaves of a ternary tree and a second player, who we call Carol the Computer, queries the leaves until she can correctly evaluate the root label. The payoff to Luis is the number of leaves Carol reads, and Carol loses this same amount. A pure strategy for Luis is a particular setting of the leaves, and a mixed strategy is a distribution on the leaves. A pure strategy for Carol is a deterministic algorithm, and a mixed strategy is an algorithm

with internalized randomness. The game is zero-sum, and Von Neumann's minimax theorem guarantees that there exist optimal mixed strategies for both players and a well-defined value $V$. As we have seen, if Luis employs the mixed strategy defined by the distribution described above, then Carol can do no better than to lose $(7/3)^h$. Conversely, no matter what mixed strategy Carol employs, Luis can find a pure strategy such that Carol should expect to lose at least $(7/3)^h$. This general game theoretic approach was first applied insightfully to the study of complexity theory by Yao.

## References


[1] J. Jayram, R. Kumar, and D. Sivakumar. Two applications of information complexity. *STOC* 2003: 673-682.

[2] A.C. Yao. Probabilistic computations: Toward a unified measure of complexity. *FOCS* 1977: 222-227.