Exact size of the smallest min-depth branching programs solving the Tree Evaluation Problem

Dustin Wehr

January 21, 2012

All the definitions needed to read this document are in the paper *Pebbles and Branching Programs for Tree Evaluation*, ACM Transactions on Computation Theory, Vol. 3, No. 2, Article 4, Publication date: January 2012.
You can find it here: `http://www.cs.toronto.edu/~wehr/research_docs/TCT_2012_Pebbles_and_Branching_Programs_for_the_Tree_Evaluation_Problem.pdf`

We write $\text{TE}^h(k)$ as an abbreviation for "$\text{BT}^h_2(k)$ or $\text{FT}^h_2(k)$". We show that a BP solving $\text{TE}^h(k)$ has minimum depth (defined below) if and only if it is both thrifty and read-once (Fact 3), and that the upper bound of $(k+1)^h - k$ non-output states for $\text{FT}^h_2(k)$ is the exact minimum for these (very restricted) BPs (Theorem 4).

Define the depth of a deterministic BP to be the maximum number of states visited by any input, with the output state included. It is easy to prove that depth 2^h is required to solve $\text{TE}^h(k)$ by considering those inputs all of whose internal node functions are quasigroups (see [Weh10] for a proof). Let us say a BP solving $\text{TE}^h(k)$ is **min-depth** if it has depth 2^h. We will use the following results from [Weh10]:

Lemma 1 *Every min-depth BP solving $TE^h(k)$ is thrifty.* [1]

Lemma 2 *For every input I to a min-depth BP solving $TE^h(k)$, the $2^h - 1$ input variables queried by I are exactly the $2^h - 1$ distinct thrifty input variables of I. Hence such a BP is read-once.*

Theorem 4 is the goal. We will not use the next fact in the proof, but it may be worth noting: Lemmas 1 and 2 charaterize the min-depth BPs solving $\text{TE}^h(k)$, in the following sense:

Fact 3 *A BP solving $TE^h(k)$ is min-depth iff it is both thrifty and read-once.*

Proof: The left-to-right direction follows from Lemmas 1 and 2. For the right-to-left direction, we use the fact from [Weh10] (page 10, second paragraph and lemma 4) that in a thrifty BP, every input must query all and only its $2^h - 1$ thrifty variables. Since the BP is also read-once, every input visits exactly $2^h - 1$ states (including an output state). ■

[1] Hence from the lower bound on thrifty programs in [Weh10], we get that every min-depth BP solving $\text{TE}^h(k)$ has at least k^h non-output states. This is the bound that we are slightly improving on here.

Theorem 4 *Every min-depth BP solving TE$^h(k)$ has at least $(k+1)^h - k$ non-output states.*

Proof: For B a min-depth BP that solves TE$^h(k)$ and for each $l \leq h$ let $\mathsf{States}(B,l)$ be the states of B that query a height-l node variable. By Lemma 5 the theorem follows if we can show for arbitrary such B that:

$$\begin{aligned} & |\mathsf{States}(B,1)| \geq (k+1)^{h-1} \\ \text{and} \quad & |\mathsf{States}(B,l)| \geq k^2(k+1)^{h-l} \quad \text{for } 2 \leq l \leq h \end{aligned} \tag{1}$$

Lemma 5

$$(k+1)^h - k = (k+1)^{h-1} + k^2 \sum_{l=2}^{h} (k+1)^{h-l}$$

Proof: Since $(k+1)^h - (k+1)^{h-1} = k(k+1)^{h-1}$, after subtracting $(k+1)^{h-1}$ from both sides we can write the equations as:

$$k(k+1)^{h-1} - k = k^2 \sum_{l=2}^{h} (k+1)^{h-l}$$

We add k to both sides, divide both sides by k, and then prove the resulting family of equations

$$(k+1)^{h-1} = 1 + k \sum_{l=2}^{h} (k+1)^{h-l}$$

by induction on $h \geq 2$. For $h = 2$ it is clear. Now let $h \geq 3$ be arbitrary and assume the equation holds for $h-1$.

