New Results for Tree Evaluation

Sui Man Chan James Cook Stephen Cook Phuong Nguyen
Dustin Wehr

December 20, 2010

Work in Progress – Not for Distribution

Contents

1 Leaf Queries

The next two theorems show that for $h \geq 3$, proving a lower bound of $\Omega(k^e)$ on the number of states for BPs solving TEP for T_2^h is essentially equivalent to proving a lower bound of $\Omega(k^{e-2})$ on the number of states making leaf queries for BPs solving TEP for T_2^{h-1}.

Theorem 1 *For $h \geq 3$, any BP with s states solving the TEP for T_2^h can be transformed to a BP solving the TEP for T_2^{h-1} in which the number of states querying leaves is at most s/k^2. If the original BP is deterministic, then so is the transformed BP.*

Proof: Let B be a BP with s states which solves the TEP for T_2^h. Then for some $r, r' \in [k]$ there are at most s/k^2 states which make queries of the form $f_j(r, r')$ for some node j at level two (i.e. the children of j are leaves). Now we construct a BP B' with at most s/k^2 states querying leaves which solves T_2^{h-1}. The idea is that B' will simulate B for the case in which, for each level 2 node j, f_j is 1 except possibly for $f_j(r, r')$, and $f_j(r, r')$ is the value of the leaf j in T_2^{h-1}, and further the children (leaves) of node j have values (r, r').

Start by replacing every state of B that queries $f_j(r, r')$ by a state that queries the leaf j (of T_2^{h-1}). Now remove every state γ that queries $f_j(a, b)$ for some level 2 node j, where $(a, b) \neq (r, r')$, and reroute any edge into γ by sending it to the destination of the outedge of γ labelled 1. Finally remove every state δ which queries a leaf of T_2^h and replace every edge into δ by sending it to the destination of the outedge of γ labelled either r or r', depending on whether the leaf is a left child or a right child. ■

Theorem 2 *For $h \geq 3$, any BP which solves T_2^{h-1} with s states which query leaves and $O(sk^2)$ states altogether can be converted to a BP which solves T_2^h with $O(sk^2)$ states. If the original BP is deterministic, then so is the transformed BP.*

Proof: Simply replace every leaf query by a subprogram with $O(k^2)$ states which evaluates the corresponding level 2 node in T_2^h. ■

2 Lower Bound for $BT_2^3(k)$

Here we give an alternative proof for the following theorem. The earlier proof can be found in [CMW$^+$10].

Theorem 3 *Every deterministic BP solving $BT_2^3(k)$ has at least $k^3/\log_2 k$ states for sufficiently large k.*

Proof: Let B be a deterministic BP solving $BT_2^2(k)$. By Theorem 1 it suffices to show that B has at least $k/\log k$ leaf queries, for large k (we use $\log k$ for $\log_2 k$).

We say that a state q of B is a V*-state* if either q queries a leaf v_2 or v_3, or q is an output state. A state p is an F*-state* if p queries f_1, and B has an edge from some V-state to p.

We suppose that the initial state is a V state, by adding an extra state if necessary.

Let V be the set of V-states, and let F be the set of F-states. Let $s = |V|$ (the cardinality of V). Then $|F| \leq ks$.

For each $f : [k] \times [k] \rightarrow \{0,1\}$ define

$$\psi[f] : F \rightarrow V$$

by setting $\psi[f](p)$ to be the first V-state encountered in the computation starting from state p, when the root function f_1 is f.

The number of possible distinct functions $\psi[f]$ is at most

$$|V|^{|F|} \leq s^{ks}$$

as f varies. It is clear that distinct functions f must give distinct functions $\psi[f]$, since otherwise the computations of two distinct functions would give the same output for any pair of leaf values (v_2, v_3). Since there are 2^{k^2} choices for f, this gives

$$s^{ks} \geq 2^{k^2}$$

Thus $ks \log s \geq k^2$, so

$$s \geq k/\log s$$

From this we can conclude

$$s \geq k/\log k \ + 3$$

for sufficiently large k. The theorem follows, since s is at most 3 more than the number of leaf queries (the two output states and the initial state). ■

3 Sequential BPs

Definition 4 *A BP solving $FT_2^h(k)$ or $BT_2^h(k)$ is* sequential *if for every computation on any instance all queries to the left principle subtree (rooted at node 2) precede every query to the right principle subtree.*

Our goal is to prove a lower bound of $\Omega(k^4)$ on the size of sequential deterministic BPs solving $FT_2^4(k)$. We intend to prove a lower bound for the following problem, which implies the same lower bound for the sequential case.

Definition 5 *The problem $RES_2^h(k)$ is the same as $FT_2^h(k)$ except the root function f_1 is fixed to be addition mod k and the subtree rooted at node 2 is replaced simply by the leaf 2. A BP solving the problem must start by querying v_2, and may not query v_2 after that.*

Lemma 6 *Any sequential deterministic BP solving $FT_2^h(k)$ for $h \geq 3$ can be converted to a deterministic BP solving $RES_2^h(k)$ with fewer states.*

Proof: Let B be a sequential deterministic BP solving $FT_2^h(k)$. For $x \in [k]$ let I^x be the input to B in which the left-child function f_2 is the constant function with value x, the root function f_1 is addition mod k, and all other functions and leaf values are identically 1. Let q_x be the first state in the computation of B on input I^x that queries some node in the right principle subtree (rooted at node 3). We convert B to a BP B' solving $RES_2^h(k)$ as follows. Delete all states preceding q_x in the computation of B on input I^x, and add a new initial state which queries the leaf node 2 of an input to $RES_2^h(k)$, whose x-th outedge leads to q_x, $1 \leq x \leq k$. It is easy to see that B' has fewer states than B and correctly solves $RES_2^h(k)$. ■

We can further simplify our lower bound problem to that of counting leaf queries for height 3 trees by applying the proof of Theorem 1.

Definition 7 *The problem $RR3(k)$ is the same as $RES_2^4(k)$ except now the children of nodes v_6 and v_7 are removed, so that these two nodes become leaves. As before, a BP solving this problem must start by querying v_2, and may not query v_2 after that.*

The next lemma is proved in the same way as Theorem 1.

