DIMACS Technical Report 98-35
September 1998

Do answers help in posing questions?[1]

by

Sarmad Abbasi
Dept. of Computer Science
Rutgers University
Piscataway, New Jersey 08855
email: sabbasi@paul.rutgers.edu

[1] This work is dedicated to Vicky Choi

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, Bellcore and NEC Research Institute.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

Let T_n denote a complete binary tree of depth n. Each internal node, v, of T_n has two children denoted by left(v) and right(v). Consider the following game between two Players, Paul and Carole. For each internal node, v, Carole chooses $X(v) \in \{\text{left}(v), \text{right}(v)\}$. This naturally defines a path from the root, λ, of T_n to one of its leaves given by

$$\lambda, X(\lambda), X^2(\lambda), \ldots, X^n(\lambda).$$

Paul has to find this path by asking questions of the form:

$$Q_v\text{:``Is } X(v) = \text{left}(v)\text{?''}$$

The game proceeds in r rounds. In every round Paul can k questions. Carole supplies the answers to these k questions in parallel. We give necessary and sufficient conditions for Paul to win this game.

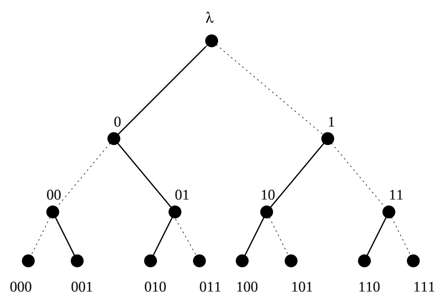

Figure 1: An instance of $G(3, k, r)$

1 Introduction

Combinatorial games have been extensively studied[2]. The topic of this paper is the investigation of a game $G(n, k, r)$ between two players. Let T_n denote a complete binary tree with n levels. Level 0 of T_n consists of a single node labeled by, λ, the empty string. In general, level i of T_n consists of 2^i nodes labeled by binary strings of length i. For any $i < n$ every node x in the i-th level has two children $\text{left}(x) = x0$ and $\text{right}(x) = x1$. The game $G(n, k, r)$ is played game between two players, Paul and Carole. For each internal node , v, Carole chooses $X(v) \in \{\text{left}(v), \text{right}(v)\}$. This naturally defines a path from the root of T_n to one of its leaves given by

$$P = \lambda, X(\lambda), X^2(\lambda), \ldots, X^n(\lambda).$$

where $X^0(v) = v$ and $X^i(v) = X(X^{i-1}(v))$ for $i \geq 1$.
Paul is allowed to ask questions of the form:

$$Q_v : \text{“Is } X(v) = \text{left}(v)\text{?”}$$

The game proceeds in rounds. In every round, Paul can asks at most k questions. These k questions can depend on all the answers provided by Carole in the previous rounds. Carole gives answers to these k questions in parallel. At the end of r rounds Paul wins if he is successful in determining P.

Figure 1 shows the tree T_3 as an instance of this game. The bold arrows emanating from each node denote choices made by Carole. The path determined by these choices is given $P = \lambda, 0, 01, 010$. Paul is required to find this path.

We will view $G(n, k, r)$ as a perfect information game between Paul and Carole. Hence, instead of Carole choosing $X(v) \in \{\text{left}(v), \text{right}(v)\}$ before the start of the game she will make her choice for $X(v)$ when confronted with the question Q_v. After r rounds, Paul wins if there is a unique paths consistent with Carole's answers.

Remark The names Paul and Carole have been suggested by Joel Spencer [4] for these type of searching games. Paul after the great Questioner-Paul Erdös. Carole maybe and thought of her acronym -Oracle.

It is interesting to contrast $G(n, r, k)$ with the k round version of Ulam's searching game

[5]. In Ulam's game, $U(n,k,r)$, Carole thinks of an x from the set $S = \{1,\dots,n\}$. Paul tries to find this "x" by asking questions of the form

$$Q_A\text{:"Is } x \in A\text{?"}$$

The game proceeds in r rounds and in each round Paul is allowed to ask k questions. After r rounds Paul wins if he can determine x. The outcome of this game only depends on the total number of questions asked; that is rk. One can easily show that Paul wins $U(n,k,r)$ if and only if $2^{kr} \geq n$. Ulam's game becomes much more interesting if Carole is allowed to lie at times. Very interesting results are obtained for Ulam's searching game with fixed number of lies [3, 4].

