Effect of parallelism on the efficiency of binary tree search

Sarmad Abbasi
Department of Computer Science
Sukkur Institute of Business Administration
Airport Road
Sukkur 65200
Sindh, Pakistan
email: sarmad_abbasi@iba-suk.edu.pk

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). Let f be a function mapping each internal node, v, to $\{\text{left}(v), \text{right}(v)\}$. This naturally defines a path from the root, λ, of T_n to one of its leaves given by

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

We consider the problem of finding this path via a deterministic algorithm that probes the values of f in *parallel*. We show that any algorithm that probes k values of f in one round requires $\frac{n}{\lfloor \log(k+1) \rfloor}$ rounds in the worst case. This indicates that the amount of information that can be extracted in parallel is, at times, strictly less than the amount of information that can be extracted sequentially.

keywords: combinatorial games, two player perfect information games, adversary strategies.

1 Introduction

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, the i-th level 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 left$(x) = x0$ and right$(x) = x1$. Let f be a function mapping each internal node, v, to $\{\text{left}(v), \text{right}(v)\}$. This function naturally defines a path from the root of

T_n to one of its leaves given by:

$$P_f = \lambda, f(\lambda), f^2(\lambda), \ldots, f^n(\lambda),$$

where $f^0(v) = v$ and $f^i(v) = f(f^{i-1}(v))$ for $i \geq 1$.

Consider the problem of finding P_f via a sequential algorithm that probes the values of f. An algorithm can probe the values of $f(\lambda), \ldots, f^n(\lambda)$ one by one and find P_f in exactly n probes. Moreover, it is also easy to see that this is an optimal algorithm. In this paper, we investigate the problem when the values of f have to be probed in *parallel*. Thus, the parallel algorithm has to proceed in rounds. In each round it is allowed to probe k values of f. We show that any such algorithm requires

$$\frac{n}{\lfloor \log(k+1) \rfloor}$$

rounds to determine P_f in the worst case. Furthermore, this bound is optimal as a very simple algorithm can achieve it. Our result can be interpreted in terms of information extraction. It states that, for this problem, probing k values of f in parallel extracts $\lfloor \log(k+1) \rfloor$ bits of information; whereas, probing k values sequentially extracts k bits of information.

Any algorithm that correctly finds P_f in r rounds, in the worst case, must also find P_f in r rounds if an adversary were to choose the values of f *on the fly*. This observation allows us to view this problem as a combinatorial game between two players; Paul (the algorithm) and Carole (the adversary). We define $G(n,k,r)$ as a perfect information game between Paul and Carole. The game is played on the complete binary tree with n levels and consists of r rounds. Every round of the game consists of the following steps:

1. Paul selects a set $S \subseteq I_n$, where I_n is the set of internal nodes of T and $|S| \leq k$.

2. Carole declares the value of $f(v)$ for every $v \in S$.

At the end of r rounds, Paul wins if he is successful in determining P_f; that is, there is a unique path that is consistent with Carole's answers.

As Paul is only allowed to ask questions of the form "What is $f(v)$?" in this game, we will often say that Paul presents Carole with a node v (or equivalently he presents Carole with a set of nodes).

Remark: Joel Spencer[3] has suggested that the players in these types of search games be called Paul and Carole: Paul represents the great questioner Paul Erdös; whereas, Carole, being a permutation of "oracle", represents the notoriously obtuse oracle of Apollo at Delphi.