Lemma 8 *To prove a lower bound of s states for sequential BPs solving the TEP for T_2^4 it suffices to prove a lower bound of s/k^2 on the number of states querying the leaves v_6 and v_7 for BPs solving $RR3(k)$ as in definition 7.*

Thus to prove a lower bound $\Omega(k^4)$ states on the size of sequential deterministic BPs solving $FT_2^4(k)$ it suffices to prove a lower bound bound of $\Omega(k^2)$ on the number of states querying v_6 and v_7 for deterministic BPs solving $RR3(k)$ The next two subsections prove such lower bounds for restricted versions of $RR3(k)$.

3.1 f_3 is fixed to some random function

Here we prove lower bounds of $\Omega(k^2)$ for restricted versions of $RR3(k)$ by assuming that f_3 is a fixed random function. The restrictions involve the number of alternations between v_6 and v_7 queries during a computation.

NOTE: This subsection is not very promising.

Theorem 9 *Let B be a deterministic BP which solves a version of $RR3(k)$ under the restriction that initially B queries v_2 and v_6 and after that it does not query either node. Then B has at least k^2 states which query v_7, for sufficiently large k.*

Proof: Let B be as in the theorem, and assume that f_3 is a fixed function chosen at random. We will show that (for sufficiently large k), for almost all choices of f_3, if B has fewer than k^2 states that query v_7 then B makes a mistake for some choice of v_2, v_6, v_7.

Let + denote addition mod k.

For $v_2, v_6 \in [k]$ let the unary function $G_{\langle v_2,v_6 \rangle}$ taking $[k]$ into $[k]$ be defined by

$$G_{\langle v_2,v_6 \rangle}(z) = v_2 + f_3(v_6, z)$$

Thus $G_{\langle v_2,v_6 \rangle}(v_7)$ is the solution to the problem $RR3(k)$.

The following result is easy to prove.

CLAIM: If f_3 is chosen uniformly at random, then with high probability each of the k^2 functions $g_{\langle v_2,v_6 \rangle}$, for $v_2, v_6 \in [k]$, is distinct from the others, for sufficiently large k.

Note that in the computation of B on any input in $RR3$, the output is determined by the last query to v_7, since v_2 and v_6 may not be subsequently queried, and all other query values are fixed. Thus this last state (in effect) computes the function $G_{\langle v_2,v_6 \rangle}(v_7)$. It follows from the CLAIM that there is a choice for f_3 (in fact almost all choices work) such that there must be at least k^2 states which query v_7. ■

Lemma 10 *Any deterministic BP solving $RR3(k)$ has at least k states which query v_7.*

Proof: Assume random f_3 as in the proof of Theorem 9. For any fixed $v_6, v_7 \in [k]$ let q_x be the first state which queries v_7 when $v_2 = x$. Then the states $q_1, \ldots, q_k$ must all be distinct, because the output is $v_2 + f_3(v_6, v_7)$. ■

Theorem 11 *Let B be a deterministic BP solving $RR3(k)$ which has at most k states which query v_7. Assume further that for each input, B queries v_7 exactly once. Then B has $\Omega(k^2)$ states which query v_6.*

Proof: Assume that B satisfies the assumptions in the theorem. Then by the lemma, B has exactly k states which query v_7. To prove the theorem it suffices to show that B has at least $k(k+1)/2$ states which query f_6, for sufficiently large k.

Fix some random function f_3. We may assume

$$\text{For each } v_6 \in [k],\ f_3(v_6, v_7) \text{ is not constant in } v_7 \tag{1}$$

because this happens with high probability. Note that there are k^3 inputs with this fixed f_3, each input is specified by the triple v_2, v_6, v_7. Note that the output function (value of v_1) is

$$Out(v_2, v_6, v_7) = v_2 + f_3(v_6, v_7) \tag{2}$$

Let $q_1, \ldots, q_k$ be the k states which query v_7.

Claim 1: For each state $q_i, 1 \le i \le k$, and for each $v_6 \in [k]$ there is exactly one $v_2 \in [k]$ such that the computation on each input $(v_2, v_6, *)$ leads to q_i.

Proof of Claim 1: By assumption, for each pair (v_2, v_6) the computation on input $(v_2, v_6, *)$ (where $*$ stands for any v_7) leads to exactly one of the k states $q_1, \ldots, q_k$. Hence if the Claim is false, then there is $v_6 \in [k]$ and distinct $v_2, v_2' \in [k]$ such that both of the inputs $(v_2, v_6, *)$ and $(v_2', v_6, *)$ lead to the same state q_j. But then for any v_7 the outputs for (v_2, v_6, v_7) and (v_2', v_6, v_7) would be the same, which violates (2).

Now for $1 \leq i \leq k$ let $\hat{q}_i(v_6)$ be the unique value of v_2 for which the computation input $(v_2, v_6, *)$ leads to the state q_i. By the proof of Claim 1 we have

Claim 2: For each $v_6 \in [k]$ and $1 \leq i < j \leq k$

$$\hat{q}_i(v_6) \neq \hat{q}_j(v_6)$$

Now let $p_1, \ldots, p_r$ denote the states which query v_6. Then the pair $\langle v_7, q_i \rangle$ uniquely determines the next state of the form p_ℓ in the computation after q_i, where we denote by p_0 the 'empty state' in case the computation terminates without reaching any of the states $p_1, \ldots, p_r$ after q_i. Thus for $i, c \in [k]$ we may define the function $G_{\langle i,c \rangle}(v_6)$ to be the output of the computation after state q_i is reached when $v_7 = c$, since by assumption, after p_ℓ the computation does not query either v_2 or v_7. Note that the state p_ℓ completely determines the function $G_{\langle i,c \rangle}$. We have

$$\begin{aligned} G_{\langle i,c \rangle}(v_6) &= Out(\hat{q}_i(v_6), v_6, c) \\ &= \hat{q}_i(v_6) + f_3(v_6, c) \end{aligned}$$

We will show that for large k there must be at least $k(k+1)/2$ different functions $G_{\langle i,v_7 \rangle}$ as q_i and v_7 vary, and hence there must be at least $k(k+1)/2$ states of the form p_ℓ.

