THE GAME OF "N QUESTIONS" ON A TREE*

Ronald L. RIVEST

M.I.T., Department of Electrical Engineering and Computer Science, Cambridge, MA 02139, U.S.A.

Received 4 February 1976
Revised 7 April 1976

We consider the minimax number of questions required to determine which leaf in a finite binary tree T your opponent has chosen, where each question may ask if the leaf is in a specified subtree of T. The requisite number of questions is shown to be approximately the logarithm (base \( \phi \)) of the number of leaves in T as T becomes large, where \( \phi = 1.61803... \) is the "golden ratio". Specifically, \( q \) questions are sufficient to reduce the number of possibilities by a factor of \( 2/F_{q+3} \) (where \( F_i \) is the \( i \)th Fibonacci number), and this is the best possible.

1. Introduction

We consider the problem of identifying a leaf in a finite binary tree T by posing a sequence of questions of the form, "is the leaf in subtree S of T?", for various S. Our main result is that (in a sufficiently large tree) \( q \) questions are sufficient to reduce the number of possibilities by a factor of \( 2/F_{q+3} \), where \( F_i \) is the \( i \)th Fibonacci number. This generalizes a well-known result that every finite binary tree contains a subtree having between 1/3 and 2/3 of all the tree's leaves [4, 6]. Our result is obtained by analyzing a "greedy" algorithm which always chooses the subtree S which has a number of leaves as nearly equal to one-half of the number of remaining leaves as possible. We show that this is the best possible worst-case result by demonstrating that the "Fibonacci trees" yield a corresponding lower bound on the achievable performance.

2. Definitions

We shall express our problem by using finite sets of finite-length words over the alphabet \( \Sigma = \{0, 1\} \) to represent binary trees, and using regular expressions to denote sets of words [2]. We say that \( T \subseteq \Sigma^* \) is a binary tree iff T is prefix-free: no word in T is a prefix of any other word in T. In this paper all binary trees will be finite. Each word in T corresponds to a path from the root to a leaf in a

* This research was supported by the National Science Foundation under research contract No. DCR74-12997.
"conventional" binary tree \([3, \text{Section 2.3}]\) in a natural manner (zeroes indicating left branches and ones indicating right branches).

If \(S \subseteq \Sigma^*\), we define

\[
\pi S = \{ x \in \Sigma^* \mid (\exists y \in \Sigma^*)xy \in S \}
\]

to be the set of: prefixes of \(S\). Note that \(S \subseteq \pi S\). For brevity we let \(\pi x\) denote \(\pi\{x\}\) for \(x \in \Sigma^*\).

To illustrate, the first six Fibonacci trees are shown in Fig. 1; they are defined by

\[
\begin{align*}
\mathcal{F}_1 &= \mathcal{F}_2 = \{ A \} \quad (A \text{ denotes the empty word}) \\
\mathcal{F}_i &= 0\mathcal{F}_{i-2} \cup 1\mathcal{F}_{i-1} \quad \text{for } i \geq 3.
\end{align*}
\]

The elements of \(\mathcal{F}_i\) are boxed; other elements of \(\pi\mathcal{F}_i\) are circled.

---

Given a binary tree \(T\) and \(x \in \pi T\), the subtree of \(x\) in \(T\) (denoted \(T_x\)) is defined

\[
T_x = x\Sigma^* \cap T.
\]

The complement \(T - T_x\) of the subtree of \(x\) in \(T\) is denoted \(T'_x\).

Consider the following two-person game played on a binary tree \(T\). Player \(A\) chooses a word \(x_0 \in T\) which player \(B\) wishes to determine by posing as few questions as possible to \(A\). All of \(B\)'s questions must be of the form, "is \(y\) a prefix of \(x_0\)?" for some \(y \in \pi T\), and player \(B\) obtains \(A\)'s response to the \(i^{th}\) question before posing his \((i+1)^{th}\) question.

The model proposed here (binary trees) corresponds reasonably well to a large number of practical applications where a hierarchical organization of concepts
forms the framework for an identification process, and specific tests exist for determining whether the unknown quantity is a number of a given category in the hierarchy. For example, the problems of identifying an unknown disease in a patient, an unknown chemical compound, or a faulty gate in a logic circuit might be viewed in this manner. The model used here is a restricted form of the general “group-testing” problem [5, 7]; the difference is that in our situation only certain subsets (corresponding to subtrees) may be tested.

It is well known [4, 6] that with one question \( B \) can reduce the number of possible candidates for \( x_0 \) to no more than \( 2 \mid T \mid /3 \) if \( \mid T \mid \geq 2 \), and that this is the best possible result (consider \( T = \{0, 10, 11\} \)). In general \( B \) can achieve this by picking \( y \) so that \( \max(\mid T_y \mid, \mid T'_y \mid) \) is as near to \( \mid T \mid /2 \) as possible.

