LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TM-133

THE SPACE COMPLEXITY OF TWO PEBBLE GAMES ON TREES

Michael C. Loui

May 1979

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

MIT/LCS/TM-133

THE SPACE COMPLEXITY OF TWO PEBBLE GAMES ON TREES

MICHAEL C. LOUI

APRIL 1979

This report was prepared with the support of the National Science Foundation Grant No. MCS77-19754 and the Fannie and John Hertz Foundation.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LABORATORY FOR COMPUTER SCIENCE

CAMBRIDGE MASSACHUSETTS 02139

The Space Complexity of Two Pebble Games on Trees

Michael C. Loui[†]

April 1979

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract. In the standard pebble game the number of pebbles required to pebble the root of a tree can be computed in time linearly proportional to the number of nodes. For the black/white pebble game the number of pebbles necessary to pebble the root of a complete tree is derived.

Key Words: pebble game, tree, computational complexity.

[†]Supported by the Fannie and John Hertz Foundation.

1. Introduction

A combinatorial "pebble" game on graphs has been used to establish trade-offs between time and space required for arithmetic expression evaluation [P] and for Turing machine simulation [HPV]. Given a directed acyclic graph Γ and several pebbles, one places pebbles on the nodes of Γ in steps according to the following rules:

Standard Pebble Game

(1) A step consists of either
  (a) a placement of a pebble on an empty node, or
  (b) a removal of a pebble from a node, or
  (c) a shift of a pebble to an empty node from one of its immediate predecessors.

(2) A pebble may be placed on or shifted to a node only if there are pebbles on all immediate predecessors of the node. (Thus, a node with no predecessors can be pebbled.)

A configuration specifies the nodes of a pebbled graph that hold pebbles. A computation is a finite sequence of configurations $(C_0, ..., C_n)$ such that for each t, either $C_{t-1} = C_t$, or configuration C_{t-1} is transformed into configuration C_t by a step that satisfies restriction (2). The object of the game is to pebble a designated node of Γ, starting from a configuration in which no pebbles are on the graph. Ordinarily, there is a limit on the number of pebbles that can appear in each configuration during a computation.

This pebble game models the evaluation of an expression by a straight-line program. Each pebble represents a storage register. Pebbling a node corresponds to storing a value in a register, removing a pebble from a node to releasing a register, and pebbling the designated node to computing the value of the expression. The number of pebbles used measures the amount of storage (space) used by the program. The number of steps equals the length of the program, equivalently, the time that the program requires.

It is known that $O(n/\log n)$ pebbles suffice to pebble every n node graph with bounded indegree [HPV], and that $O(n/\log n)$ is the best possible general upper bound [PTC]. In general,

determining whether N pebbles suffice to pebble a designated node of a graph is a problem that is complete in polynomial space [GLT], [L].

Generalizing the standard game, Cook and Sethi [CS] introduced a black/white pebble game, which is also played on the nodes of a directed acyclic graph. We shall study a version of the black/white game; without white pebbles, this game is identical to the standard pebble game.

Black/White Pebble Game

(1) A step consists of either
  (a) a placement of a pebble on an empty node, or
  (b) a removal of a pebble from a node, or
  (c) a shift of a black pebble to an empty node from one of its immediate predecessors, or
  (d) a shift of a white pebble to an empty node from one of its immediate successors.

(2) A black pebble may be placed on or shifted to a node only if there are pebbles on all immediate predecessors of the node. (Thus, a black pebble can be placed on a node with no predecessors.)

(3) A white pebble may be removed or shifted from a node only if there are pebbles on all immediate predecessors of the node at the end of the step. (Thus, a white pebble can be removed from a node with no predecessors.)

As in the standard game, the goal of the black/white game is to pebble a designated node of the graph. Let x be a node of graph Γ. A computation on Γ ensures x if it starts from an empty configuration (no pebbles on the graph) and ends at a configuration with just one pebble, a black pebble on x. A computation promises x if it starts from an empty configuration, reaches a configuration in which x holds a black pebble, and ends at an empty configuration. A computation uses N pebbles if N is the maximum total number of pebbles – number of blacks plus number of whites – on the graph during the computation.

This black/white game models the proof of a theorem. Each node represents an assertion that can be deduced via some inference rule when its predecessors have been proved. Placing a black pebble on a node corresponds to proving that assertion. Placing a white pebble on a node corresponds to assuming that assertion; the assumption is later justified by proving the predecessors.

As for the standard game, there exists an infinite family of graphs G_n with n nodes, each of

which has indegree 2, such that to promise some node of G_n requires $\Omega(n/\log n)$ pebbles in the black/white game [GT]. If a node of a graph can be ensured by a black/white computation that uses N pebbles, then it can be pebbled with $N(N + 1)/2$ pebbles in the standard game [M].