[NOTE: For this argument it may be easier to let $f_3(a, b) = (a+b)^3$, as in Siuman's Read-Once proof]

Let us abbreviate $f_3(v_6, c)$ as a function of v_6 by f^c, so

$$G_{\langle i,c \rangle} = \hat{q}_i + f^c$$

We now define a sequence

$$S_1 \subset S_2 \subset \cdots \subset S_k$$

of sets of functions of the form $G_{\langle i,c \rangle}$ as follows:

$$S_i = \{G_{\langle j,c \rangle} \mid 1 \leq j \leq i, c \in [k]\}$$

Claim 3: For large k, for random f_3, with high probability $|S_1| = k$, and $|S_{i+1} \setminus S_i| \geq k - i$ for $i = 1, \ldots k-1$.

Proof: Since f_3 is chosen uniformly at random, the unary functions $f^1, f^2, \ldots, f^k$ are chosen independently uniformly at random. Hence very likely for each $i \in [k]$ the functions

$$G_{\langle i,1 \rangle}, G_{\langle i,2 \rangle}, \ldots, G_{\langle i,k \rangle}$$

are all distinct, so in particular $|S_1| = k$. Also for all $i, j, c, d, c', d' \in [k]$ with $j < i+1$ and $(c, d) \neq (c', d')$ it is very unlikely that both $G_{\langle i+1,c \rangle} = G_{\langle j,d \rangle}$ and $G_{\langle i+1,c' \rangle} = G_{\langle j,d' \rangle}$, since otherwise $f^c - f^d = f^{c'} - f^{d'}$. Thus likely at most i of the functions of the form $G_{\langle i+1,c \rangle}$ (for $c = 1, \ldots, k$) occur in S_i. This proves Claim 3.

From Claim 3 it follows that likely there are at least $k(k+1)/2$ distinct functions of the form $G_{\langle i,c \rangle}$, and hence at least $k(k+1)/2 - 1$ states $p_1, \ldots, p_r$. ■

Alternative Proof of the Last Part:
We give an alternative proof that there must be at least $k^2/2$ different functions of the form $G_{\langle i,c\rangle}$. The main point is the same as before, namely:
Fact: For all $i, j, c, d, c', d' \in [k]$ with $i \neq j$ and $(c,d) \neq (c',d')$ it is very unlikely that both $G_{\langle i,c\rangle} = G_{\langle j,d\rangle}$ and $G_{\langle i,c'\rangle} = G_{\langle j,d'\rangle}$

The reason is that otherwise $f^c - f^d = f^{c'} - f^{d'}$.

It follows easily from the Fact that there can be at most $\binom{k}{2}$ equal pairs $G_{\langle i,c\rangle}, G_{\langle i',c'\rangle}$ (for distinct pairs $(i,c),(i',c')$). Now write $\langle i,c\rangle \sim \langle i',c'\rangle$ if $G_{\langle i,c\rangle} = G_{\langle i',c'\rangle}$, and let s be the number of equivalence classes among the k^2 pairs $\langle i,c\rangle$. We want to get a lower bound on s. Note that given s, the number of equal pairs is minimized when all equivalence classes have the same size. Then each class has k^2/s members, and so there are

$$\binom{k^2/s}{2}s$$

equal pairs. Thus

$$\binom{k^2/s}{2}s \leq \binom{k}{2}$$

so $s \geq k^3/(2k-1) > k^2/2$.

Generalizing Theorem 11

Now suppose that instead of k states learning v_7 we have t states $q_1, \ldots, q_t$ learning v_7, where

$$t = k^{2-\epsilon}$$

for some $\epsilon > 0$. Then we can define the function

$$\hat{q}_i(v_6) = v_2$$

as before (because the pair q_i, v_6 determines v_2), but now $\hat{q}_i$ is a partial function in general. If we set

$$D_i = domain(\hat{q}_i)$$

then $[k] \times [k]$ (think the set of all pairs (v_6, v_2)) is the disjoint union of sets E_i where

$$E_i = \{(v_6, \hat{q}_i(v_6)) \mid v_6 \in D_i\}$$

so

$$\sum_{i=1}^{t} |D_i| = k^2 \tag{3}$$

Assumption:
We partition $[k]$ into $k^{1-\epsilon}$ blocks of size k^ϵ, and each domain D_i is equal to one of these blocks.

Now pick any of these blocks D, and let

$$I = \{i \mid D_i = D\}$$

Then $|I| \geq k$, since for any fixed $v_6 \in D$, $\hat{q}_i(v_6)$ ranges over all k values of v_2 as i ranges over I.

In fact $|I| = k$, since $D \times [k]$ is the disjoint union of the sets $E_i, i \in I$ (see proof of (3)

Now reasoning as in the proof of Theorem 11 there must be at least $k^2/2$ functions in the set

$$\{G_{\langle i,c\rangle} \mid i \in I, c \in [k]\}$$

and hence there must be at least $k^2/2$ states p_i.
QED

Discussion: In the above we set $t = k^{2-\epsilon}$, which in general is not an integer. We will think of k as the independent variable, and other parameters such as t are functions of k. Then we should write

$$t \sim k^{2-\epsilon}$$

instead of $t = k^{2-\epsilon}$, where $f(k) \sim g(k)$ means

$$\lim_k f(k)/g(k) = 1$$

Lemma 12 *Fix $\epsilon > 0$. Given subsets $D_1, \ldots, D_t$ of $[k]$, where $t \sim k^{2-\epsilon}$, such that every $x \in [k]$ occurs in exactly k of the subsets, the average size $AveInt(k)$ of intersections $D_i \cap D_j, i \neq j$ is*

$$AveInt(k) \sim 1/k^{1-2\epsilon}$$

Proof: For each $j_0 \in [t]$ we have

$$\sum_{i \neq j_0} |D_i \cap D_{j_0}| = (k-1)|D_{j_0}|$$

since each $x \in D_{j_0}$ occurs in exactly k sets. Hence

$$\begin{aligned} \sum_{i \neq j} |D_i \cap D_j| &= (k-1) \sum_j |D_j| \\ &= k^2(k-1) \end{aligned}$$

by (3). Since there are $t(t-1)$ pairs i, j with $i \neq j$, the average intersection size is

$$AveInt(k) = k^2(k-1)/t(t-1) \sim k^3/(k^{2-\epsilon})^2 = 1/k^{1-2\epsilon}$$

■

Observation: Consider the setup of Theorem 6, except instead of allowing just k states which query v_7 we allow some large number t states $q_1, \ldots, q_t$ which query v_7. As before, each computation must query v_7 exactly once. Assume further that none of the states $q_1, \ldots q_t$ distinguishes between $v_6 = 1$ and $v_6 = 2$. (That is, if $v_6 \in \{1, 2\}$ then the state q_i reached depends only on v_2.) Then there exists f_3 such that there must be at least k^2 states (after $q_1, \ldots, q_t$) which query v_6.