$$\begin{aligned} 1 + k\textstyle\sum_{l=2}^{h}(k+1)^{h-l} &= 1 + k(k+1)^{h-2} + \ldots + k(k+1) + k \\ &= (k+1) + k(k+1)^{h-2} + \ldots + k(k+1) \\ &= (k+1)[1 + k(k+1)^{h-3} + \ldots + k] \\ &= (k+1)[1 + k\textstyle\sum_{l=2}^{h-1}(k+1)^{(h-1)-l}] \\ &= (k+1)(k+1)^{h-2} \quad \text{by I.H.} \end{aligned}$$

■

The next lemma shows that it suffices to prove the lower bound on the number of states that query height-2 variables.

Lemma 6 *If $|\mathsf{States}(B,2)| \geq k^2(k+1)^{h-2}$ for every h and B, then (1) holds for every h and B.*

Proof: Assume the hypothesis holds. Let B be a min-depth BP that solves $\text{BT}_2^h(k)$ (the proof is the same for $\text{FT}_2^h(k)$).

To show $|\mathsf{States}(B,1)| \geq (k+1)^{h-1}$, we transform B into a min-depth BP B' that solves $\text{BT}_2^{h+1}(k)$. Replace each state that queries a leaf variable with a copy of the BP for $\text{FT}_2^2(k)$ in the obvious way. Each such replacement involves adding k^2 height-2 querying states. Hence if B has less than $(k+1)^{h-1}$ leaf-querying states then B' has less than $k^2(k+1)^{h-1} = k^2(k+1)^{(h+1)-2}$ height-2 querying states, which contradicts the hypothesis. We still need to argue that we haven't increased the depth by too much. Since every input to B visits exactly 2^{h-1} leaf-querying states, and B has depth 2^h, it is not hard to see that every computation path in B' has length $2^h + 2 \cdot 2^{h-1} = 2^{h+1}$.

Now we assume $h \geq 3$ and give the argument for $|\mathsf{States}(B{,}3)| \geq k^2(k{+}1)^{h-3}$. It will be clear how to generalize it to get $|\mathsf{States}(B{,}l)| \geq k^2(k+1)^{h-l}$ for all $3 \leq l \leq h$. We transform B into a min-depth BP B' that solves $\mathrm{BT}_2^{h-1}(k)$. The height-3 querying states of B will become the height-2 querying states for B'. Hence B must have at least $k^2(k+1)^{h-3}$ height-3 querying states, since otherwise B' would have fewer than $k^2(k+1)^{(h-1)-2}$ height-2 querying states, which contradicts the hypothesis. Let E_1 be the inputs to B all of whose leaf values are 1. The computation path of each input I' to B' will be derived in a simple way from the computation path of some $I \in E_1$. First, remove every state in B that queries a variable in

$$\{f_u(a,b) \mid u \text{ is a height-2 node and } \langle a,b\rangle \neq \langle 1,1\rangle\}$$

Also, for every leaf-querying state q, remove the $k-1$ out-edges of q labeled $2,\ldots,k$. Removing those states and edges does not break the path of any input in E_1; this is clear for the edges, and for the states it follows from the thrifty property (Lemma 1). We need to be a bit more careful about removing the leaf-querying states, since they *are* visited by inputs in E_1. Place a token on the start state, which must be a leaf state by thriftiness. Repeat the following while there remains some leaf-querying state q. We know q has a single out-edge labeled 1; let q' be the state that edge points to. Redirect all the edges going into q so that they go into q' instead. If the token is on q then move it to q'. Finally remove q. When this process finishes, the token will be resting on a height-2 querying state q^* with no in-edges; specifically some state that queries a variable in the set $V = \{f_u(1,1) \mid u \text{ is a height-2 node}\}$. The last step is just to relabel the states that query variables in V: for each height-2 node u, change every occurrence of the state label $f_u(1,1)$ to $l_{\lfloor u/2 \rfloor}$. The start state of the resulting BP B' is q^*.