We will denote the worst-case size of the subset that \( B \) can constrain \( x_0 \) to lie in after asking \( i \) questions by \( P_i(T) \):

\[
P_0(T) = \mid T \mid,
\]

and

\[
P_{i+1}(T) = \min_{y \in \pi T} (\max(P_i(T_y), P_i(T'_y))) \quad \text{for } i \geq 0.
\]

For example, \( P_i(\mathcal{F}_{i+3}) = 2 \) for \( i \geq 0 \), as we shall prove later.

In order to talk meaningfully about the usefulness of a number of questions, it is necessary that the tree \( T \) be large enough so that target leaf is not identified before all the questions are asked. With this understanding, we define

\[
r_i = \text{lub}\{P_i(T) / \mid T \mid : P_i(T) \geq 2\}
\]

to be the least upper bound on the fraction of \( T \) that \( B \) can constrain \( x_0 \) to lie in after \( i \) questions. Given that \( \mid T \mid \) is large enough (say, \( > 2^i \)), player \( B \) can reduce the number of possibilities for \( x_0 \) to at most \( r_i \mid T \mid \) with \( i \) questions.

In the next section we show that \( r_i \geq 2/F_{i+3} \) for \( i \geq 1 \), using Fibonacci trees. In Section 4 we show that \( r_i \leq 2/F_{i+3} \) for \( i \geq 1 \) using the “greedy algorithm”.

3. The lower bound

**Theorem 3.1.** \( r_i \geq 2/F_{i+3} \) for \( i \geq 0 \).

**Proof.** We prove this by demonstrating that \( P_i(\mathcal{F}_{i+3}) \geq 2 \) for all \( i \), using induction on \( i \).

By inspection, \( P_i(\mathcal{F}_i) = 2 \), so we have that \( r_i \geq 2/3 \).

For the inductive step, we first remark that if \( T = 0R \cup 1S \) is a binary tree, then \( P_i(T) \leq P_i(U) \) for all \( i \) if \( I \supseteq aR \cup bS \) where \( a, b \in \Sigma^* \) such that \( \{a, b\} \) is prefix-free. A question about \( aR \cup bS \) can be transformed into an equivalent question about \( T \) by replacing an initial \( a \) or \( b \) with 0 or 1, respectively.

We next note that for any \( x \in \pi \mathcal{F}_i \), at least one of \( (\mathcal{F}_i) \), and \( (\mathcal{F}_i)' \), includes a tree
\( G = a\mathcal{F}_{i-2} \cup b\mathcal{F}_{i-3} \) for some \( \{a, b\} \subseteq \Sigma^* \) which is prefix-free. There are four cases depending on \( x \):

(i) If \( x \in 0\Sigma^* \), we have \( G \subseteq (\mathcal{F}_i)^* \) with \( a = 11 \) and \( b = 10 \).
(ii) If \( x = 1 \), we have \( G \subseteq (\mathcal{F}_i)^* \) with \( a = 11 \) and \( b = 10 \).
(iii) If \( x \in 10\Sigma^* \), then \( G \subseteq (\mathcal{F}_i)^* \) with \( a = 0 \) and \( b = 111 \).
(iv) If \( x \in 11\Sigma^* \), then \( G \subseteq (\mathcal{F}_i)^* \) with \( a = 0 \) and \( b = 10 \).

These are trivial consequences of the definition of \( \mathcal{F}_i \). The definition of \( P, \) now yields immediately that \( P_i(\mathcal{F}_{i+3}) \geq 2 \), proving that

\[
 r_i \geq 2/F_{i+3} \quad \text{for} \quad i \geq 1.
\]

4. The upper bound

We now show that \( r_i \leq 2/F_{i+3} \) for \( i \geq 1 \) by demonstrating that the "greedy algorithm" (which always asks the \( y \in \pi T \) which minimizes the value of \( \max(|T_1|, |T_2|) \)) is at least this efficient.

For notational convenience we shall use the variables \( a, b, c, \) etc., in \( \pi T \) to denote \( |T_1|/|T_2| \), etc., in addition to their usual meaning.

Let \( a \) denote the longest word in \( \pi T \) such that \( a > 1/2 \), (there is clearly only one), and let \( b, c \) denote \( a0, a1 \) in an order so that \( b \geq c \).

**Lemma 4.1.** One of \( y = a \) or \( y = b \) minimizes \( \max(y, 1 - y) \) for \( y \in \pi T \).

**Proof.** Let \( y \) be the word minimizing \( \max(y, 1 - y) \). If \( y > 1/2 \), then \( y \in \pi a \); \( y = a \) is the word in \( \pi a \) minimizing \( \max(y, 1 - y) \). If \( y < 1/2 \) then \( y \in \{z0, z1\} \) for some \( z \in \pi a \). But if \( y = z0 \) and \( z1 \in \pi a \), then \( z1 \) is closer to 1/2 than \( y \) since of two positive real numbers whose sum is less than one, the larger is always closer to 1/2. Thus for \( y < 1/2, \) \( y = b \) minimizes \( \max(y, 1 - y) \).