Proof: We design $f_3(v_6, v_7)$ so that there are k^2 output functions Out_{v_2,v_7} of the form

$$Out_{v_2,v_7}(v_6) = v_2 + f_3(v_6, v_7)$$

restricted to $v_6 \in \{1, 2\}$, as the pair (v_2, v_7) varies. For example, we can take $f_3(v_6, v_7) = 1 + (v_6 - 1) \cdot v_7$. Then there must be a distinct state which queries v_6 for each pair v_2, v_7.
QED

3.2 The input includes f_3

Here we prove lower bounds of $\Omega(k^2)$ on the number of states querying the leaves v_6 and v_7 for BPs which solve $RR3(k)$ (Definition 7) under various restrictions concerning the alternations of states querying leaves v_6 and v_7 and states querying f_3 during a computation. Recall that by Lemma 8 a lower bound of $\Omega(k^2)$ on the number of states querying v_6 and v_7 for deterministic BPs solving $RR3(k)$ (with no such restriction on the alternations) implies a lower bound of $\Omega(k^4)$ for sequential deterministic BPs solving $FT_2^4(k)$.

Here we assume that our BPs are dags in which the states are organized into layers. Each dag has a single source node which queries v_2, and after that each layer consists either of states which query leaf nodes (i.e. v_6 and v_7) (called *V-layers*) or states which query f_3 (called *f_3-layers* or *F-layers*). Our goal is to prove a lower bound on the total number of states in all V-layers (i.e. the total number of 'V-states').

3.2.1 The VF, FV, and VFV cases

Theorem 13 *Any deterministic BP solving $RR3(k)$ which consists only of a V-layer followed by a f_3 layer has at least k^2 V-states.*

Proof: For each triple (v_2, v_6, v_7) associate the first state in the computation with input (v_2, v_6, v_7, f_3) (for any f_3) which queries f_3. These states must all be distinct since for any two distinct triples there is some f_3 which gives different output values for the two. Hence there must be at least k^3 such initial f_3 states. It follows that there must be at least k^2 V-states, because each such V-state has only k outedges and each initial f_3 state is reached by at least one outedge from a V-state. ■

Theorem 14 *Any deterministic BP solving $RR3(k)$ which consists only of an f_3 layer followed by a V-layer has at least k^{k^2} V-states.*

Proof: The proof is similar to the proof of the previous theorem. Fix $v_2 = 1$. There are k^{k^2} possible function f_3. For each such function f_3 the computation on input f_3 must reach a distinct initial V-state, since any two distinct functions f_3 differ on some argument (v_6, v_7), so the output differs. ■

Theorem 15 *Any deterministic BP solving $RR3(k)$ which consists only of a V-layer followed by an f_3-layer followed by a V-layer has at least k^2 V-states.*

Proof: Let us refer to the three layers as layer I, layer II, and layer III, respectively. We will show that either layer I or layer III has at least k^2 V-states.

Suppose there are fewer than k^2 states in layer I. Then there exist distinct triples (v_2, v_6, v_7) and (v_2', v_6', v_7') such that the computations on both triples lead to the same initial state p of layer II (see the proof of Theorem 13). Note that the two pairs (v_6, v_7) and (v_6', v_7') cannot be the same, since otherwise $v_2 \neq v_2'$ so the two outputs $v_2 + f_3(v_6, v_7)$ and $v_2' + f_3(v_6, v_7)$ would be different for any f_3, but no query in either of the layers II and III could distinguish them.

Note that the computation from state p to the first state q of layer III depends only on f_3. It suffices to prove the following.

Claim: There must be a distinct initial state q of layer III for each of the k^2 possibilities for the pair of outputs $(v_2 + f_3(v_6, v_7), v_2' + f_3(v_6', v_7'))$ as f_3 varies.

For otherwise there exists functions f_3 and $\hat{f}_3$ such that both f_3 and $\hat{f}_3$ end up in the same state q starting from state p and either $v_2 + f_3(v_6, v_7) \neq v_2 + \hat{f}_3(v_6, v_7)$ or $v_2' + f_3(v_6', v_7') \neq v_2' + \hat{f}_3(v_6', v_7')$. In the first case the BP cannot distinguish between the two inputs (v_2, v_6, v_7, f_3) and $(v_2, v_6, v_7, \hat{f}_3)$ and in the second case it cannot distinguish between the two inputs (v_2', v_6', v_7', f_3) and $(v_2', v_6', v_7', \hat{f}_3)$, but in either case it should distinguish. ■

Definition 16 *The* Boolean $RR3(k)$ *problem is the same as* $RR3(k)$ *except the output is to determine whether the value of the root is 1.*

The next two results show that $\Theta(k^2/\log k)$ V-states are necessary and sufficient for a three-layer (V-f_3-V) deterministic BP to solve the Boolean $RR3(k)$ problem.

Theorem 17 *The Boolean $RR3(k)$ problem can be solved by a deterministic BP with $O(k^2/\log k)$ V-states, where the BP has three layers as in Theorem 15: a V-layer followed by and f_3-layer followed by a V-layer.*

Proof: The idea is to divide the k^2 possible pairs (v_6, v_7) into $O(k^2/\log k)$ equivalence classes of size about $\log k$ each. Layer I remembers v_2, queries v_6 and v_7, and passes on v_2 together with the equivalence class of (v_6, v_7) to the initial states of Layer II. Thus there are $O(k^3/\log k)$ initial states of Layer II, and Layer I has a total of $O(k^2/\log k)$ states.