Now we need to argue that we have *decreased* the depth enough. Consider an input $I \in E_1$ to B. The construction above determines the input I' to B' that I gets mapped to. Since I is thrifty, it does not visit any of the height-2 querying states in B that were removed. We also know that I visits exactly 2^{h-1} states in B that query leaf variables. It follows that the computation path of I' is shorter than that of I by exactly 2^{h-1}, and so it has length 2^{h-1}. ■

Fix h, k and a depth 2^h BP B that solves $\mathrm{TE}^h(k)$. Let E be the set of inputs to B. We want to show B has at least $k^2(k+1)^{h-2}$ height-2 querying states. Let Q^2 be the states of B that query a height-2 variable. For $t \leq 2^{h-2}$ let Q^2_t be the states $q \in Q^2$ such that q is the t-th Q^2-state visited by some input to B.

Lemma 7 *$Q^2_{t_1} \cap Q^2_{t_2} = \emptyset$ for distinct $t_1, t_2 \leq 2^{h-2}$.*

Proof: Otherwise, there are t_1, t_2 with $t_2 < t_1$ such that there is a state q that is the t_1-th state visited by some input I_1 and the t_2-th state visited by some other input $I_2 \neq I_1$. Since B has depth 2^h, by Lemma 2 we get that I_1 visits $2^h - t_1$ states after q and I_2 visits $2^h - t_2$ states after q. However, since B is read-once, every syntactic computation path is a semantic (i.e. consistent) computation path, so there must be some input I_3 whose computation path is the same as that of I_1 up to q, and then the same as that of I_2 from q until the output. But then the computation path of I_3 has length $t_1 + 2^h - t_2 > 2^h$, a contradiction. ■

Next, for $2 \leq l \leq h$ we will define a sequence $\vec{z}_l$ of positive integers of length 2^{h-l}. To see the purpose of $\vec{z}_l$, consider the min-depth BP B^* for $\mathrm{FT}_2^4(k)$ that we get from the optimal black pebbling that always pebbles the left subtree before the right subtree. If you draw the tree minus the leaves, and label each node u with the number of states in B^* that query a u-node, then your picture should look like this:

$$\begin{array}{ccccccc} & & & k^2 & & & \\ & k^2 & & & & k^3 & \\ k^2 & & k^3 & & k^3 & & k^4 \end{array}$$

$\vec{z}_l$ gives the exponents of the height l nodes in such a picture, read from left to right. So for $h = 4$, we have $\vec{z}_2 = 2, 3, 3, 4$. Formally: $\vec{z}_h = 2$, and for $2 \leq l \leq h - 1$, $\vec{z}_l$ is $\vec{z}_{l+1}$ followed by the sequence obtained by adding 1 to each element of $\vec{z}_{l+1}$. We write $\vec{z}_l(t)$ for the t-th element of $\vec{z}_l$. Later we will appeal to the following equivalent definition of $\vec{z}_2$.

Fact 8 *Let #ones(t) be the number of 1s in the binary representation of $t \geq 0$. Then $\vec{z}_2(t) = 2 + \#ones(t - 1)$ for $t \geq 1$.*

Eventually we will get the quantity $k^2(k + 1)^{h-2}$ using the following simple lemma:

Lemma 9 $\sum_{t=1}^{2^{h-l}} k^{\vec{z}_l(t)} = k^2(k + 1)^{h-l}$ *for every* $2 \leq l \leq h$

Proof: Easy by induction on $h - l$. ■

We assign to each input I a pebbling sequence C^I of length exactly 2^h such that the following Property 1 holds. Because of the depth restriction, which implies B is thrifty (Lemma 1), there is exactly one way to do this. The definition follows the statement of Property 1.

