Pebbles and Branching Programs

Stephen Cook

On the occasion of
Les Valiant's 60th Birthday Celebration
May 30, 2009

Joint work with

Mark Braverman
Pierre McKenzie
Rahul Santhanam
Dustin Wehr

"Perhaps the principal embarrassment of complexity theory at the present time is its failure to provide techniques for proving non-trivial lower bounds on the complexity of some of the commonest combinatorial and arithmetic problems."

"Perhaps the principal embarrassment of complexity theory at the present time is its failure to provide techniques for proving non-trivial lower bounds on the complexity of some of the commonest combinatorial and arithmetic problems."

Les Valiant, STOC 1975
On Non-Linear Lower Bounds in Computational Complexity

(Constructs linear size superconcentrators)

Complexity Classes

$$\mathbf{AC}^0(6) \subseteq \mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{LogCFL}$$
$$\subseteq \mathbf{AC}^1 \subseteq \mathbf{NC}^2 \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PH}$$

As far as is known, $\mathbf{AC}^0(6)$ cannot determine whether a majority of its input bits are ones.

Yet it is open whether $\mathbf{AC}^0(6) = \mathbf{PH}$.

Here we introduce the

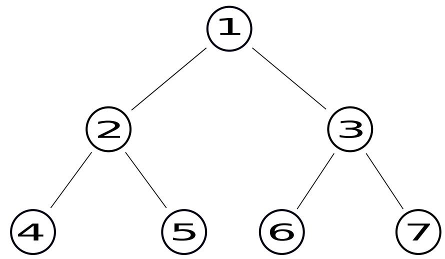
Tree Evaluation Problem (TEP)

We show TEP is in $\mathbf{LogDCFL}$.

We are trying to prove TEP $\notin \mathbf{L}$
(and TEP $\notin \mathbf{NL}$)

Tree Evaluation Problem

(Generalizes a problem in [Taitslin05])



Tree of height $h = 3$ with heap numbering

T_d^h: Balanced d-ary tree of height h
DEFAULT: $d = 2$
$[k] = \{1, ..., k\}$

TEP(h, k) Applies to T_2^h. Assume $h, k \geq 2$
Input:
$v_i \in [k]$ for each leaf i
Function $f_i : [k] \times [k] \to [k]$ for each internal node i
(Thus every node i gets a value $v_i \in [k]$)

Output: root value $v_1 \in [k]$

Decision Problem: Does $v_1 = 1$?

Claim: TEP$(h, k) \in \mathbf{LogDCFL}$

Space-efficient algorithms for TEP come from pebbling

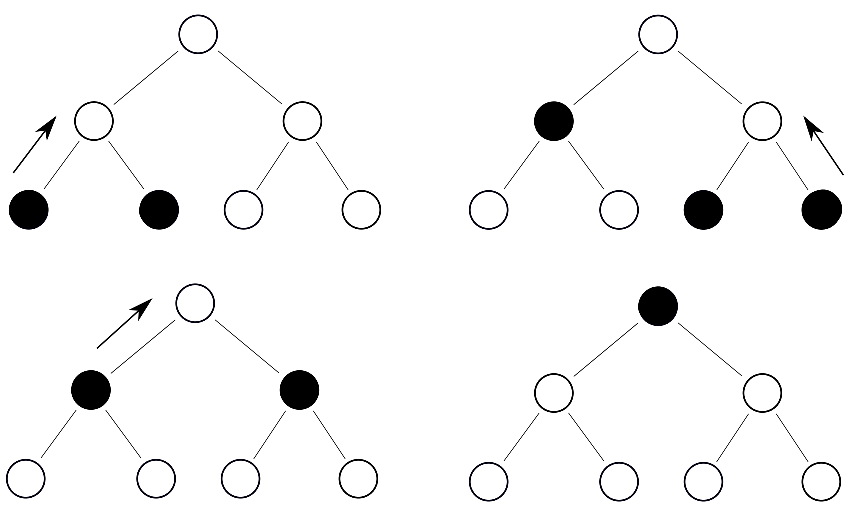
Deterministic algorithms come from 'black' pebbling. [Paterson/Hewitt70]