Layer II evaluates f_3 at (v_6, v_7) for each pair (v_6, v_7) in the relevant equivalence class, remembers the subset of pairs which make $v_2 + f_3(v_6, v_7) = 1$, and passes this subset to one of the initial states of Layer III.

We assume that there are at most $\log_2 k - \log_2 \log k$ members of each equivalence class, so there are at most $k/\log k$ initial states of Layer III. For each initial state (i.e. each subset) the BP queries v_6 and then v_7 and determines whether (v_6, v_7) is in the accepting subset and hence whether to accept the input. Thus Layer III has $k/\log k$ initial states which query v_6 and $k^2/\log k$ states which query v_7, making a total of $O(k^2/\log k)$ states. ■

We can improve the constant in the above bound by arguing more carefully.

Theorem 18 *The Boolean $RR3(k)$ problem can be solved by a deterministic BP with $k^2/\log_2 k$ V-states, for sufficiently large k, where the BP has three layers as in Theorem 15: a V-layer followed by and f_3-layer followed by a V-layer.*

Proof: The proof is similar to that of the previous theorem. Now we divide the k possible values of v_6 into $k/(\log_2 k + 1)$ equivalence classes ("bins") of size $\log_2 k + 1$ each. Layer I remembers v_2, queries v_6 and v_7, and passes on v_2 and v_7 and the bin number of v_6 to the initial states of Layer II. There are $k^2/(\log_2 k + 1)$ states in Layer I.

Layer II evaluates $f_3(v_6, v_7)$ for each pair v_6, v_7 in the relevant bin, remembers the subset of pairs which make $v_2 + f_3(v_6, v_7) = 1$, and passes this subset to one of the initial states of Layer III.

In fact this initial state of Layer III only needs to know which subset of the $\log_2 k + 1$ possible values in the bin for v_6 makes $v_2 + f_3(v_6, v_7) = 1$. Hence there are

$$2^{\log_2 k+1} = 2k$$

states of Layer III. Each state queries v_6 to determine whether the actual value of v_6 is in the good subset, and outputs 0 or 1 accordingly.

The total number of V-states in Layers I and III is $k^2/(\log_2 k + 1) + 2k$, which is at most $k^2/\log_2 k$ for sufficiently large k. ■

Theorem 19 *Any deterministic BP with three layers (V-f_3-V) solving the Boolean $RR3(k)$ problem has more than $k^2/(2\log_2 k)$ V-states.*

Proof: The proof is similar to the proof of Theorem 15. Suppose that there are at most $k^2/(2\log_2 k)$ states in Layer I. Then there are at most $k^3/(2\log_2 k)$ initial states of Layer II, and hence there is a set of at least $2\log_2 k$ distinct triples $(v_2^i, v_6^i, v_7^i), 1 \leq i \leq r, r \geq 2\log_2 k$ of triples such that the computation on input any one of of these triples leads to the same initial state p of Layer II. As in the earlier proof, it is easy to see that if $i \neq j$ then the pairs (v_6^i, v_7^i) and (v_6^j, v_7^j) are distinct.

Let $Out_{f_3}(i)$ be the Boolean predicate on $\{1, \ldots, r\}$ which is true iff $v_2^i + f_3(v_6^i, v_7^i) = 1$; i.e. the input $(v_2^i, v_6^i, v_6^i, f_3)$ leads to the output TRUE. Note that there are $2^r \geq 2^{2\log_2 k} = k^2$ possible choices for the predicate Out_{f_3} as f_3 varies.

Claim: There must be a distinct initial state q of layer III for each of the $\geq k^2$ possibilities for Out_{f_3} as f_3 varies.

The proof of the Claim is as before, and the theorem follows. ■

3.2.2 The FVF case

The FVF case is much more difficult than the VFV case, and we give several different proofs. The second proof (Theorem 28) is the simplest, but the final proof (Theorem 38, in the next subsection) stands a better chance of generalizing to the VFVF case.

We will name the three layers F_1, V, F_2. The goal (as before) is to show that the total number of V-states must be at least k^2. We approach this problem by restricting the number of initial V-states. The initial V-states are those that have an edge from layer F_1 leading to them, and hence they are the only states that carry information from F_1.

Let F be the set of all functions $f_3 : [k] \times [k] \to [k]$.

In the following we fix a BP B with three layers F_1, V, F_2.

Let $Q = \{q_1, \ldots q_\ell\}$ be the set of initial V-states in B.

We assume $\ell < k^2$, since we are trying to prove the number of V-states is at most k^2.

Let $\varphi = \varphi_B : [k] \times F \to Q$ be defined by $\varphi(v_2, f_3) = q_i$ if an input (v_2, v_6, v_7, f_3) to B reaches the state q_i. (Note that v_6, v_7 are irrelevant.)

For each f_3, define $\varphi_{f_3}(v_2) = \varphi(v_2, f_3)$

Lemma 20 *For each f_3, the function $\varphi_{f_3} : [k] \to Q$ is injective.*

Proof: This is because for fixed f_3, v_6, v_7 the output $v_2 + f_3(v_6, v_7)$ varies with v_2. ■

It follows that there are at least k initial V-states, so $\ell \geq k$.

We now give a simple proof for the case in which there are only k initial V-states.

Theorem 21 *If there are exactly k initial V-states, then B has at least k^2 V-states.*

Proof: The proof is similar to the case of two layers V-F (Theorem 13). Recall $q_1, \ldots, q_k$ are the initial V-states. It suffices to show that there are at least k^3 initial F_2 states, since then there must be at least k^2 V-states to generate the k^3 edges leading to the initial F_2 states. We show this by showing that each distinct triple (q_i, v_6, v_7) leads to a distinct initial F_2 state. (Note that (q_i, v_6, v_7) determines the initial F_2 state.)

Suppose to the contrary that $(q_i, v_6, v_7) \neq (q_j, v_6', v_7')$ but both triples lead to the same initial F_2 state.

Case I: $q_i = q_j$.
Choose any f_3 such that $f_3(v_6, v_7) \neq f_3(v_6', v_7')$. Choose v_2 such that $\varphi_{f_3}(v_2) = q_i = q_j$ (see Lemma 20). Then the inputs (v_2, v_6, v_7, f_3) and (v_2, v_6', v_7', f_3) both lead to the same initial F_2 state but have different outputs, so B makes a mistake on one of them.