In order to compare the standard game and the black/white game further, we study the space requirements of the two games on trees. In Section 2 we show that in the standard game, the exact number of pebbles required to pebble the root of a tree can be computed in time linearly proportional to the number of nodes; this result was previously obtained by Redziejowski [R] in a different form. Section 3 defines concepts and notations for the black/white game. In Section 4 we extend the ideas of Section 2 to the black/white game. Whereas $(m - 1)h + 1$ pebbles are necessary and sufficient to pebble the root of a complete m-ary tree of height h in the standard game, the black/white game requires

$\lceil (m - 1)h/2 + (m + 1)/2 \rceil$ pebbles to ensure the root,

$\lfloor (m - 1)h/2 + (m + 1)/2 \rfloor$ pebbles to promise the root.

The results of Section 4 also apply to a parallel version of the black/white game that we treat in Section 5. Section 6 presents a few remarks about the time complexity of pebble games on trees. Section 7 indicates topics for further study.

2. Standard Game on Trees

In this section we refer to the tree of Figure 1. The immediate predecessors of each node are its direct descendants, the immediate successor is its parent node.

For a computation $(C_0, ..., C_n)$, the configuration at time t is C_t.

Proposition 1. Let the root x of tree T have direct descendants $x_1, ..., x_m$. Let $T_1, ..., T_m$ be the subtrees of T with roots $x_1, ..., x_m$. In the standard pebble game let N_i be the number of pebbles necessary and sufficient to advance a pebble to x_i starting from an empty configuration on T_i. Suppose $N_1 \leq ... \leq N_m$. Exactly

$$\max \{N_m, N_{m-1} + 1, ..., N_1 + m - 1\}$$

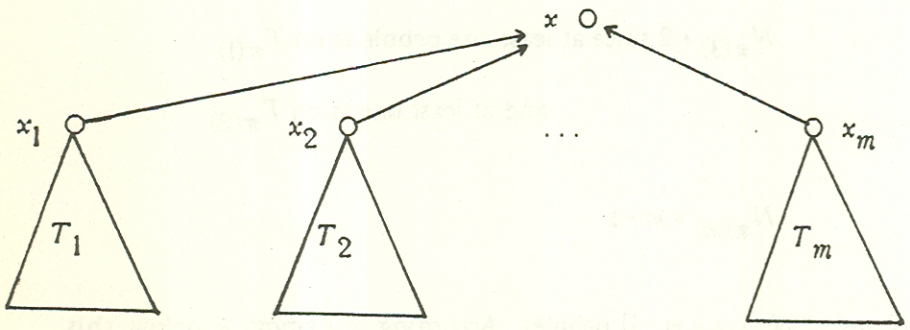
pebbles are required to pebble x, starting from an empty configuration on T.

Proof. Sufficiency. First, use N_m pebbles on T_m to advance a pebble to x_m, and clear the rest of T_m. Next, placing pebbles on T_{m-1}, use N_{m-1} pebbles to pebble x_{m-1} and leave pebbles on x_{m-1} and x_m. Continue in this order: $x_{m-2}, ..., x_1$. Finally, shift the pebble from x_1 to x and remove the rest of the pebbles.

Necessity. Let S be a computation that pebbles x. At some time t there are pebbles on all direct descendants of x. For each i, S uses at least N_i pebbles on subtree T_i because S must pebble every x_i in order to pebble x. Let t_i be the *last* time before t at which N_i pebbles appear on subtree T_i. From t_i until t at least one pebble always appears on T_i. Let π be a permutation for which $t_{\pi(1)} \leq t_{\pi(2)} \leq ... \leq t_{\pi(m)}$. Then

at time	the number of pebbles on T is at least
$t_{\pi(1)}$	$N_{\pi(1)}$
$t_{\pi(2)}$	$N_{\pi(2)} + 1$ since at least one pebble is on $T_{\pi(1)}$

Figure 1.



at time	the number of pebbles on T is at least
$t_{\pi(3)}$	$N_{\pi(3)} + 2$ since at least one pebble is on $T_{\pi(1)}$ and at least one is on $T_{\pi(2)}$
⋮	⋮
$t_{\pi(m)}$	$N_{\pi(m)} + m - 1$

Thus, S uses at least max $\{N_{\pi(1)}, ..., N_{\pi(m)} + m - 1\}$ pebbles. According to Lemma A below, this number is at least max $\{N_m, ..., N_1 + m - 1\}$. □

Lemma A. If f and g are nondecreasing functions on $\{1, ..., m\}$, then a permutation π that minimizes

$$\max \{f(1) + g(\pi(1)), ..., f(m) + g(\pi(m))\}$$

is $\pi(1) = m$, $\pi(2) = m - 1$, ..., $\pi(m) = 1$.

Proof. Suppose $\pi(k) = 1$. Evidently,

$$\max \{f(k) + g(\pi(m)), f(m) + g(\pi(k))\} \leq f(m) + g(\pi(m))$$
$$\leq \max \{f(k) + g(\pi(k)), f(m) + g(\pi(m))\}.$$

Thus, exchanging the values of $\pi(k)$ and $\pi(m)$, we may assume that $\pi(m) = 1$. Continue the argument in this way to show that we can take $\pi(m - 1) = 2$, ..., $\pi(1) = m$. □

Declare a single node to be a tree of height 0. Two facts follow directly from Proposition 1.