Ulam's game is often compared[4] with the classical "Twenty Questions." However, few would be willing to play Twenty Questions, if they were required to ask all of the twenty questions at once! What makes Twenty Questions interesting is that "answers do help in posing questions." The game $G(n,k,r)$ turns out to be more interesting in this regard: In some sense for $G(n,k,r)$ "answers help in posing questions."

The game $G(n,k,r)$ maybe considered as a restricted version of Ulam's game. The leafs of T_n are ordered in a natural way. For any node ,v, at level l the subtree rooted at v corresponds to an interval I_v of nodes given by

$$I_v = \{vw : w \in \{0,1\}^{n-l}\}.$$

The question Q_v corresponds to asking if x is among the first 2^{n-l-1} elements of I_v. If x belongs to I_v Carole has to reply truthfully, otherwise she can answer arbitrarily.

We also study the game $G(n,k_1,\dots,k_r)$ in which Paul is allowed to ask k_i questions in round i. We prove the following general theorem.

Theorem 1.1 *Paul wins $G(n,k_1,\dots,k_r)$ if and only if $\sum_{i=1}^{r}\lfloor\log(k_i+1)\rfloor \geq n$.*

As a corollary we obtain

Corollary 1.1 *Paul wins $G(n,k,r)$ if and only if $r \geq \frac{n}{\lfloor\log(k+1)\rfloor}$.*

The analysis of this game, in some sense, answers how much information per round can Paul extract out of Carole, if he has to ask k questions in parallel. In case of Ulam's game he can extract k bits of information. For $G(n,k,r)$ Theorem 1.1 states that he can obtain $\lfloor\log(k+1)\rfloor$ bit of information per round.

The remaining paper is organized as follows. In the next section we give a good strategy for Paul. In section 3 we prove that if $\sum_{i=1}^{r}\log(k_i+1) < n$ Carole wins $G(n,k_1,\dots,k_r)$. In section 4 we improve this to show that if $\sum_{i=1}^{r}\log\lfloor(k_i+1)\rfloor < n$ then Carole wins $G(n,k_1,\dots,k_r)$. In the last section we describe Chaudhuri's game $C(n,k,r)$ and point out some interesting problems.

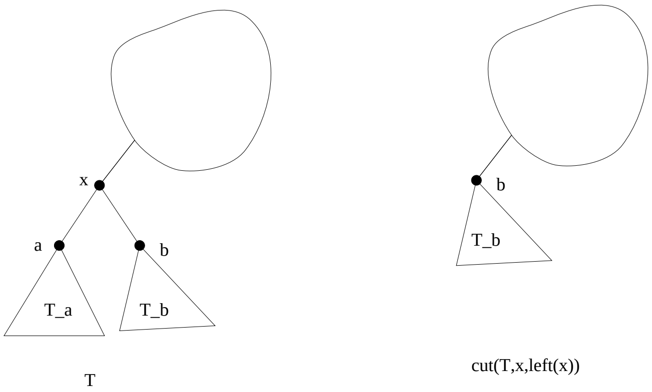

Figure 2: Cutting a tree

2 A Trivial Strategy

In this section we present a good strategy for Paul.

Lemma 2.1 *If* $\sum_{i=1}^{r} \log \lfloor (k_i + 1) \rfloor \geq n$ *then Paul wins* $G(n, k_1, \ldots, k_r)$.