Case II: $q_i \neq q_j$.
Choose any f_3 so $f_3(v_6, v_7) = f_3(v_6', v_7')$. Choose v_2, v_2' so $\varphi_{f_3}(v_2) = q_i$ and $\varphi_{f_3}(v_2') = q_j$, so $v_2 \neq v_2'$. Then (v_2, v_6, v_7, f_3) and (v_2', v_6', v_7', f_3) lead to the same initial state of F_2 but have different outputs. ■

To handle the general F_1-V-F_2 case it suffices to show that if the set of initial V-states is $\{q_1, \ldots, q_\ell\}$ and $\ell < k^2$, then the number of initial F_2-states is at least k^3. The intuition is that the layer F_1 has no knowledge of v_6, v_7, and hence it cannot summarize useful information about f_3 using just k^2 output states. Thus some version of the proof for the V-F case should give us the required lower bound for the number of V-states.

The next two results will be used in the proof of Theorem 24 to get a lower bound on the number of initial F_2-states.

NOTE: The next result is useful, but then skip to Theorem 28.

Lemma 22 *If triples (q_i, v_6, v_7) and (q_j, v_6', v_7') lead to the same initial F_2-state then for all v_2, v_2', f_3, if $\varphi(v_2, f_3) = q_i$ and $\varphi(v_2', f_3) = q_j$ then*

$$v_2 + f_3(v_6, v_7) = v_2' + f_3(v_6', v_7')$$

Proof: Under the hypothesis of the lemma, both of the inputs (v_2, v_6, v_7, f_3) and (v_2', v_6', v_7', f_3) lead to the same F_2-state, and since they share the same f_3 they must have the same output. ■

SKIP TO THEOREM 28

Lemma 23 *Let A and B be finite sets, and $g : A \to B$. Suppose we are given any probability distribution over A. When elements a, a' of A are chosen independently according to the distribution let $\epsilon = Pr[g(a) = g(a')]$. Then $|B| \geq 1/\epsilon$.*

Proof:

$$\begin{aligned} Pr[g(a) = g(a')] &= \sum_{b \in B} Pr[g(a) = b] \cdot Pr[g(a') = b] && (4) \\ &= \sum_{b \in B} p_b^2 && (5) \end{aligned}$$

where $p_b = Pr[g(a) = b]$ when a is chosen according to the given probability distribution. Now $\{p_b : b \in B\}$ is a probability distribution on B, and $\sum_{b \in B} p_b^2$ is minimized when the p_b are all equal, i.e. when each $p_b = 1/|B|$. Thus

$$Pr[g(a) = g(a')] \geq |B|/|B|^2 = 1/|B|$$

■

Recall that $Q = \{q_1, \dots, q_\ell\}$ is the set of initial V-states, where $\ell \geq k$. For each state q_i let

$$W_i = \{f_3 \mid \exists v_2 \, \varphi(v_2, f_3) = q_i\} \tag{6}$$

Let F be the set of all possible functions f_3. By Lemma 20 each f_3 is mapped to k distinct states q_i, and so $\sum_i |W_i| = k|F|$, so

$$\text{the average size of } |W_i| \text{ is } k|F|/\ell \tag{7}$$

Similarly for intersections

$$\text{the average size of } |W_i \cap W_j| \text{ is } k^2|F|/\ell^2 \tag{8}$$

and hence by our assumption $\ell < k^2$ the averages intersection size $|W_i \cap W_j| \geq |F|/k^2$.

We are now ready to get a lower bound for the general f_3-V-f_3 case.

Theorem 24 *Any deterministic f_3-V-f_3 branching program solving $RR3(k)$ has at least $k^2(1 - o(1))$ V-states.*

Proof: As mentioned earlier, the intuition is that the initial F_1-layer is not very useful, because v_6, v_7 are not yet known. Hence useful information about which of the k^{k^2} functions f_3 is input cannot be summarized using just k^2 initial V-states, so the V layer must pretty much pass on the triple (v_2, v_6, v_7) to the final F_2 layer. In fact our proof shows that if an input $I = (v_2, v_6, v_7, f_3)$ is chosen uniformly at random, then the triple (v_2, v_6, v_7) can be determined with error probability $O(1/k^4)$ from the information of which initial F_2-state I reaches.

We refer to the notation described at the beginning of Subsection 3.2.2.

We apply Lemma 23 with A, B as follows.

$$\begin{aligned} A &= \{(v_2, v_6, v_7) \mid v_2, v_6, v_7 \in [k]\} \\ B &= \text{the set of all initial } F_2\text{-states} \end{aligned}$$

We use the uniform distribution over A.

Instead of defining a single function G, we define a family $\{G_{f_3^0} : f_3^0 \in F\}$ of functions, where

$$G_{f_3^0} : A \to B$$

is defined by $G_{f_3^0}(v_2, v_6, v_7)$ is the initial F_2-state reached on input (v_2, v_6, v_7, f_3^0). We will show that when f_3^0 is chosen uniformly at random from $F = \{f_3 : [k] \times [k] \to [k]\}$ and (v_2, v_6, v_7) and (w_2, w_6, w_7) are chosen independently then

$$\Pr[G_{f_3^0}(v_2, v_6, v_7) = G_{f_3^0}(w_2, w_6, w_7)] \leq 1/k^3 + O(1/k^4) \tag{9}$$

Thus there exists some particular f_3^0 for which (9) holds, so according to Lemma 23 there must be at least $k^3(1 - o(1))$ initial F_2-states, and the theorem follows.

Recall the definition (6) of W_i.