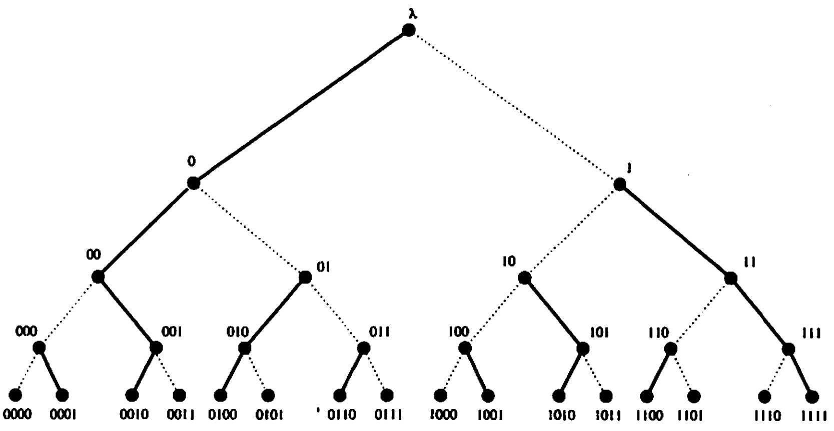

Figure 1: An instance of $G(4, k, r)$

It is interesting to contrast $G(n, r, k)$ with the k round version of Ulam's searching game [4]. In Ulam's game, $U(n, k, r)$, Carole thinks of an x from the set $S = \{1, \ldots, n\}$. Paul tries to find this "x" by asking questions of the form:

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

where A can be any subset of $\{1, \ldots, n\}$. 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 depends only 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 known for Ulam's game with fixed number of lies [2, 3].

Ulam's game is often compared[3] with the classical "TWENTY QUESTIONS." Note that 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 help in posing questions." The game $G(n, k, r)$ also turns out to be more interesting in this regard: for $G(n, k, r)$ answers *do* help in posing questions.

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

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

By taking $k_i = k$ for all $i = 1,\dots,r$ and observing that $G(n,k,r) = G(n,\underbrace{k,\cdots,k}_{r \text{ times}})$ we obtain:

Theorem 1.2 *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 only $\lfloor\log(k+1)\rfloor$ bits of information.

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 $\sum_{i=1}^{r}\lfloor\log(k_i+1)\rfloor < n$ 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.

2 A Trivial Strategy

In this section we present a good strategy for Paul. Note that Paul wins $G(0,k_1,\dots,k_r)$ for any sequence $k_1,\dots,k_r \geq 0$. This is because the trivial tree of depth 0 contains a unique path from the root to the only leaf (which is also the root) and Paul can declare this path without asking any questions. In case $r = 0$ Paul wins the game if $n = 0$; otherwise, Carole wins the game. Now we can state the following simple result:

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

Proof. The proof is by induction on r. For $r = 0$ there is nothing to prove as n must be 0. Assume that $r \geq 1$ and let l be the largest integer such that $2^l - 1 \leq k_1$. Note that $l = \lfloor\log(k_1+1)\rfloor$. Paul selects the set of questions $\{v : \text{level}(v) \leq l-1\}$. Once he gets the answers to these questions, he has determined the first l nodes of the path P_f. Let u be the node on level l that lies on the path P_f. 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 Paul can win $G(n-l,k_1,\dots,k_{r-1})$ by induction, he can win $G(n,k_1,\dots,k_r)$. □

The above lemma seems to say nothing about $G(n,2,r)$. As Paul can win $G(n,1,r)$ if $r \geq n$, he can clearly win $G(n,2,r)$ if $r \geq n$. However, we will show in the Section 4 that this result is the best possible.

3 An Adversary Strategy

In this section we will prove a partial converse of Lemma 2.1. Towards this end, we describe an adversary strategy for Carole. When Carole plays the adversary strategy she does not decide on the function f at the beginning of the game. Instead she constructs it *on the fly*. At the end of the game she wins if there are more than one possible paths from the root to a leaf that are consistent with her answers. It is not hard to see that if Paul can win $G(n, k_1, \ldots, k_r)$ then he can win even if Carole plays an adversary strategy.