The previous lemma implies that the greedy algorithm will either use \( a \) or \( b \) as the next question: \( a \) if \( 1 - a > b \), and \( b \) if \( 1 - a \leq b \).

We need to introduce notation analogous to the \( P_i(T) \) notation which includes as a parameter the worst-case split obtainable in \( T \), because our analysis depends heavily on the fact that if one question yields a poor split, then the next question is guaranteed to do somewhat better. Let

\[
 R_i(s) = \text{lub}\{P_i(T) / |T| : P_i(T) \geq 2 \land P_i(T) / |T| = s\}
\]

denote the least upper bound on the fraction of \( T \) that \( B \) can constrain \( x_0 \) to lie in, given that \( P_i(T) \geq 2 \) and that the worst result of the first "greedy" question contains exactly \( s / |T| \) leaves. The domain of \( R_i \) is \( 1/2 \leq s \leq 2/3 \), since \( P_i(T) / |T| \) is always in this range (if \( a > 2/3 \), then \( b \geq a/2 > 1/3 \).
The game of $N$ questions on a tree

Theorem 4.2.

$$R_i(s) \leq \begin{cases} 2s/F_{i+2} & \text{for } 1/2 \leq s \leq F_{i+2}/F_{i+3} \\ 2(1-s)/F_{i+1} & \text{for } F_{i+2}/F_{i+3} \leq s \leq 2/3. \end{cases}$$

Proof. We first observe that the theorem implies that $1/F_{i+2} \leq R_i(s) \leq 2/F_{i+3}$ for $1/2 \leq s \leq 2/3$. The proof proceeds by induction on $i$. For $i = 1$ we obtain $R_1(s) \leq s$ for $1/2 \leq s \leq 2/3$ directly.

For larger $i$, the greedy algorithm first asks the question $y$ (here $y = a$ or $y = b$). Let $U$ denote the subtree of $T$ (either $T_y$ or $T'_y$) with size $s \cdot |T|$, $(T_a$ if $y = a, T'_b$ if $y = b$), and let $V$ denote the complement of $U$ with respect to $T$.

If $x_0 \in V$, then we can say that

$$R_i(s) \leq \frac{|V|}{|T|} \cdot \max_{1/2 = s \leq 2/3} (R_{i-1}(s)) \leq \frac{1}{2} \cdot \frac{2}{F_{i+2}} = \frac{1}{F_{i+2}}.$$ 

But $1/F_{i+2}$ is the minimum value obtained by the claimed upper bound for $R_i(s)$, so in this case the upper bound is correct.

On the other hand, if $x_0 \in U$, then we can argue that $P_i(U) \leq |V|$. If $y = a$, then $b < 1 \cdot a$, $U = T_a$, and $P_i(U) \leq |T_b| \leq |T'_b|$. Or if $y = b$, then $1 \cdot a = b$, $U = T'_b$, and $P_i(U) \leq |T_a| \leq |T_b|$ (remember that $b$ is larger than its brother $c$, so that $\max(c, 1-a) = 1-a$). In either case we have that

$$R_i(s) \leq s \cdot \max_{1/2 = s \leq 2/3} (R_{i-1}(t)),$$

since $|V|/|U| = (1-s)/s$. For $1/2 \leq s \leq F_{i+2}/F_{i+3}$ this directly yields

$$R_i(s) \leq s \cdot \max_{1/2 \leq t \leq 2/3} R_{i-1}(t) = 2s/F_{i+2}.$$ 

For $F_{i+2}/F_{i+3} \leq s \leq 2/3$ we obtain (since $(1-s)/s \leq F_{i+1}/F_{i+2}$)

$$R_i(s) \leq s \cdot \max_{1/2 \leq t \leq (1-s)/s} R_{i-1}(t) = s \cdot R_{i-1}((1-s)/s) = 2(1-s)/F_{i+1}.$$ 

This finishes the proof of the theorem.

The functions $R_1(s)$, $R_2(s)$, and $R_3(s)$ are plotted in Fig. 2.

Corollary. $r_i = 2/F_{i+3}$.

Thus, with two questions player $B$ can reduce the possibilities for $x_0$ by a factor of $2/5$, and so on. The efficiency of each question approaches the limit:

$$r = \lim_{i \to \infty} (r_i)^{1/i} = \emptyset^{-1} = 0.61803\ldots,$$

the inverse of the golden ratio $\emptyset$. 
We remark that although the greedy algorithm suffices to give us an upper bound on $r_n$, there exist trees for which the greedy algorithm is not the best strategy. The “greedy algorithm” is shown to perform very poorly in a similar testing situation in [1].

References