Corollary 1. In the standard game, to pebble the root of a complete m-ary tree of height h requires exactly $(m - 1)\, h + 1$ pebbles.

Corollary 2. For every tree with n nodes which each have at most m direct descendants, the exact number of pebbles required to pebble the root can be computed in time $O(nm \log m)$.

Proof. At each node x at most m numbers $N_1, ..., N_m$ need to be sorted. □

3. Definitions for the Black/White Game

In the black/white pebble game on a graph Γ, we represent a configuration by a pair $\langle W,B\rangle$, where W and B are disjoint sets of nodes of Γ; in this configuration, W is the set of nodes that hold white pebbles, B is the set of nodes that hold black. Write $\langle W,B\rangle \rightarrow \langle W',B'\rangle$ if configuration $\langle W,B\rangle$ can be transformed into configuration $\langle W',B'\rangle$ by a single legal step (i.e., a step that satisfies restrictions (2) and (3) in the definition in Section 1). Let $\rightarrow^*$ denote the reflexive transitive closure of the relation $\rightarrow$.

Let $S = (\langle W_0,B_0\rangle, ..., \langle W_n,B_n\rangle)$ be a computation on Γ. We use the notation

$$S:\ \langle W_0,B_0\rangle \rightarrow^* \langle W_n,B_n\rangle \text{ on } \Gamma$$

to state that computation S starts from configuration $\langle W_0,B_0\rangle$ and ends at configuration $\langle W_n,B_n\rangle$ on graph Γ. At <u>time</u> t, computation S is in configuration $\langle W_t,B_t\rangle$; the notation $[t_1,t_2]$ denotes the <u>interval</u> of times $t_1, t_1+1, ..., t_2$. Computation S <u>pebbles node x at time t</u> if the tth step, which transforms $\langle W_{t-1},B_{t-1}\rangle$ into $\langle W_t,B_t\rangle$, is a placement or a shift of a pebble onto x. There are N pebbles on Γ at time t if $|W_t| + |B_t| = N$; the computation S <u>uses N pebbles</u> if in every configuration there are at most N pebbles on Γ, and in some configuration there are N.

Let x be a node of Γ. Computation S <u>ensures</u> x if

$$S:\ \langle \varnothing,\varnothing\rangle \rightarrow^* \langle \varnothing,\{x\}\rangle \text{ on } \Gamma.$$

Computation S <u>promises</u> x if

$$S:\ \langle \varnothing,\varnothing\rangle \rightarrow^* \langle W_t,B_t\rangle \rightarrow^* \langle \varnothing,\varnothing\rangle \text{ on } \Gamma,$$

where $x \in B_t$ at some intermediate time t.

Let Γ be a directed acyclic graph and Γ' be a subgraph of Γ with node set V'. The <u>restriction</u> of computation $S = (\langle W_0,B_0\rangle, ..., \langle W_n,B_n\rangle)$ to Γ' during $[t_1,t_2]$ is the sequence of configurations

$$(\langle W_{t_1} \cap V', B_{t_1} \cap V'\rangle, ..., \langle W_{t_2} \cap V', B_{t_2} \cap V'\rangle).$$

One can confirm that the restriction of S to Γ' is itself a computation.

A routine inductive argument establishes a duality principle [CS]: on every graph Γ, if

$$S\text{: } \langle W,B\rangle \rightarrow^{*} \langle W',B'\rangle \text{ on } \Gamma,$$

using N pebbles, then there is a computation S^{rev} such that

$$S^{rev}\text{: } \langle B',W'\rangle \rightarrow^{*} \langle B,W\rangle \text{ on } \Gamma,$$

and S^{rev} uses N pebbles. Call S^{rev} the reversal of S. In informal terms, S^{rev} interchanges whites and blacks and runs S backward in time. Extending our terminology, say S^{rev} promises x if S promises x: at some time S^{rev} pebbles x with a white pebble.

4. Black/White Game on Complete Trees

We shall determine the number of black and white pebbles necessary to ensure and to promise the root of a complete tree. The addition of white pebbles yields a savings in space because the white pebbles can be used to defer the pebbling of about half the nodes in the tree until after the root is pebbled.