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 her plan work, she needs to measure the progress made by Paul as the game proceeds. She will maintain a set A_i consisting of all the questions that have been posed by Paul and answered by her up to round i of the game. Initially, A_0 is empty. After every round some new internal nodes are added to it. Thus, $\emptyset = A_0 \subseteq A_1 \cdots \subseteq A_r$ is a sequence of sets. The function f is constructed during the course of the game. Hence, we view it as a sequence: $f_0, f_1, \ldots, f_r : I_n \mapsto \{\text{left}, \text{right}, *\}$, where

$$\begin{array}{lcll} f_i(A_i) & \subseteq & \{\text{left}, \text{right}\}, & \\ f_i(I_n \setminus A_i) & = & \{*\}, & \text{and} \\ f_i(v) & = & f_{i-1}(v) & \text{for all } v \in A_{i-1} \text{ and for all } i = 1, \ldots, r. \end{array}$$

Round i of the game consists of the following steps:

1. Paul selects a set $S \subseteq I_n \setminus A_{i-1}$, where $|S| \leq k_i$; that is he sets
$$A_i := A_{i-1} \cup S.$$
2. For every $v \in S$, Carole assigns the value of $f_i(v)$ to either left(v) or right(v).

We say that a path $\lambda = v_0, v_1, \ldots, v_s$ is *consistent* with an f_i if

$$f_i(v_j) \neq * \text{ implies } v_{j+1} = f_i(v_j) \text{ for } j = 1, \ldots, s-1.$$

After the end of the i-th round, the pair (A_i, f_i) completely determines the *state* of the game. We would like to measure the progress of Paul at this point. We define the weight of each node, v, with respect to (A_i, f_i), as follows:

$$W_v^i = \begin{cases} 1 & \text{if } v \text{ is a leaf node,} \\ W_{\text{left}(v)}^i + W_{\text{right}(v)}^i & \text{if } v \notin A_i, \\ W_{f_i(v)}^i & \text{if } v \in A_i. \end{cases}$$

With the above definition the weights of the nodes can be computed in a bottom up fashion. The weight of a node, v, is the number of paths, starting from v and ending at a leaf node, that are consistent with the function f_i. We define the weight of the game, $W^i(T_n)$, with respect to (A_i, f_i), to be the weight of the root of T_n. It turns out that $W^i(T_n)$ is a good way to measure the progress made by Paul. We start with the following simple fact:

Lemma 3.1 *Paul wins $G(n, k_1, \ldots, k_r)$ if and only if at the end of r rounds*

$$W^r(T_n) = 1.$$

□

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

Lemma 3.2 *Suppose the game is in state (A_i, f_i) after i rounds and $W^i(T_n)$ is its weight with respect to (A_i, f_i). Let $S \subseteq I_n \backslash A_i$ be any set of k questions posed by Paul in the $(i+1)$-st round. Carole can answer these questions in such a way that the game ends in a state (A_{i+1}, f_{i+1}) and the weight $W^{i+1}(T_n)$, with respect to (A_{i+1}, f_{i+1}), satisfies*

$$W^{i+1}(T_n) \geq \frac{W^i(T_n)}{k+1}.$$

Proof. We prove this lemma by induction on k. The base case ($k = 0$) is trivial. Let $k \geq 1$. Assume that the lemma is true for all $k' < k$. We prove the lemma for k by induction on n. Let λ be the root of T_n. Let T_a and T_b denote the trees rooted at a and b respectively, where $a = \text{left}(\lambda)$ and $b = \text{right}(\lambda)$. Also, let S_a (resp. S_b) be the nodes of S which belong to the tree rooted at a (resp. b). Define $k_l = |S_a|$ and $k_r = |S_b|$. We consider three cases:

Case 1 $\lambda \in S$. Carole defines

$$f_{i+1}(\lambda) = \begin{cases} a \text{ if } \frac{W_a^i}{k_l+1} \geq \frac{W_b^i}{k_r+1}, \\ b \text{ otherwise.} \end{cases}$$