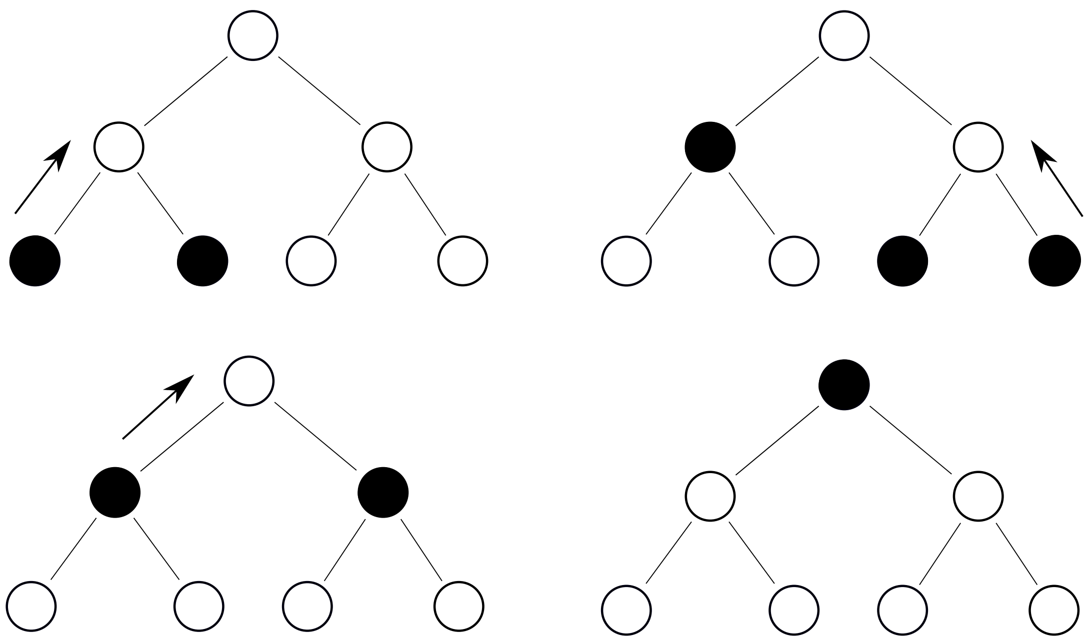
Rules:

- Place a pebble on any leaf.
- If both children of node i are pebbled, slide one of them to the parent.
- Remove any pebble at any time.

Goal: Pebble the root using a minimum number of pebbles.

Easy Theorem: T_2^h requires exactly h pebbles.





Recall: T_2^h requires exactly h pebbles.

Corollary: TEP$(h,k) \in$ DSPACE$(h \log k)$

This is NOT a log space algorithm.

Input size $n = (2^h - 1)k^2 \log k$
$\log n = \Theta(h + \log k)$

k-way Branching Programs

A k-way BP B solving TEP(h, k) is a directed multigraph with nodes called *states*. Each non-final state q is labeled either with a leaf node i, with k outedges from q labeled $1, ..., k$ indicating the possible values for v_i, or labeled with (i, x, y) where i is an internal node and the outedges are labeled with the possible values for $f_i(x, y)$. Each final state has a label from $[k]$ indicating the output v_1.

$Size(B)$ is the number of states in B.

A Turing machine M solving TEP(h, k) in space $s(h, k)$ can be simulated by a family of BPs of size $2^{O(s(h,k))}$ (the number of possible configurations of M).

$Size(h,k)$ is the number of states in the smallest deterministic BP solving TEP(h,k).

$Size_h(k) = Size(h,k)$ for fixed h.

Lemma $Size_h(k) = O(k^h)$
Proof: h pebbles suffice to pebble T_2^h, and for fixed h, the number of steps in the pebbling of T_2^h is constant.