Throughout this section let T be a complete m-ary tree of height h. Let x be the root of T with direct descendants $x_1, \ldots, x_m$. For $i = 1, \ldots, m$, let T_i be the subtree with root x_i. For each i, let $x_{i1}, \ldots, x_{im}$ be the direct descendants of x_i and T_{ij} be the subtree with root x_{ij}. See Figure 2. In the black/white game, let $E(m,h)$ be the number of pebbles needed to ensure x and $P(m,h)$ be the number to promise x. By duality, $E(m,h)$ pebbles are necessary and sufficient to achieve

$$\langle\{x\},\varnothing\rangle \rightarrow^{*} \langle\varnothing,\varnothing\rangle \text{ on } T.$$

Proposition 2. For every m and $h \geq 1$,

$$E(m,h) = \lceil (m-1)h/2 + (m+1)/2 \rceil;$$

$$P(m,h) = \lfloor (m-1)h/2 + (m+1)/2 \rfloor.$$

We devote the rest of this section to proving this proposition. For every m,

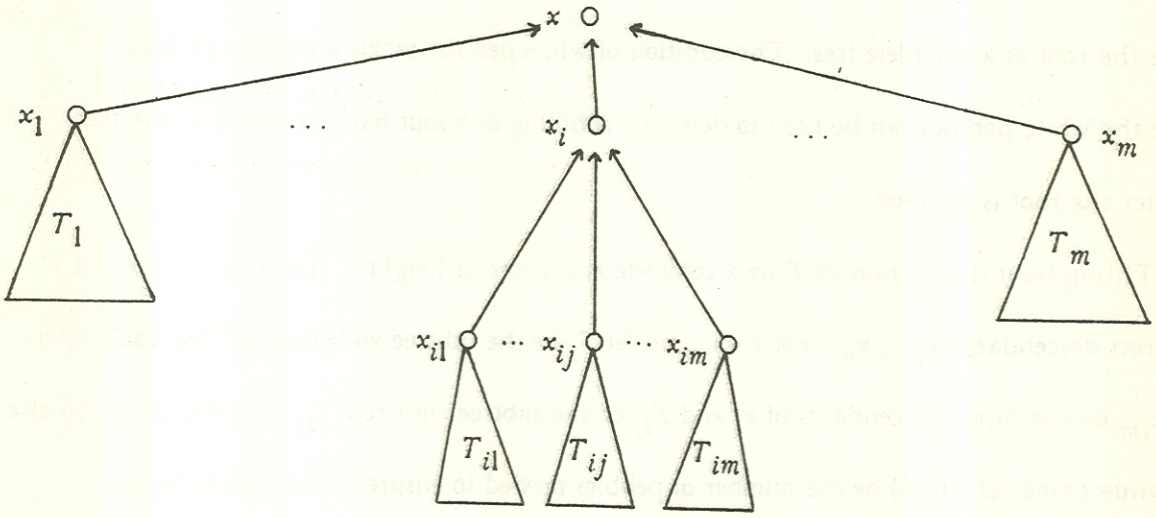
$$E(m,1) = P(m,1) = m. \tag{1}$$

Lemma B. For every m and h,

$$E(m,h) \leq P(m,h) + 1. \tag{2}$$

Proof. From a computation S_p that promises x we can obtain a computation S_e that ensures x and uses at most one additional pebble. Computation S_e simply performs S_p, except that the pebble on x is not removed once S_p pebbles x. □

Figure 2. A complete m-ary tree.



Lemma C. For every m,

$$E(m,h) \leq E(m,h-2) + m - 1 \text{ for } h \geq 3; \tag{3}$$

$$E(m,2) \leq m + \lfloor m/2 \rfloor. \tag{4}$$

Proof. The computation $S(h)$ for ensuring x at height h will employ computations $S(h-2)$ and $S(h-2)^{rev}$ on subtrees T_{ij}. (We think of computation $S(h-2)$ in a generic way because all subtrees T_{ij} are isomorphic.)

Computation $S(0)$: A tree of height 0 is a single node. Merely place a black pebble on that node.

Computation $S(1)$: Place black pebbles on $x_1, \ldots, x_m$. Shift the pebble from x_1 to x and remove the rest of the pebbles from the tree.

Computation $S(h)$ for $h \geq 2$: Let $r = \lceil m/2 \rceil + 1$ and $s = \lfloor m/2 \rfloor$. This computation is divided into five parts. In essence, $S(h)$ places black pebbles on half of the direct descendants of x and white pebbles on the other direct descendants. After placing a black pebble on x, computation $S(h)$ clears the white pebbles from the tree.

Part 1: $\langle \varnothing, \varnothing \rangle \rightarrow^{*} \langle \varnothing, \{x_1, \ldots, x_{r-1}\} \rangle$

for $i = 1, \ldots, r-1$

begin

for $j = 1, \ldots, r-1$ use $S(h-2)$ on T_{ij} to ensure x_{ij};

place white pebbles on $x_{ir}, \ldots, x_{im}$;

shift the black pebble from x_{i1} to x_i;

remove the black pebbles from $x_{i2}, \ldots, x_{i,r-1}$;

for $j = r, \ldots, m$ use $S(h-2)^{rev}$ on T_{ij} to clear the white pebble from x_{ij};

end

Part 2: $\langle \varnothing, \{x_1, \ldots, x_{r-1}\} \rangle \rightarrow^{*} \langle \varnothing, \{x_1, \ldots, x_{r-1}, x_{r1}, \ldots, x_{rs}\} \rangle$

for $j = 1, \ldots, s$ use $S(h - 2)$ on T_{rj} to ensure x_{rj};