Without loss of generality, we assume she chooses $f_{i+1}(\lambda) = a$. By induction on k she can define f_{i+1} on S_a so that

$$W_a^{i+1} \geq \frac{W_a^i}{k_l+1}.$$

She extends f_{i+1} to S_b arbitrarily. With these choices

$$W^{i+1}(T_n) = W^{i+1}_\lambda = W^{i+1}_a \geq \frac{W^i_a}{k_l + 1}.$$

Now,

$$\begin{aligned} W^i(T_n) &= W^i_a + W^i_b \\ &\leq W^i_a + \left(\frac{k_r + 1}{k_l + 1}\right) W^i_a \\ &= \left(\frac{k_r + k_l + 2}{k_l + 1}\right) W^i_a \\ &= \left(\frac{k + 1}{k_l + 1}\right) W^i_a, \end{aligned}$$

where the last equality holds since $k_l + k_r = k - 1$.

Case 2 $\lambda \notin S$. Assume that $k_l > 0$ and $k_r > 0$. By induction on k Carole can extend f_i to S_a (resp. S_b) such that,

$$W^{i+1}_a \geq \frac{W^i_a}{k_l + 1}, \left(\text{resp. } W^{i+1}_b \geq \frac{W^i_b}{k_r + 1}\right).$$

By making these choices,

$$\begin{aligned} W^{i+1}(T_n) &= W^{i+1}_a + W^{i+1}_b \\ &\geq \frac{W^i_a}{k_l + 1} + \frac{W^i_b}{k_r + 1} \\ &\geq \frac{W^i_a}{k + 1} + \frac{W^i_b}{k + 1} \\ &= \frac{W^i(T_n)}{k + 1}. \end{aligned}$$

Case 3 $\lambda \notin S$ and either $k_l = 0$ or $k_r = 0$. The result now follows from induction on n and a simple calculation. □

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

Lemma 3.3 *If* $\sum_{i=1}^r \log(k_i + 1) < n$ *then Carole wins* $G(n, k_1, \ldots, k_r)$.

Proof. When the game starts the weight of T_n is 2^n. If Carole plays the strategy given in Lemma 3.2 then after r rounds the game ends in a state (A_r, f_r), the weight of the game is at least

$$\frac{2^n}{\prod_{i=1}^r (k_i + 1)} > 1$$

and she wins the game. □

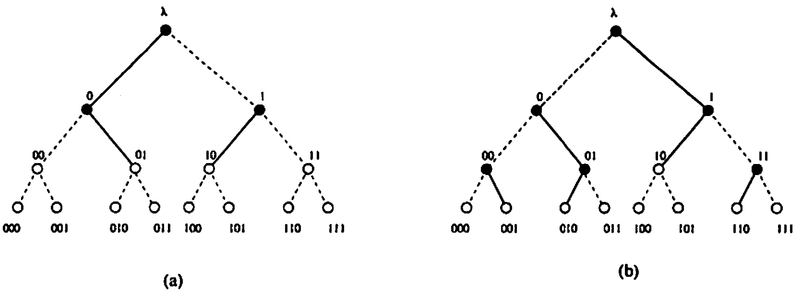

Figure 2: $G(n,3,r)$ versus $G(n,6,r)$: dark nodes indicate the questions posed by Paul. (a) Paul proceeds 2 levels in one round. (b) Carole thwarts an obvious attempt by Paul to proceed 3 levels by asking 6 questions.

Corollary 3.1 *If* $r < \frac{n}{\log(k+1)}$ *then Carole wins* $G(n,k,r)$. □

This simple argument is good enough to solve the problem completely for the case when the number of questions is $2^m - 1$.