Proof. The proof is by induction of r. For $r = 0$ there is nothing to prove. Now let us assume that $r \geq 1$. Let l be the largest integer such that $2^l - 1 \leq k_1$. Note that $l = \lfloor \log(k_1 + 1) \rfloor$. Paul asks the set of questions $\{Q_v : \text{level}(v) \leq l - 1\}$. Once he get the answers to these questions he has determined the first l nodes of the path P. Let u be the node on level l which is on the P. The game now is restricted to the subtree of T_n rooted at u. This subtree is isomorphic to T_{n-l} and $\sum_{i=2}^{r} \log \lfloor (k_i + 1) \rfloor \geq n - l$. Since, by induction Paul can win $G(n - l, k_1, \ldots, k_{r-1})$ he can win $G(n, k_1, \ldots, k_r)$. ∎

Consider Lemma 2.1 for $G(n, 2, r)$. It says that Paul can even win if $r \geq n$. This is quite trivial as Paul can win $G(n, 1, r)$ if $r \geq n$. However, we will show in the next section that he cannot do better.

3 An Adversary Strategy

In this section we describe an adversary strategy for Carole. The idea is that Carole will keep track of Paul's progress. In each round she will answer the questions posed by Paul in such a way that he makes as little progress as possible. To make our plan work we need to measure the progress made by Paul during the course of the game. Towards this end, it is convenient for us to introduce a new game.

Let T be a (not necessarily a complete) binary tree; that is, each internal node of T two children denoted by left(v) and right(v). First we describe an operation on T. Let x be an internal node of T, $a = \text{left}(x)$ and $b = \text{right}(x)$. $T' = \text{cut}(T, x, \text{left}(x))$ denotes a tree constructed as follows. We cut the subtree rooted at a from T. Next we identify

b with x (See Figure 2.) In particular the label x no longer appears in $\mathrm{cut}(T, x, \mathrm{left}(x))$. Another operation $\mathrm{cut}(T, x, \mathrm{right}(x))$ is defined identically. For technical reasons we define $\mathrm{cut}(T, x, \mathrm{left}(x) = \mathrm{cut}(T, x, \mathrm{right}(x) = T$, if the x is not an internal node of T.

Let S be a set of internal nodes of T and A be answers to S; that is $A(v) \in \{\mathrm{left}(v), \mathrm{right}(v)\}$ for all $v \in S$. We define $\mathrm{cut}(T, A)$ as follows. Let us an arbitrary ordering of $S = \{v_1, \ldots, v_s\}$. Let $T_0 = T$ and define

$$T_i = \mathrm{cut}(T_{i-1}, v_i, A(v_i)).$$

We define

$$\mathrm{cut}(T, A) = T_s.$$

It can be shown that $\mathrm{cut}(T, A)$ is independent of the choice of the ordering of S.

Paul and Carole can now play a game $G(T, k_1, \ldots, k_r)$ as follows. If T is a single node Paul is declared the winner without any moves. Otherwise, Paul presents Carole with a set, S, of at most k_1 internal nodes of T. Carole returns a set A of answers to S; that is, $A(v) \in \{\mathrm{left}(v), \mathrm{right}(v)\}$ for every $v \in S$. Now Paul and Carole play the game $G(\mathrm{cut}(T, A), k_2, \ldots, k_r)$. The winner of this game is declared the winner of $G(T, k_1, \ldots, k_r)$.

For a tree T we define the weight of any node in the tree as follows:

$$W_v = \begin{cases} 1, & \text{if } v \text{ is a leaf node} \\ W_{\mathrm{left}(v)} + W_{\mathrm{right}(v)} & \text{Otherwise.} \end{cases}$$

With the above definition the weights of the nodes can be computed in a bottom up fashion. The weight of every v node is the number of leaf nodes in contained in the subtree rooted at v. We define, $W(T)$, the weight of the tree, T, to be the weight of the root of T. It turns out that $W(T)$ is a good way to measure the progress made by Paul.

We start with the following simple facts:

Lemma 3.1 *$W(T) = 1$ if and only if T consists of a single node.* ∎

Lemma 3.2 *Paul wins $G(n, k_1, \ldots, k_r)$ if and only if he wins $G(T_n, k_1, \ldots, k_r)$.* ∎

In the next lemma we show that Carole can limit the progress made by Paul in one round.