Part 3: $\langle \varnothing, \{x_1, \ldots, x_{r-1}, x_{r1}, \ldots, x_{rs}\} \rangle \rightarrow^{*} \langle \{x_r, \ldots, x_m\}, \{x, x_{r1}, \ldots, x_{rs}\} \rangle$

place white pebbles on $x_r, \ldots, x_m$;

shift the black pebble from x_1 to x;

remove the black pebbles from $x_2, \ldots, x_{r-1}$;

Part 4: $\langle \{x_r, \ldots, x_m\}, \{x, x_{r1}, \ldots, x_{rs}\} \rangle \rightarrow^{*} \langle \{x_{r+1}, \ldots, x_m\}, \{x\} \rangle$

place white pebbles on $x_{r,s+1}, \ldots, x_{r,m-1}$;

shift the white pebble from x_r to x_{r_m};

remove the black pebbles from $x_{r1}, \ldots, x_{rs}$;

for $j = s + 1, \ldots, m$ use $S(h - 2)^{rev}$ on T_{rj} to clear the white pebble from x_{rj};

Part 5: $\langle \{x_{r+1}, \ldots, x_m\}, \{x\} \rangle \rightarrow^{*} \langle \varnothing, \{x\} \rangle$

for $i = r + 1, \ldots, m$

begin

for $j = 1, \ldots, s$ use $S(h - 2)$ on T_{ij} to ensure x_{ij};

place white pebbles on $x_{i,s+1}, \ldots, x_{i,m-1}$

shift the white pebble from x_i to x_{im};

remove the black pebbles from $x_{i1}, \ldots, x_{is}$;

for $j = s + 1, \ldots, m$ use $S(h - 2)^{rev}$ on T_{rj} to clear the white pebble from x_{rj};

end

One can verify that whenever computation $S(h)$ invokes $S(h - 2)$ or $S(h - 2)^{rev}$ on subtree T_{ij} there are at most $m - 1$ pebbles elsewhere on T. Furthermore, if $h = 2$, then there are always at most $m + \lfloor m/2 \rfloor$ pebbles on the tree. □

For example, let $m = 4$; then $r = 3$ and $s = 2$. The computation $S(h)$ uses $S(h - 2)$ to pebble x_{11} and x_{12} with black pebbles, puts whites on x_{13} and x_{14}, shifts the pebble on x_{11} to x_1, and clears the rest of the pebbles from T_1. Similarly, it pebbles x_2 with a black pebble. After $S(h - 2)$ is used to pebble x_{31} and x_{32} with black pebbles, whites are placed on x_3 and x_4, a black pebble shifted to x from x_1, and the black pebble on x_2 removed. A white pebble is placed on x_{33}; the white pebble on x_3 is shifted to x_{34}. The black pebbles on x_{31} and x_{32} are removed, and $S(h - 2)^{rev}$ is invoked twice to clear pebbles from T_3. Next, $S(h)$ uses $S(h - 2)$ twice to pebble x_{41} and x_{42}, places a white pebble on x_{43}, shifts the white pebble on x_4 to x_{44}, and removes the black pebbles on x_{41} and x_{42}. Finally, $S(h - 2)^{rev}$ is employed twice to clear pebbles from T_4.

Lemma D. For $h \geq 1$,

$$E(m,h) \geq P(m,h - 1) + \lfloor m/2 \rfloor \text{ for every } m, \tag{5}$$

$$P(m,h) \geq P(m,h - 1) + \lfloor m/2 \rfloor \text{ for all odd } m. \tag{6}$$

Proof. Let S_e be a computation that ensures x. Let t be a time just before S_e places a black pebble on x; at time t there are pebbles on $x_1, \ldots, x_m$. Let r_i be the last time before t at which no pebbles are on subtree T_i; let s_i be the first time after t at which no pebbles are on T_i. Between r_i and s_i there is always at least one pebble on T_i.

Computation S_e uses $P(m,h - 1)$ pebbles on each T_i: the restriction of S_e to T_i during $[r_i,s_i]$ computes

$$\langle \varnothing,\varnothing \rangle \rightarrow^* \langle W_t,B_t \rangle \rightarrow^* \langle \varnothing,\varnothing \rangle \text{ on } T_i,$$

$$\text{where } x_i \in W_t \text{ or } x_i \in B_t.$$

Without loss of generality, let $x_1, \ldots, x_k$ be the nodes x_i such that at some time between r_i and t computation S_e uses $P(m,h - 1)$ pebbles on T_i; by definition of r_i, from time $r_i + 1$ until time t there is always at least one pebble on this subtree. Then $x_{k+1}, \ldots, x_m$ are the nodes x_j such that at some time between t and s_j computation S_e uses $P(m,h - 1)$ pebbles on T_j; from time t until time $s_j - 1$

there is always at least one pebble on this subtree. As in the proof of Proposition 1 we can establish that S_e uses at least

$$\max \{P(m,h-1), P(m,h-1)+1, \ldots, P(m,h-1)+k-1\}$$

pebbles before (or at) time t. After time t, there is a pebble on x; an analogous argument shows that S_e uses at least