This is the best upper bound known for the order of $Size_h(k)$.

Lemma: A lower bound of $Size_h(k) = \Omega(k^{r(h)})$ for some unbounded function $r(h)$ implies $L \neq \mathbf{LogDCFL}$.

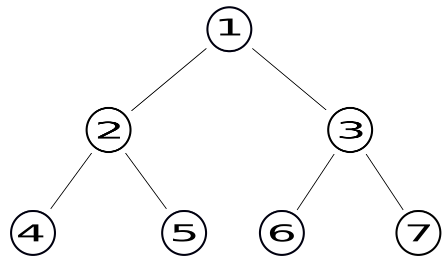
Recall best known upper bound:
$Size_h(k) = O(k^h)$

Best known lower bounds:
$Size_h(k) = \Omega(k^3)$ for each $h \geq 3$.
(Tight bounds are known for $h = 2$ and $h = 3$)

$h = 2$: $Size_2(k) = \Omega(k^2)$
This is obvious because each state of the BP can only make one query of the form (i, x, y), and there are k^2 possible values for (x, y).

$h = 3$: $Size_3(k) = \Omega(k^3)$
This is *not* obvious, because the number of input variables is only $O(k^2)$.
Proof I: Use Nečiporuk's method
Proof II: Use the "state sequence" method.

Nečiporuk's method counts the number of BPs on s states and compares this with the number of functions obtainable by various restrictions of $\text{TEP}_h(k)$. This method cannot beat $\Omega(n^2)$ states, and so cannot show TEP $\notin \mathbf{L}$.



Theorem: $Size_3(k) \geq k^3$

Proof: ("State Sequence" method)
For $r, s \in [k]$ let $E^{r,s}$ be the set of inputs I s.t.
- $f_1^I(x, y) = (x + y) \bmod k$
- $f_2^I(x, y) = f_3^I(x, y) = 0$ for all $(x, y) \neq (r, s)$
- $v_4^I = v_6^I = r$ and $v_5^I = v_7^I = s$

Thus $|E^{r,s}| = k^2$ because each $I \in E^{r,s}$ determined by v_2^I, v_3^I.

Let $\Gamma^{r,s}$ be the set of states which query either $f_2(r, s)$ or $f_3(r, s)$. It suffices to show
(*) $\qquad |\Gamma^{r,s}| \geq k$ for all $r, s \in [k]$.

Proof of (*): (γ^I, v_i^I) determines the output of $\mathcal{C}(I)$ (the computation on input I), where γ^I is the last state of $\mathcal{C}(I)$ in $\Gamma^{r,s}$, and i is the node queried by γ^I.

Thrifty Branching Programs

A deterministic BP solving $\mathsf{TEP}_h(k)$ is *thrifty* if for every query $f_i(x, y)$ (for every input), (x, y) are the values of the children of node i.

Thrifty BPs can implement black pebbling, and hence solve $\mathsf{TEP}_h(k)$ with $O(k^h)$ states. It turns out that this is also a lower bound.

Theorem: Thrifty deterministic BPs solving $\mathsf{TEP}_h(k)$ have $\Omega(k^h)$ states.

The proof is nontrivial.

Thus any BP beating the $O(k^h)$ upper bound must make queries $f_i(x, y)$ which are irrelevant to the value v_i of the node i.

Thrifty Hypothesis: Thrifty BPs are optimal among deterministic BPs solving $\mathsf{TEP}_h(k)$.

(Not true for solving the decision version of TEP)

Nondeterministic Branching Programs

Black/White Pebbling: A white pebble can be placed on any node at any time (representing a guess as to the value). The pebble can be removed if the node is a leaf, or both children have pebbles.

T_2^h can be B/W pebbled with $\lceil h/2 \rceil + 1$ pebbles. (This is optimal.)

Recall T_2^h requires h pebbles to black pebble it.