Corollary 3.2 *If* $k = 2^m - 1$ *for some integer* $m \geq 0$ *then Paul wins* $G(n,k,r)$ *if and only if*

$$r \geq \frac{n}{\log(k+1)} = \frac{n}{m}. \quad \square$$

4 The Balanced Generosity of Carole

Let $m \geq 0$ be an integer. Let us compare the game when Paul asks $k_1 = 2^m - 1$ questions per round versus when he asks $k_1' = 2^{m+1} - 2 = 2(2^m - 1)$ questions per round. As $\lfloor \log(k_1 + 1) \rfloor = \lfloor \log(k_1' + 1) \rfloor = m$, Lemma 2.1 merely says that Paul can win both games in $\lceil \frac{n}{m} \rceil$ rounds. However, in the second case he is asking twice as many questions per round. A naive attempt to improve the trivial strategy by asking k_1 questions in level m is easily thwarted by Carole (Figure 2). Is it possible for Paul to use the extra k_1 questions to extract more information from Carole by using some sophisticated method? Intuitively the answer seems to be "No". This section is devoted to showing that this intuition is indeed correct; there is no way for Paul to use these extra questions towards his advantage.

Consider the game $G(n,k,r)$ after i rounds in a state (A_i, f_i). Let λ be the root of T and a and b be its two children. Suppose the state (A_i, f_i) is imbalanced in the sense that $W_a^i \ll W_b^i$. In this case, Paul can make a lot of progress by asking more questions in the subtree rooted at b. However, for Paul to exploit such an imbalance he has to create one first. To create an imbalance, Paul must deviate from "the optimal" strategy and this gives an advantage to Carole. The main idea is that Carole will not allow Paul to create such an imbalance in the weights. To achieve this goal, every time Paul tries to create an imbalance, she will "generously" answer some questions that he has not even posed. Carole can afford this generosity because Paul is pursuing a suboptimal strategy. Thus Carole's generosity is not directed towards Paul, it is just a book-keeping tool that helps in establishing the converse of Lemma 2.1.

Note that at the beginning of the game, the state (A_0, f_0) is balanced in the following sense: for any internal node v we have

$$W^0_{\text{left}(v)} = W^0_{\text{right}(v)}.$$

Now consider a state (A_i, f_i). We call a node, v, *reachable* if the path $\lambda = v_0, v_1, \ldots, v_r = v$ from the root to v is consistent with f_i; that is

$$v_t \in A_i \Rightarrow f(v_t) = v_{t+1} \text{ for } t = 0, \ldots, r-1.$$

An internal node, v, is called *open* if it is reachable and $v \notin A_i$. We will call a state (A_i, f_i) *balanced* if for every open node, v, we have

$$W^i_{\text{left}(v)} = W^i_{\text{right}(v)}.$$

A node x is called the *pseudo-root* of T_n if

1. x is open.

2. all predecessors of x are not open.

Figure 3 illustrates these definitions. If x is a pseudo-root of T_n in round i then it is clear that in all the subsequent rounds the weight of the root λ is the same as that of x. More formally,

$$W^j_\lambda = W^j_x \text{ for } j \geq i.$$

We now define a new game $G'(n, k_1, \ldots, k_r)$. This game is the same as the previously defined game $G(n, k_1, \ldots, k_r)$ except that Carole is also allowed to declare the answers to zero or more questions that Paul has not inquired about. More precisely, the i-th round of the game proceeds as follows:

1. Paul selects a set $S \subseteq I_n \setminus A_{i-1}$, where $|S| \leq k_i$.

2. Carole selects a (possibly empty) set $Z \subseteq I_n \setminus (A_{i-1} \cup S)$; that is, she sets
$$A_i := A_{i-1} \cup S \cup Z.$$
For every $v \in Z \cup S$ Carole assigns $f_i(v)$ to either left(v) or right(v).