$$\max \{P(m,h-1)+1, \ldots, P(m,h-1)+m-k\}$$

pebbles after time t. Therefore, S_e uses at least

$$\max \{P(m,h-1)+k-1, P(m,h-1)+m-k\}$$

pebbles. This expression, which gives a lower bound on $E(m,h)$, is minimized for $k = \lceil m/2 \rceil$. Thus,

$$E(m,h) \geq P(m,h-1) + \lfloor m/2 \rfloor.$$

Let S_p be a computation that uses $P(m,h)$ pebbles to promise x. As before, let all direct descendants of x hold pebbles at time t. The preceding line of reasoning demonstrates that for some k, computation S_p uses at least $P(m,h-1)+k-1$ pebbles before time t and at least $P(m,h-1)+m-k-1$ pebbles after time t. The larger of these two quantities is minimized for $k = \lceil m/2 \rceil$. Ergo, when m is odd,

$$\begin{aligned} P(m,h) &\geq \max \{P(m,h-1)+k-1, P(m,h-1)+m-k-1\} \\ &\geq P(m,h-1) + \lfloor m/2 \rfloor. \quad \square \end{aligned}$$

Lemma E. Let m be even. For $h \geq 2$,

$$P(m,h) \geq P(m,h-2) + m - 1. \tag{7}$$

Proof. Let S be a computation that promises x. We shall prove that in some configuration of S there are at least $P(h-2)+m-1$ pebbles on the tree T.

Let t be a time at which $x_1, \ldots, x_m$ all hold pebbles. For each i, if x_i holds a black at time t, let t_i be the last time before t at which x_i is pebbled; during $[t_i,t]$ there is a black pebble on x_i. If x_i holds a white at time t, then let t_i be the first time after t such that the pebble on x_i is removed at

time $t_i + 1$; during $[t,t_i]$ there is a white pebble on x_i. We may assume that $t_1 \leq \ldots \leq t_m$.

If x_i holds a black at time t, then there are pebbles on its direct descendants $x_{i1}, \ldots, x_{im}$ at time $t_i - 1$; if x_i holds a white at time t, then there are pebbles on $x_{i1}, \ldots, x_{im}$ at time $t_i + 1$. For each j let r_{ij} be the last time before $t_i - 1$ (resp. $t_i + 1$) at which T_{ij} is empty, s_{ij} the first time after $t_i - 1$ (resp. $t_i + 1$) at which T_{ij} is empty. If x_i holds a black at time t, then $r_{ij} < t_i - 1 < s_{ij}$; if x_i holds a white at time t, then $r_{ij} < t_i + 1 < s_{ij}$.

The restriction of S to T_{ij} during $[r_{ij},s_{ij}]$ promises x_{ij}. For each i,j, let p_{ij} be a time between r_{ij} and s_{ij} at which $P(m,h - 2)$ pebbles appear on T_{ij}. We may assume that $p_{i1} \leq \ldots \leq p_{im}$.

Observe that if $k > 1$ and $p_{ik} < t_i$, then $r_{ij} < p_{ij} \leq p_{ik} < s_{ij}$ for $j < k$, hence there are pebbles on on $T_{i1}, \ldots, T_{i,k-1}$ at time p_{ik}; thus, at time p_{ik} there are at least $P(m,h - 2) + k - 1$ pebbles on the tree T. Analogously, if $k < m$ and $p_{ik} > t_i$, then there are $P(m,h - 2) + m - k$ pebbles on the tree at time p_{ik}.

First, we claim that we can suppose that for each i, $p_{i1} \leq t_i \leq p_{im}$. If $p_{im} < t_i$, then according to the last paragraph, at time p_{im} there are $P(m,h - 2) + m - 1$ pebbles on the tree, which was to be proved. Similarly, if $p_{i1} > t_i$, then at time p_{i1} there are $P(m,h - 2) + m - 1$ pebbles on the tree.

By hypothesis, m is even. Fix $k = m/2$. Reversing S, if necessary, we may assume that $t_k \leq t$; at time t_k there is a black pebble on x_k. Consider the possible relationships of $p_{k,k+1}$ to $t_1, \ldots, t_m$.

Case 1: $t_{k-1} \leq p_{k,k+1} < t_k$. At time $p_{k,k+1}$ there are pebbles on $x_1, \ldots, x_{k-1}$ (since $t_k \leq t$) and at least one each on $T_{k1}, \ldots, T_{kk}$ (since $r_{ij} < p_{ij} \leq p_{k,k+1} < t_i \leq s_{ij}$ for $j \leq k$). So at this time there are $P(m,h - 2) + (k - 1) + k = P(m,h - 2) + m - 1$ pebbles on the tree T.