Nondeterministic Branching Programs

Black/White Pebbling: A white pebble can be placed on any node at any time (representing a guess as to the value). The pebble can be removed if the node is a leaf, or both children have pebbles.

T_2^h can be B/W pebbled with $\lceil h/2 \rceil + 1$ pebbles. (This is optimal.)

Recall T_2^h requires h pebbles to black pebble it.

Nondeterministic BPs implement B/W pebbling, so $NSize_h(k) = O(k^{\lceil h/2 \rceil + 1})$.

For $h = 3$ this gives $O(k^3)$ states, but best lower bound is $k^{2.5}$ states (via both Nečiporuk and 'state-sequence' methods).

This led us to discover "fractional pebbling".

T_2^3 can be B/W pebbled with 2.5 pebbles, so

$NSize_3(k) = \Theta(k^{2.5})$.

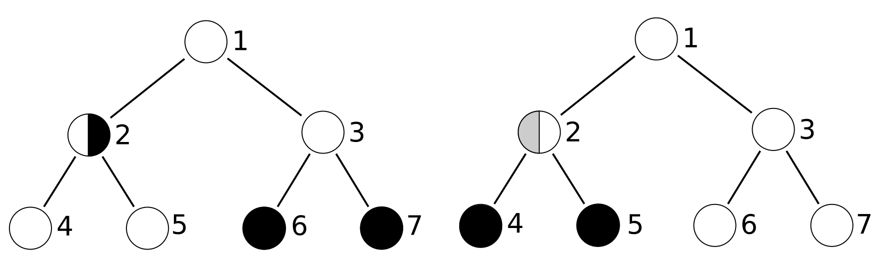
Fractional Pebbling

Fractional pebbling is like B/W pebbling, except now a node i can have a pair $(b(i), w(i))$ of real values, where

$$0 \leq b(i), w(i) \qquad b(i) + w(i) \leq 1$$

If both children of node i have total pebble value 1, then $w(i)$ can be set to 0, and any black fraction can be slid up from the children to increase $b(i)$.

The tree T_2^3 can be fractionally pebbled with 2.5 pebbles.



Theorem Thrifty nondeterministic BPs can implement fractional pebbling to solve $\mathsf{TEP}_h(k)$.

Theorem Bounds on fractional pebbling.
$\#\text{FRpebbles}(T_2^3) = 2.5$
$\#\text{FRpebbles}(T_2^4) = 3$
$h/2 - 1 \leq \#\text{FRpebbles}(T_2^h) \leq h/2 + 1$

Theorem(Repeat) Thrifty nondeterministic BPs can implement fractional pebbling.

Corollary $NSize_3(k) = \Theta(k^{2.5})$
$NSize_4(k) = O(k^3)$
$NSize_h(k) = O(k^{h/2+1}), h \geq 2$
(All upper bounds use thrifty BPs)

Theorem $ThriftyNSize_4(k) = \Theta(k^3)$

Open Question: Can nondeterministic Thrifty BPs beat fractional pebbling bound for $h > 4$?

(Recall that black pebbling is optimal for deterministic thrifty BPs.)

Conclusion

Thrifty Hypothesis: Thrifty BPs are optimal among deterministic k-way BPs solving $\mathsf{TEP}_h(k)$.
(i.e. $Size_h(k) = \Omega(k^h)$.)

In other words, the black pebbling method is the most space-efficient deterministic method for solving $\mathsf{TEP}_h(k)$.

A proof implies $\mathbf{L} \neq \mathbf{LogDCFL}$
(so $\mathbf{NC}^1 \subsetneq \mathbf{NC}^2$).

A disproof would involve a new space-efficient algorithm and would also be interesting
(think superconcentrators).

Next Step: Prove or disprove $Size_4(k) = \Omega(k^4)$

(Best known bound: $Size_4(k) = \Omega(k^3)$.)

Separating $\mathbf{L}$ from P is important!