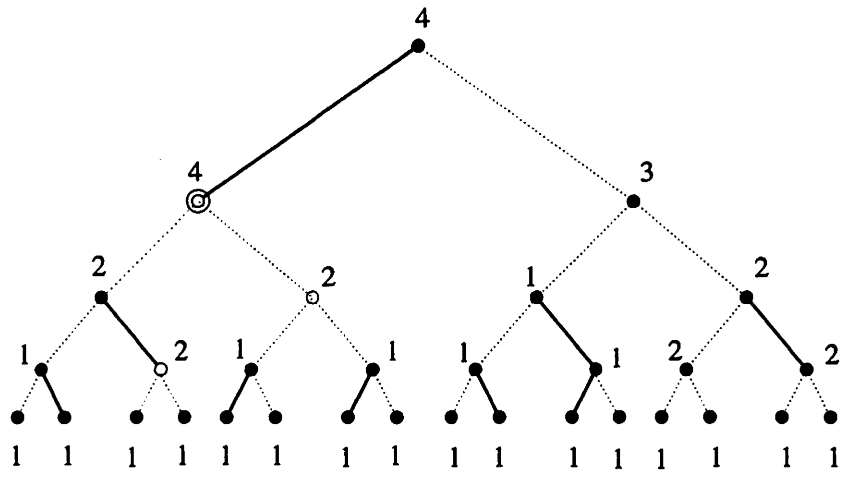

Figure 3: A game in a balanced state: white nodes are open, grey nodes are reachable but not open and the pseudo-root is double circled.

The following lemma is immediate.

Lemma 4.1 *Carole can win $G'(n, k_1, \ldots, k_r)$ if and only if Carole wins $G(n, k_1, \ldots, k_r)$.* □

Lemma 4.2 *Suppose the game is in a* **balanced state** *(A_i, f_i) after i rounds and $W^i(T_n)$ is its weight with respect to (A_i, f_i). Let $S \subseteq I_n \setminus A_i$ be any set of at most $2^m - 1$ questions posed in the $(i+1)$-st round. Carole can find a set Z and answers to the questions in $S \cup Z$ in such a way that the game ends in a* **balanced state** *(A_{i+1}, f_{i+1}). Furthermore, the weight $W_n^{i+1}(T_n)$, with respect to (A_{i+1}, f_{i+1}), satisfies*

$$W^{i+1}(T_n) = \frac{W^i(T_n)}{2^m}.$$

Proof. We prove this lemma by induction on m. If $m = 0$ then there is nothing to prove. Let $m \geq 1$ and $|S| \leq 2^m - 1$. Let x be the root pseudo-root of T_n. Since

$$W^j(T_n) = W^j_\lambda = W^j_x \text{ for all } j \geq i,$$

we can focus our attention on the subtree rooted at x. 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). It is easy to see that either $|S_a| \leq 2^{m-1} - 1$ or $|S_b| \leq 2^{m-1} - 1$. Without loss of generality we may assume that

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

By induction, Carole can find a set Z_a and answers A_a to the questions in $S_a \cup Z_a$ in such a way that

$$W_a^{i+1} = \frac{W_a^i}{2^{m-1}}$$

and T_a is in a balanced state; that is,

$$W_{\text{left}(v)}^{i+1} = W_{\text{right}(v)}^{i+1} \text{ for all open nodes, } v, \text{ of } T_a. \qquad (1)$$

Carole still has to answer the questions in S_b. She chooses $f_{i+1}(v) \in \{\text{left}(v), \text{right}(v)\}$ for $v \in S_b$ arbitrarily. Lastly, she defines $f_{i+1}(x) = \text{left}(x) = a$, regardless of whether Paul asks about x or not.

In the state (A_{i+1}, f_{i+1}) all the nodes in T_b are unreachable. Furthermore, x is no longer open and the nodes belonging to T_a satisfy Equation (1). Hence (A_{i+1}, f_{i+1}) is a balanced state. Lastly, we observe that as (A_i, f_i) was balanced, we have $W_x^i = 2W_a^i$. As $f_{i+1}(x) = a$,

$$W_x^{i+1} = W_a^{i+1} = \frac{W_a^i}{2^{m-1}} = \frac{W_x^i}{2^m}.$$

Here the second last equality holds by induction. Notice that Carole has chosen an answer for x *even* if $x \notin S$ and this generosity is *precisely* what helps Carole keep (A_{i+1}, f_{i+1}) balanced. □