Case 2: $t_k \leq p_{k,k+1} \leq t$. At time $p_{k,k+1}$ there are pebbles on $x_1, \ldots, x_k$ and at least one each on $T_{k,k+2}, \ldots, T_{km}$ (since $r_{ij} < t_i \leq p_{k,k+1} \leq p_{ij} < s_{ij}$ for $j > k + 1$), a total of $P(m,h - 2) + k + (m - k - 1) = P(m,h - 2) + m - 1$ on the tree.

Case 3: $t \le p_{k,k+1} \le t_{k+1}$. At time $p_{k,k+1}$ there are pebbles on $x_{k+1}, \ldots, x_m$ and at least one each on $T_{k,k+2}, \ldots, T_{km}$, a total of $P(m,h-2) + (m-k) + (m-k-1) = P(m,h-2) + m - 1$ on the tree.

Case 4: $p_{k,k+1} < t_{k-1}$. Recall that $t_{k-2} \le t_{k-1}$ and $p_{k-1,1} \le t_{k-1}$. For every $i \le k$, since $r_{i1} < p_{i1} \le t_{i1} - 1 < s_{i1}$, there is a pebble on T_{i1} during $[p_{i1}, t_i - 1]$. In general, for $2 \le i \le k-1$, if p_{i1}, $p_{i+1,1}$, ..., $p_{k-1,1}$, and $p_{k,k+1}$ are all strictly less than t_i, but $\max\{p_{i1}, p_{i+1,1}, \ldots, p_{k-1,1}, p_{k,k+1}\} \ge t_{i-1}$, then at this maximum time there are $P(m,h-2)$ pebbles on one of the subtrees $\{T_{i1}, T_{i+1,1}, \ldots, T_{k-1,1}, T_{k,k+1}\}$ plus additional pebbles on the other $k-i$ of the $k-i+1$ subtrees $\{T_{i1}, T_{i+1,1}, \ldots, T_{k-1,1}, T_{k,k+1}\}$, on $x_1, \ldots, x_{i-1}$, and on $T_{k1}, \ldots, T_{kk}$, – a total of at least

$$P(m,h-2) + (k-i) + (i-1) + k = P(m,h-2) + m - 1$$

pebbles. If $t_1 = \max\{t_1, p_{21}, p_{31}, \ldots, p_{k-1,1}, p_{k,k+1}\}$, then since $p_{11} \le t_1$ too, at time $\max\{p_{11}, p_{21}, \ldots, p_{k-1,1}, p_{k,k+1}\}$ there are $P(m,h-2) + m - 1$ pebbles on the tree.

Case 5: $t_{k+1} < p_{k,k+1}$. Similar to Case 4. □

Proof of Proposition 2. By (1), (3), and (4),

$$E(m,h) \le \lceil (m-1)h/2 + (m+1)/2 \rceil \text{ for } h \ge 1. \tag{8}$$

Use (5), (3), and (8) to obtain for $h \ge 2$,

$$\begin{aligned} P(m,h) &\le E(m,h+1) - \lfloor m/2 \rfloor \\ &\le E(m,h-1) + \lceil m/2 \rceil - 1 \\ &\le \lceil (m-1)(h-1)/2 + (m+1)/2 \rceil + \lceil m/2 \rceil - 1 \\ &= \lceil (m-1)h/2 \rceil + \lceil m/2 \rceil. \end{aligned}$$

Rewrite the last expression:

$$P(m,h) \le \lfloor (m-1)h/2 + (m+1)/2 \rfloor \text{ for } h \ge 2. \tag{9}$$

By (5) and (2),

$$P(m,2) \ge E(m,2) - 1 \ge P(m,1) + \lfloor m/2 \rfloor - 1,$$

hence by (1),

$$P(m,2) \geq m - 1 + m/2 \text{ for even } m. \tag{10}$$

The combination of (1), (6), (7), and (10) yields

$$P(m,h) \geq \lceil (m - 1)h/2 \rceil + \lceil m/2 \rceil = \lfloor (m - 1)h/2 + (m + 1)/2 \rfloor \text{ for all } m \text{ and } h \geq 1. \tag{11}$$

Finally, employ (5) and (11) to obtain for $h \geq 2$

$$E(m,h) \geq \lceil (m - 1)(h - 1)/2 \rceil + \lfloor m/2 \rfloor + \lceil m/2 \rceil = \lceil (m - 1)h/2 + (m + 1)/2 \rceil. \tag{12}$$

The asserted values for $E(m,h)$ and $P(m,h)$ follow immediately from (8), (9), (11), and (12). □

5. Parallel Black/White Pebble Game

A variation of the rules of the black/white pebble game permits a change of several pebbles on the graph at each step. This variation is the black/white game originally defined by [CS]. Let Γ be a directed acyclic graph.

Parallel Black/White Pebble Game

(1) A configuration is a pair $\langle W,B\rangle$, where W and B are disjoint sets of nodes of Γ.

(2) A configuration $\langle W,B\rangle$ derives another configuration $\langle W',B'\rangle$, written $\langle W,B\rangle \Rightarrow \langle W',B'\rangle$, if

(a) for each x in W, either $x \in W'$, or x has no immediate predecessors, or all immediate predecessors of x are in $W' \cup B'$;

(b) for each x in B', either $x \in B$, or x has no immediate predecessors, or all immediate predecessors of x are in $W \cup B$.