Lemma 25 *For fixed $v_2, w_2 \in [k]$, if f_3^0 is chosen uniformly at random from F and we set $q_i = \varphi(v_2, f_3^0)$ and $q_j = \varphi(w_2, f_3^0)$ then the probability that $|W_i \cap W_j| \leq |F|/k^8$ is at most $1/k^4$.*

Proof: For each fixed pair r, s, if $|W_r \cap W_s| \leq |F|/k^8$ then the probability that $f_3^0 \in W_r \cap W_s$ is at most $1/k^8$. Since there are at most k^2 states $q_1, \ldots, q_\ell$, there are at most k^4 choices for r, s, so the probability that f_3^0 is in $W_r \cap W_s$ for some small $W_r \cap W_s$ is at most $k^4/k^8 = 1/k^4$. ■

Define

$$U = \{(v_6, v_7) \mid v_6, v_7 \in [k]\}$$

We use the variables $\vec{x}, \vec{y}, \vec{v}, \vec{w}$ to range over U. We define the function

$$\psi : Q \times U \to B$$

by $\psi(q_i, \vec{x})$ is the initial F_2-state reached from state q_i when the inputs (v_6, v_7) are $\vec{x}$.

For $1 \leq i, j \leq \ell$ define

$$S_{ij} = \{(\vec{x}, \vec{y}) \mid \vec{x} \neq \vec{y} \text{ AND } \psi(q_i, \vec{x}) = \psi(q_j, \vec{y})\}$$

The next lemma will be applied when the set S is S_{ij}.

Lemma 26 *If S is a set of more than $2t(2t-1)$ ordered pairs of distinct elements then there is a subset S' of S with $t+1$ pairs such that each pair in S' has at least one element that is distinct from all elements in the other pairs of S'.*

Proof: Note that the pairs in S must involve more than $2t$ distinct elements. ■

Recall $F = \{f : [k] \times [k] \to [k]\}$.

Lemma 27 *If $|S_{ij}| \geq 2t(2t-1)$ then $|W_i \cap W_j| \leq |F|/k^t$.*

Proof:

Assume $|S_{ij}| \geq 2t(2t-1)$. Then by Lemma 26 there is a subset S'_{ij} of S_{ij} consisting of at least $t+1$ pairs $(\vec{x}, \vec{y})$ such that each pair has at least one element $\vec{x}$ or $\vec{y}$ that is distinct from all elements in the other pairs of S'_{ij}. By Lemma 22, if $(\vec{x}, \vec{y}) \in S'_{ij}$ then

$$v_2^{f_3} + f_3(\vec{x}) = w_2^{f_3} + f_3(\vec{y}) \qquad \text{for all } f_3 \in W_i \cap W_j \tag{10}$$

where

$$v_2^{f_3} = \varphi_{f_3}^{-1}(q_i) \qquad w_2^{f_3} = \varphi_{f_3}^{-1}(q_j) \tag{11}$$

We argue that (10) limits the number of f_3's in $W_i \cap W_j$ as follows. For each $f_3 \in W_i \cap W_j$, the first pair $(\vec{x}_1, \vec{y}_1)$ in S'_{ij} fixes the difference $(w_2^{f_3} - v_2^{f_3})$ to be $f_3(\vec{x}_1) - f_3(\vec{y}_1)$. Each of the other t pairs reduces the choices for f_3 by a factor of k by specifying $f_3(\vec{x})$ or $f_3(\vec{y})$ in terms of another value of f_3 and the difference $(w_2^{f_3} - v_2^{f_3})$. ■

Now suppose that $(v_2, \vec{v})$ and $(w_2, \vec{w})$ are chosen independently from A, and f_3^0 is chosen uniformly from F. Then the probability that $(v_2, \vec{v}) = (w_2, \vec{w})$ is exactly $1/k^3$. Thus it suffices to show that the conditional probability

$$\Pr[G_{f_3^0}(v_2, \vec{v}) = G_{f_3^0}(w_2, \vec{w}) \mid (v_2, \vec{v}) \neq (w_2, \vec{w})] = O(1/k^4) \tag{12}$$

Let $q_i = \varphi(v_2, f_3^0)$ and $q_j = \varphi(w_2, f_3^0)$. If $G_{f_3^0}(v_2, \vec{v}) = G_{f_3^0}(w_2, \vec{w})$ then we cannot have $v_2 \neq w_2$ but $\vec{v} = \vec{w}$, since this violates Lemma 22 (with $f_3 = f_3^0$). Thus by our condition $(v_2, \vec{v}) \neq (w_2, \vec{w})$ we may assume

$$\vec{v} \neq \vec{w}$$

By Lemma 25, in order to prove (12) we may assume $|W_i \cap W_j| > |F|/k^8$, so by Lemma 27 with $t = 8$ it follows that $|S_{ij}| < 2t(2t-1) = 240$. Hence there are fewer than 240 pairs of distinct vectors $(\vec{x}, \vec{y})$ such that $\psi(q_i, \vec{x}) = \psi(q_j, \vec{y})$, and the probability that randomly chosen $(\vec{v}, \vec{w})$ equals one of these pairs is at most $240/k^4$. Thus (12) follows. ■

SKIP TO HERE

The next result is an improved version of Theorem 24 due to James.

Theorem 28 *(James' Alternative to Theorem 24.) Any deterministic f_3-V-f_3 branching program solving $RR3(k)$ has at least $k^2(1 - 2/k)$ V-states.*

The statement of this theorem differs from that of Theorem 24 only by making the $o(1)$ term explicit, but the proofs are somewhat different.

Proof: As before, the intuition is that the initial F_1-layer is not useful. This time, we begin by finding a large class of functions f_3 which the F_1-layer does not distinguish at all: that is to say, when f_3 is retstricted to be from this class, then the initial V-state depends only on v_2.

Let B be any f_3-V-f_3 branching program solving $RR3(k)$, and assume B has less than $k^2 - 2k$ V-states.

Step 1: The F_1-layer does not distinguish a large class of functions $F_* \subseteq F$. As in the proof of the previous theorem, let $Q = \{q_1, \ldots, q_\ell\}$ be the set of initial V-states, with $\ell < k^2 - 2k$. Let Ψ be the set of functions $[k] \to Q$, and let $\varphi : F \to \Psi$ be defined by setting $\varphi(f_3)(v_2)$ to be the first V-layer state reached by B when given those v_2 and f_3 values in the input. (This is similar to the function $\varphi : [k] \times F \to Q$ used in the previous proof.) Since $|\Psi| < k^{2k}$ and $|F| = k^{k^2}$, we can find some $\psi_* \in \Psi$ such that $|\varphi^{-1}(\psi_*)| > k^{k^2-2k}$. Let $F_* = \varphi^{-1}(\psi_*)$, so $|F_*| > k^{k^2-2k}$.