Property 1 *For each pair of adjacent states q_1, q_2 on the computation path of I, if C_1^I and C_2^I are the associated pebbling configurations, then a pebble is added to a node u in the move $C_1^I \to C_2^I$ iff q_1 queries u, and a pebble is removed from a non-root node u in the move $C_1^I \to C_2^I$ iff q_1 queries the parent of u.*

Fix I and let P be the computation path of I. The pebbling sequence assignment can be described inductively by starting with the last state on P and working backwards. The pebbling configuration for the last state in P (i.e. the output state) has just a black pebble on the root. Assume we have defined the pebbling configurations for q and every state following q on P, and let q' be the state before q on P. This inductive construction, together with Lemmas 1 and 2, ensures that q' queries some node u that is pebbled in q (see page 10 of [Weh10] for a more-detailed argument). The pebbling configuration for q' is obtained from the configuration for q by removing the pebble from u and adding pebbles to both children of u (if u is an internal node - otherwise you only remove the pebble from u).

We will use the next lemma in the proof of Lemma 11.

Lemma 10 *For every input I and $t \leq 2^{h-2}$, if C is the pebbling configuration associated with the t-th Q^2-state visited by I, then there are at least $\vec{z}_2(t)$ pebbled nodes in C.*

Proof:

Let C and t be as in the statement of the Lemma, and let u be the height 2 node that gets pebbled in the next configuration after C. By Property 1, the two children of u are pebbled in C. Also by Property 1, there are exactly $t-1$ height 2 nodes –namely, the height 2 nodes pebbled earlier– that are "covered" by a pebbled node in C, meaning either v or some ancestor of v is pebbled in C. Recall #ones from Fact 8. It is not hard to see that #ones$(t-1)$ is the smallest number m such that there exists a set of m nodes U which, if pebbled, would cover *exactly* $t-1$ height 2 nodes; #ones$(t-1)$ is the number of terms needed to represent $t-1$ as a sum of distinct powers of 2, and the presence of the term 2^i corresponds to a node in U at height $2+i$. Now, since the children of u are pebbled in C, and cannot cover a height 2 node, C must have a total of at least $2+$#ones$(t-1)$ pebbled nodes. Then by Fact 8 we conclude C has at least $\vec{z}_2(t)$ pebbled nodes. ■

Recall E is the set of all inputs to the BP B. Let E_q be the inputs that visit state q.

Lemma 11 *For all $t \leq 2^{h-2}$ and q in Q^2_t: $|E_q| \leq |E|/k^{\vec{z}_2(t)}$.*

Proof: (*sketch*)

Here is the **idea**. Given Lemma 10, this proof is an easy adaptation of the thrifty lower bound proof from [Weh10]. In fact, for $t = 2^{h-2}$ it is exactly the same proof, since $\vec{z}_2(2^{h-2}) = h$ and for $t = 2^{h-2}$ we are counting the states q such that q is the last height-2 querying state visited by some input. Note it is necessary to use the fact that for every input I and all of the pebbling configurations C that we assigned to I, there is at most one pebbled node in C on any path from the root to a leaf in T_h. ■

Using Lemma 7, we have:

$$|Q^2| = \sum_{t=1}^{2^{h-2}} |Q^2_t|$$

Clearly $\{E_q\}_{q \in Q^2_t}$ is a partition of E for every $t \leq 2^{h-2}$. So from Lemma 11 we get that $|Q^2_t| \geq k^{\vec{z}_2(t)}$ for every $t \leq 2^{h-2}$. Combining this with the previous equation we have:

$$|Q^2| \geq \sum_{t=1}^{2^{h-2}} k^{\vec{z}_2(t)}$$

Finally, combining the previous equation with Lemma 9 (for $l = 2$), we finish the proof:

$$|Q^2| \geq k^2(k+1)^{h-2}$$

■

References

[Weh10] Dustin Wehr. Pebbling and branching programs solving the tree evaluation problem, 2010. arXiv:1002.4676.