The above lemma can be strengthened. It remains true even if the number of questions posed in the $(i+1)$-st round is doubled. Lemma 4.2 is needed in the proof of the following stronger lemma:

Lemma 4.3 *Suppose the game is in a* **balanced state** (A_i, f_i) *after i rounds and $W^i(T_n)$ is its weight with respect to (A_i, f_i). Let $S \subseteq I_n \setminus A_i$ be any set of at most $2^{m+1} - 2$ questions posed in the $(i+1)$-st round. Carole can find a set Z and answers to the questions in $S \cup Z$ in such a way that the game ends in a* **balanced state** (A_{i+1}, f_{i+1}). *Furthermore, the weight $W_n^{i+1}(T_n)$, with respect to (A_{i+1}, f_{i+1}), satisfies*

$$W^{i+1}(T_n) = \frac{W^i(T_n)}{2^m}.$$

Proof. Once again we apply induction on m where the base case ($m = 0$) is trivial. Let $m \geq 1$ and $|S| \leq 2^{m+1} - 2$. Again, let x be the pseudo-root of T_n. Since

$$W^j(T_n) = W_\lambda^j = W_x^j \text{ for all } j \geq i,$$

we can focus our attention on the subtree rooted at x. Define $a = \text{left}(x)$ and $b = \text{right}(x)$. We let S_a (resp. S_b) denote the nodes of S which belong

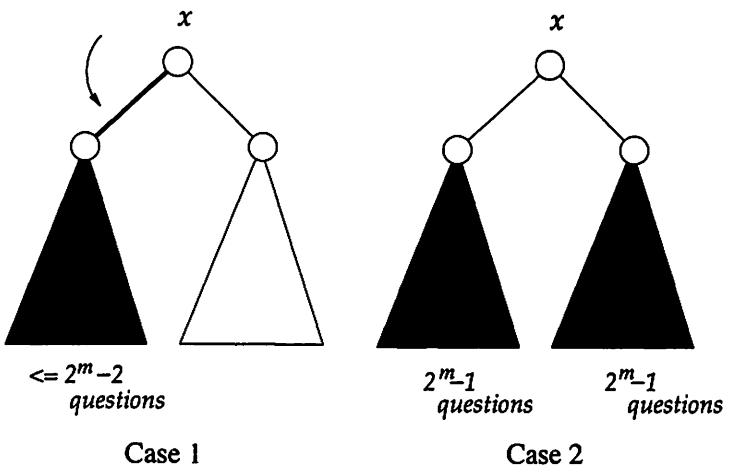

Figure 4: Two cases of Lemma 4.3: Carole uses induction on the shaded trees. In case 1 she answers x regardless of Paul's asking about it and in case 2 she does not.

to T_a (resp. T_b). To prove this lemma we have to consider two cases which are illustrated in Figure 4.

Case 1 $|S_a| \leq 2^m - 2$ or $|S_b| \leq 2^m - 2$. Without loss of generality we may assume that

$$|S_a| \leq 2^m - 2.$$

By induction, Carole can find a set Z_a and answers A_a to the questions in $S_a \cup Z_a$ in such a way that

$$W_a^{i+1} = \frac{W_a^i}{2^{m-1}}$$

and T_a is in a balanced state; that is,

$$W_{\text{left}(v)}^{i+1} = W_{\text{right}(v)}^{i+1} \text{ for all open nodes, } v, \text{ of } T_a. \tag{2}$$

She chooses $f_{i+1}(v) \in \{\text{left}(v), \text{right}(v)\}$ for $v \in S_b$ arbitrarily. Lastly, she defines $f_{i+1}(x) = \text{left}(x) = a$.