Lemma 3.3 *Let T be any tree and S be any set of at most k internal nodes of T. Carole can find answers A to S such that*

$$W(\mathit{cut}(T, A)) \geq \frac{W(T)}{k+1}.$$

Proof. We prove this lemma by induction on k. Let $k \geq 1$ we assume that we have proved the lemma for all $k' < k$. We prove the lemma for k by induction on the number of nodes in T. We may assume that $|S| = k$ otherwise we can apply induction on k. Let λ be the root of T. We let $a = \mathrm{left}(\lambda)$ and $b = \mathrm{right}(\lambda)$. Let T_a and T_b denote the trees rooted at a and b

respectively. We also let S_a (resp. S_b) be the nodes of S which belong to the tree rooted at a (resp. b). Define $k_a = |S_a|$ and $k_b = |S_b|$. We consider three cases.
Case 1 $\lambda \in S$. Carole defines $A(\lambda) = a$ if

$$\frac{W_a}{k_l + 1} \geq \frac{W_b}{k_r + 1} \tag{1}$$

and $A(\lambda) = b$ otherwise. Without loss of generality we assume (1) holds and she chooses $A(\lambda) = a$. By induction she can answer the k_l questions with A_a so that

$$W(\text{cut}(T_a, A_a)) \geq \frac{W_a}{k_l + 1}.$$

She defines $A(v) = A_a(v)$ for all $v \in S_a$. Finally she choose $A(v)$ for every $v \in S_b$ arbitrarily. With this choice of A we have

$$W(\text{cut}(T, A)) = W(\text{cut}(T_a, A_a)).$$

As $k_l + k_r = k - 1$ and (1) holds, a simple calculation gives the desired result.

Case 2 $x \notin S$. Furthermore, we assume that $k_l > 0$ and $k_r > 0$. By induction on k there is a choice of answers A_a (resp. A_b) for S_a (resp. S_b) such that,

$$W(\text{cut}(T_a, A_a)) \geq \frac{W_a}{k_l + 1}, \left(\text{resp. } W(\text{cut}(T_b, A_b)) \geq \frac{W_b}{k_r + 1} \right).$$

We let $A(v) = A_a(v)$ for $v \in S_a$ and $A(v) = A_b(v)$ for $v \in S_b$. We have,

$$W(\text{cut}(T, A)) = W(\text{cut}(T_a, A_a)) + W(\text{cut}(T_b, A_b)).$$

Since $k_r + k_l = k$ a simple calculation completes the proof.
Case 3 $x \notin S$ and either $k_l = 0$ or $k_r = 0$. The result now follows from induction on n and a simple calculation. We omit the details. ∎

Applying the above lemma we obtain the main result of this section.

Lemma 3.4 *If* $\sum_{i=1}^{r} \log(k_i + 1) < n$ *then Carole wins* $G(n, k_1, \ldots, k_r)$. ∎

4 The Balanced Generosity of Carole

Lemma 3.4 is a partial converse of Lemma 2.1. In this section we improve this bound and obtain a complete converse of Lemma 2.1. Let λ be the root of T and a and b be its two children. If $W_a \ll W_b$ then Paul can try to make a lot of progress by asking more questions in the subtree rooted at b. The main idea is that Carole will try to keep the tree as balanced as possible. Thus preventing Paul from making more progress. However, for Paul to exploit such a imbalance he has to first create one. Carole will not let Paul create such an imbalance as she would "generously" answer some questions that he has not even posed.

Note that a complete binary tree, T_n, is balanced in the following sense. For any internal node x of we have

$$W(\text{left}(x)) = W(\text{right}(x)).$$

Hence we shall refer to these trees as balanced trees.

We start by defining a new game $G'(T, k_1, \ldots, k_r)$. This game is the same as $G(T, k_1, \ldots, k_r)$ except that at the end of each round, after answering Paul's questions, Carole can declare answers to zero or more questions Q_v that Paul has not inquired about. This generosity of Carole is, of course, not directed at Paul. It is directed at us since this makes the analysis of our game much easier.
The following Lemma is immediate.