A computation is a sequence of configurations $(\langle W_0,B_0\rangle, \ldots, \langle W_n,B_n\rangle)$ such that for each t, $\langle W_t,B_t\rangle \Rightarrow \langle W_{t+1},B_{t+1}\rangle$. One can verify that the proofs of Section 4 apply to this game as well; for example, if $\langle W,B\rangle \rightarrow \langle W',B'\rangle$, then $\langle W,B\rangle \Rightarrow \langle W',B'\rangle$.

Proposition 3. In the parallel black/white pebble game on a complete m-ary tree of height h, the number of pebbles necessary and sufficient to ensure the root and the number to promise the root are the same as in the black/white game, namely,

$\lceil (m-1)h/2 + (m+1)/2 \rceil$ to ensure the root,

$\lfloor (m-1)h/2 + (m+1)/2 \rfloor$ to promise the root.

6. Time Complexity

We show that for the black/white pebble game on a tree, every computation that uses N pebbles and ensures or promises the root can be modified to obtain a computation that uses at most N pebbles and pebbles each node only once. The same reasoning also applies to the standard game: any computation that pebbles the root of a tree needs to pebble each node only once. In particular, these results apply to computations that use the minimum required number of pebbles.

Proposition 4. For every N, the shortest computation that uses at most N pebbles to ensure (resp. to promise) the root of a tree pebbles each node only once.

Proof. Let S be the shortest computation that uses at most N pebbles to ensure the root; the proof for a promising computation is identical. Suppose S pebbles some node twice; this hypothesis will compel a contradiction.

Let z be the node closest to the root that S pebbles twice. Let z hold a pebble during $[t_1,t_2]$ and $[t_3,t_4]$, where $t_2 < t_3$. We may assume that z is not the root of the tree; otherwise, we could omit from S all the placements on z except the last and obtain a shorter computation that ensures z.

Let y be the parent of z. Every pebbling of z must be useful: during $[t_1,t_2]$ a black pebble is placed on y or a white pebble is removed from y; otherwise, the placement at time t_1 could be omitted. Similarly, during $[t_3,t_4]$, a black pebble is placed on y or a white pebble is removed from y. We have inferred that S pebbles y twice, but z is the highest node in the tree that S pebbles twice. Contradiction. □

This argument can be extended. In directed acyclic graph Γ let x be the only node with no successors. Suppose y is a node of Γ and computation uses N pebbles to ensure (or to promise) x. Then there is a computation S that uses N pebbles to ensure (promise) x such that the number of times that S pebbles y is bounded above by the number of distinct paths from y to x.

7. Open Problems

Although Corollary 1 and Proposition 2 together suggest that the addition of white pebbles to the standard (black pebble) game reduces the number of pebbles necessary to pebble a graph by only a constant factor, the precise relationship between the two pebble games remains unresolved. Notwithstanding the result of [M], no graph has been presented for which an ensuring black/white computation uses fewer than half the number of pebbles needed to pebble a node in the standard game.

The analysis of the black/white game on complete trees might be extended to arbitrary trees.

Acknowledgments. Michael Paterson suggested the terms *promise* and *ensure*. Jeffrey Jaffe helped strengthen an earlier lower bound for the black/white game. Thomas Lengauer has obtained Proposition 2 independently.

References

[CS] S.A. Cook and R. Sethi, "Storage requirements for deterministic polynomial time recognizable languages." *J. Comp. Sys. Sci.* 13 (1976) 25-37.

[GT] J.R. Gilbert and R.E. Tarjan, "Variations of a pebble game on graphs." Res. Rep. CS78-661, Dept. Comp. Sci., Stanford Univ., Sep. 1978.

[GLT] J.R. Gilbert, T. Lengauer, and R.E. Tarjan, "The Pebbling Problem is Complete in Polynomial Space." *Proc. 11th Ann. ACM Symp. on Theory of Computing*, 1979.

[HPV] J. Hopcroft, W. Paul, and L. Valiant, "On time versus space." *J. ACM* 24 (1977) 332-337.

[L] M.C. Loui, "Minimum Register Allocation is Complete in Polynomial Space." Tech. Memo. TM-128, Lab. for Comp. Sci., Mass. Inst. Tech., Feb. 1979.

[M] F. Meyer auf der Heide, "A comparison between two variations of a pebble game on graphs." Fakultät für Mathematik, Univ. Bielefeld, West Germany, Nov. 1978.

[PTC] W.J. Paul, R.E. Tarjan, and J.R. Celoni, "Space bounds for a game on graphs." *Math. Sys. Th.* 10 (1977) 239-251.

[P] N. Pippenger, "A time-space trade-off." *J. ACM* 25 (1978) 509-515.

[R] R.R. Redziejowski, "On arithmetic expressions and trees." *Comm. ACM* 12 (1969) 81-84.