In the state (A_{i+1}, f_{i+1}) all the nodes in T_b are unreachable. Furthermore, x is no longer open and the nodes belonging to T_a satisfy Equation (2). Hence (A_{i+1}, f_{i+1}) is a balanced state. Lastly, we observe that as (A_i, f_i) was balanced, we have $W_x^i = 2W_a^i$. As $f_{i+1}(x) = a$,

$$W_x^{i+1} = W_a^{i+1} = \frac{W_a^i}{2^{m-1}} = \frac{W_x^i}{2^m}.$$

Here the second last equality holds by induction. Notice that Carole has chosen an answer for x *even* if $x \notin S$ and again this generosity is *precisely* what helps Carole keep (A_{i+1}, f_{i+1}) balanced.

Case 2 $|S_a| = |S_b| = 2^{m-1} - 1$. Since we have $|S| = 2^m - 2$, $x \notin S$. In this case we apply Lemma 4.2 to both T_a and T_b. Carole finds $Z_a \subseteq T_a$ and $Z_b \subseteq T_b$ and answers to the questions in $S_a \cup Z_a$ and $S_b \cup Z_b$ such that in the next state (A_{i+1}, f_{i+1}) both T_a and T_b are balanced. Furthermore, we have

$$W_a^{i+1} = \frac{W_a^i}{2^m}, W_b^{i+1} = \frac{W_b^{i+1}}{2^m}.$$

Since the state (A_i, f_i) was balanced,

$$W_a^i = W_b^i = \frac{W_x^i}{2}.$$

This implies that $W_a^{i+1} = W_b^{i+1}$. Notice that this state is balanced since all the nodes of T_n that are not in T_x are unreachable. Also,

$$W_x^{i+1} = W_a^{i+1} + W_b^{i+1} = \frac{W_x^i}{2^m}.$$

In this case Carole does not answer x generously. She can be stingy and keep the tree balanced as $W_a^{i+1} = W_b^{i+1}$. □

Now we can obtain the full converse of Lemma 2.1.

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

Proof. We may assume that $k_i = 2^{m_i+1} - 2$; otherwise, we can increase k_i. The weight of T_n is 2^n when the game starts and the tree is balanced. Now if Carole follows the strategy described in Lemma 4.3 in every round then the game remains balanced at the end of each round. Furthermore, the weight of T_n at the end of the game is

$$\frac{2^n}{\prod_{i=1}^{r} 2^{m_i}}.$$

As this quantity is greater than 1, she wins the game. □

5 Another Interesting Game

The game $G(n, k, r)$ is inspired by another game of Shiva Chaudhuri[1]. Briefly, Chaudhuri's game $C(n, k, r)$ is played on a mesh which is diagonally cut into half (See Figure 5). Hence, there are n levels. Each level i consists

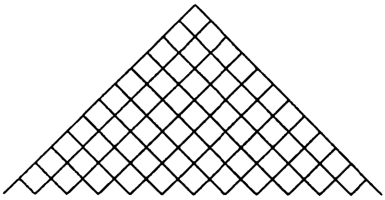
Figure 5: Chaudhuri's game

of i nodes labeled $(i,1),\ldots,(i,i)$. A node (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 neighbors, $X(v)$, at level $i+1$. This defines 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, Paul wins $C(n,2,r)$ if $r > \frac{2n}{3}$. To see this, consider Figure 6. Paul first asks the questions regarding the nodes that are circled in the figure. Now depending on the answer of the root node, he asks two left most questions on level 2 and 3. This way he can move down three levels in two rounds. Whether this is the best possible strategy for $k=2$ is also an intriguing open question.

Acknowledgments

I would like to thank Shiva Chaudhuri for bringing this problem to my attention and Asif Jamshed for proof reading an earlier draft of this pa-

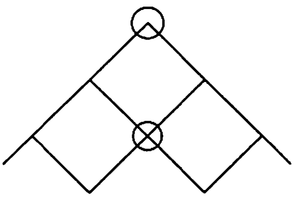
Figure 6: A strategy of $C(k,2,r)$

per. This work is dedicated to my colleague Vicky Chao for sharing her enthusiasm with me.

References

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

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

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

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