Lemma 4.1 *If Carole wins $G'(n, k_1, \ldots, k_r)$ then Carole wins $G(n, k_1, \ldots, k_r)$.*

At the beginning of the the game $G(T_n, k_1, \ldots, k_r)$ the tree T_n is balanced. When Paul will presents Carole with a set S of at most k_1 questions. Carole will provide Paul with the answers A to questions $S \cup Z$ for some set Z. Her answers will guarantee that the $\text{cut}(T_n, A)$ is balanced. Furthermore, the progress of Paul is limited. To show that Carole can achieve this we have to prove the following two results.

Lemma 4.2 *Let S be any set of at most $2^s - 1$ internal nodes of T_n. Carole can find a set Z and answers, A, for $S \cup Z$ such that*

$$W(cut(T_n, A)) = \frac{W(T_n)}{2^s}.$$

Furthermore, $cut(T_n, A)$ is a balanced binary tree. ∎

Lemma 4.2 can be proved by induction on s. The proof is is simpler and also similar to the proof of the next lemma. Therefore, it is omitted.

Lemma 4.3 *Let S be any set of at most $2^s - 2$ internal nodes of T_n. Carole can find a set Z and of answers, A, for $S \cup Z$ such that*

$$W(cut(T_n, A)) = \frac{W(T_n)}{2^{s-1}}.$$

Furthermore, $cut(T_n, A)$ is a balanced binary tree.

Proof. We assume that Lemma 4.2 is true. We prove this lemma by induction on s. If $s = 1$ then there is nothing to prove. Let $s \geq 2$ and $|S| = 2^{s-1} - 2$. Let λ be the root of T_n. Define $a = \text{left}(x)$ and $b = \text{right}(x)$. We let S_a (resp. S_b) denote the nodes of S which belong to T_a (resp. T_b).

We have to consider two cases.

Case 1 $|S_a| \leq 2^{s-1} - 2$ or $|S| \leq 2^{s-1} - 2$.

Without loss of generality we may assume that

$$|S_a| \leq 2^{s-1} - 2.$$

By induction Carole can find a set Z_a and answers A_a to $S_a \cup Z_a$ such that

$$W(\mathrm{cut}(T_a, A_a)) = \frac{W_a}{2^{s-2}}$$

and $W(\mathrm{cut}(T_a, A_a))$ is balanced binary tree. We now let $A(x) = \mathrm{left}(x) = a$ and $A(v) = A_a(v)$ for all $v \in S_a \cup Z_a$. We choose $A(v) \in \{\mathrm{left}(v), \mathrm{right}(v)\}$ for $v \in S_b$ arbitrarily. Lastly, we choose $A(x) = \mathrm{left}(x) = a$; even if $x \notin S$.

Consider $\mathrm{cut}(T_n, A)$, it is equal to $\mathrm{cut}(T_a, A_a)$. Hence it is balanced. Furthermore,

$$W(\mathrm{cut}(T, A)) = W(\mathrm{cut}(T_a, A_a)) = \frac{W_a}{2^{s-2}}.$$

Since $W_x = 2W_a$ for balanced binary trees the result follows. Notice that we have chosen an answer for x even if $x \notin S$. This generosity is precisely what helps Carole to maintain the balance.

Case 2 $|S_a| = |S_b| = 2^{s-1} - 1$.

Since we have $|S| = 2^s - 2$ hence $x \notin S$. By Lemma 4.2 there is a set Z_a and answers A_a to $S_a \cup Z_a$ such that,

$$W(\mathrm{cut}(T_a, A_a)) = \frac{W_a}{2^{s-1}}.$$

and $\mathrm{cut}(T_a, A_a)$ is balanced binary tree. Similarly there is a Z_b and answers A_b to $S_b \cup Z_b$ such that,

$$W(\mathrm{cut}(T_b, A_b)) = \frac{W_b}{2^{s-1}}.$$

Again $\mathrm{cut}(T_b, A_b)$ is balanced.

Now the critical point is that

$$W(\mathrm{cut}(T_b, A_b)) = W(\mathrm{cut}(T_b, A_b)).$$

Now if we define $A(v) = A_a(v)$ for $v \in S_a \cup Z_a$ and $A(v) = A_b(v)$ for $v \in S_b \cup Z_b$ then $\mathrm{cut}(T, A)$ is balanced. Furthermore,

$$W(\mathrm{cut}(T, A)) = W(\mathrm{cut}(T_a, A_a)) + W(\mathrm{cut}(T_b, A_b)).$$

Since

$$W(T) = W_\lambda = W_a + W_b = W(T_a) + W(T_b)$$

the lemma follows. ∎

Now we can obtain the a full converse of Lemma 2.1.

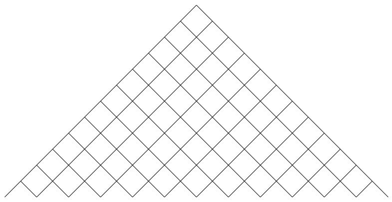

Figure 3: Chaudhuri's Game

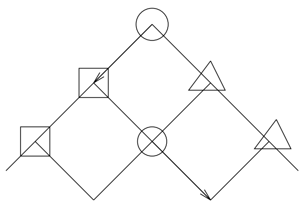

Figure 4: A Strategy of $C(k, 2, r)$

Lemma 4.4 *If* $\sum_{i=1} \lfloor \log(k_i + 1) \rfloor < n$ *then Carole wins* $G'(T_n, k_1, \ldots, k_r)$.

Proof. We may assume that $k_i = 2^{s_i} - 2$. Otherwise, we increase k_i. Now if Carole follows the strategy described in Lemma 4.3 in every round she wins $G'(T_n, k_1, \ldots, k_r)$. ∎

5 Another Interesting Game

The game $G(n, k, r)$ is inspired by another game of Shiva Chaudhuri[1]. We describe briefly describe this game. Chaudhuri's game $C(n, k, r)$ is played on mesh which is cut into half diagonally (See Figure 3). Hence, there are n levels. Each level i consists of i nodes labeled $(i, 1), \ldots, (i, i)$. Two nodes (i, a) in level i is connected to two nodes $(i+1, a)$ and $(i+1, a+1)$ in level $i+1$. Once again for every node v at level i $(i < n)$, Carole chooses one of its neighbor $X(v)$ at level $i + 1$. This define a path from node $(1, 1)$ to one of the nodes in the last level. Paul has to find this path by asking questions of the form
Q_v: "What is $X(v)$?"

Again the game has r rounds and Paul may ask k questions per round. For this game only an analog of Lemma 2.1 is known.

Lemma 5.1 ([1]) *Let* $k = \frac{l(l+1)}{2}$. *If* $r \geq \frac{n}{l}$ *then Paul wins* $C(n, k, r)$. ∎

No strategy, for Carole is known which shows that for some $r = \frac{n}{f(k)}$ Carole can win the game, where $f(k) = o(k)$.

On the other hand if $k = 2$ one can show that Paul can win $C(n, 2, r)$ in if $r > \frac{2n}{3}$. To see this, consider figure 4. We first ask the questions regarding the nodes that are circles in the figure. Now depending on the answer of the root node we ask two left most questions on level 2 and 3. This way we can move down three levels in two rounds. Whether this is the best possible strategy for $k = 2$ is also open.

Acknowledgments: I would like to thank Shiva Chaudhuri for bringing this problem to my attention. I would also like to thank E. Szemerédi for his constant encouragement and guidance.

References

[1] S. P. Chaudhuri, 1997. personal communication.

[2] A. Fraenkel. Combinatorial games. *Electronic Journal of Combinatorics.*, Dynamic Surveys:1–45, 1996.

[3] A. Plec. Ulam's problem on searching with a lie. *J. Combin. Thoery. Ser. A.*, 44:129–140, 1987.

[4] J. Spencer. Ulam's searching game with fixed number of lies. *Theoretical Computer Science*, 95:307–321, 1992.

[5] S. Ulam. *Adventures of a Mathematician.* Sribners, New York, 1977.