Henceforth we concern ourselves only with inputs where $f_3 \in F_*$. Observe that in this case, the initial V-state reached by B is equal to $\varphi_*(v_2)$, and hence does not depend on f_3. B now effectively has two layers, called V and F_2.

Step 2: A useful property of F_*. Define the following relation $P \subseteq ([k]^2)^2$: $P(\vec{v}, \vec{w})$ holds iff for all $f, g \in F_*$, $f(\vec{v}) - f(\vec{w}) = g(\vec{v}) - g(\vec{w})$, where $-$ denotes subtraction modulo k. Observe that P is an equivalence relation.

In the next step we will see that the initial F_2-state determines both v_2 and the equivalence class of (v_6, v_7) under the relation P. But first, we will show that the number of equivalence classes is more than $k^2 - 2k$. As a warm-up, if $F_* = F$, then there are k^2 equivalence classes, since for any $\vec{v} \neq \vec{w} \in [k]^2$, there exist functions $f, g \in F$ such that $f(\vec{v}) - f(\vec{w}) = 0$ but $g(\vec{v}) - g(\vec{w}) = 1$.

Now, assume only that $|F_*| > k^{k^2-2k}$. Pick a representative from each equivalence class to form the set $E \subseteq [k]^2$. We will show that a function $f \in F_*$ is determined entirely by its values on E: it follows that $|E| > k^2 - 2k$.

Indeed, let $f, g \in F_*$ be any two functions such that $\forall \vec{v} \in E, f(\vec{v}) = g(\vec{v})$; we must show $f = g$. Consider any $\vec{w} \in [k]^2$, and let $\vec{v} \in E$ be the representative of its equivalence class. Then $f(\vec{w}) = f(\vec{v}) + (f(\vec{w}) - f(\vec{v})) = g(\vec{v}) + (g(\vec{w}) - g(\vec{v})) = g(\vec{w})$.

Step 3: There are at least $k^3 - 2k^2$ initial F_2-states. For $v_2 \in [k]$ and $\vec{v} = (v_6, v_7) \in E$, let $G(v_2, \vec{v}) \in C$ be the first F_2-state reached by B on input (v_2, v_6, v_7, f_3) (as long as $f_3 \in F_*$, f_3 does not matter). To show that there are $k^3 - 2k^2$ initial F_2-states, all that remains is to show that any two distinct pairs $(v_2, \vec{v}) \neq (w_2, \vec{w}) \in [k] \times E$ lead to distinct inital F_2-states.

Suppose $(v_2, \vec{v}) \neq (w_2, \vec{w})$ lead to the same initial F_2 state. If $\vec{v} = \vec{w}$, then B gives the same output in both cases, even though the outputs should be different, which is a contradiction.

The remaining case is that $\vec{v} \neq \vec{w}$. Then $\vec{v}$ and $\vec{w}$ are representatives of different equivalence classes under P, so there must exist some $f, g \in F_*$ such that $f(\vec{v}) - f(\vec{w}) \neq g(\vec{v}) - g(\vec{w})$. In particular, we may assume that $f(\vec{v}) - f(\vec{w}) \neq w_2 - v_2$. Rearranging the inequality gives $v_2 + f(\vec{v}) \neq w_2 + f(\vec{w})$, so B must distinguish the inputs $(v_2, \vec{v}_2, f)$ and $(w_2, \vec{w}_2, f)$, but it does not. This is a contradiction: so it is true that each pair $(v_2, \vec{v})$ leads to a distinct initial F_2-state, and so there are at least $k^3 - 2k^2$ of them.

As in the previous proof, this implies that there are at least $k^2 - 2k$ V-states. ■

We can improve the lower bound in the previous theorem from $k^2 - 2k$ to k^2 by arguing a little more carefully and assuming that the V-states are layered.

Definition 29 *A set of contiguous states in a branching program is* layered *if all syntactic paths from any initial state in the set to any final state in the set have the same finite length.*

Note that any BP B is equivalent to a layered BP B' where the number of states in B' is at most the square of the number of states in B. So for the purpose of showing a problem is not in log space, it suffices to consider layered BPs.

Note that any BP solving TEP that comes from a pebbling algorithm is layered.

Theorem 30 *(Improvement to Theorem 28) For any deterministic f_3-V-f_3 branching program solving $RR3(k)$ with layered V states, the number of V-states is at least k^2 if $k > 2$.*

Proof: We observe that the same counting argument in **Step 1** of the proof of Theorem 28 used to show that $|\varphi^{-1}(\psi_*)| > k^{k^2-2k}$ also shows

$$|F_*| > k^{k^2}/\ell^k = k^{k^2-k\log_k \ell} \tag{13}$$

Arguing as in **Step 2** we obtain that the number of equivalence classes $|E| > k^2 - k\log_k \ell$, and from **Step 3** we conclude that there must be at least $k^3 - k^2\log_k \ell$ initial F_2-states, and hence the number of V-states which have edges to the F_2 layer is at least $k^2 - k\log_k \ell$.

Since the V-states are layered, no initial V-state in the set $Q = \{q_1, \ldots, q_\ell\}$ has an edge to the final f_3 layer. Hence the total number of V-states is at least

$$h(\ell) = \ell + k^2 - k\log_k \ell \tag{14}$$

It is easy to see that the derivative $h'(\ell)$ is positive for $\ell \geq k > e$, and so $h(\ell)$ takes its minimum for $\ell \geq k$ at $\ell = k$. Hence for the layered case the number of V-states must be at least $h(k) = k + k^2 - k\log_k k = k^2$. ■

The next result follows easily from (14).

Corollary 31 *In the setting of Theorem 30, if the number ℓ of initial V-states is at least $k(2 + \log_k 3)$ then the total number of V-states is at least $k^2 + k$.*

The comment at the end of Section 4.2 points out that the optimal pebbling algorithm for solving $RR3(k)$ yields a BP with $k^2 + k$ V-states. The above proof can possibly be improved to give a lower bound closer to $k^2 + k$ by considering functions f_3 not in F_*.

Here we prove a lower bound of $k^2 + k$ states quite independent of the above proof, but it only works when the number of initial V states is small.

Theorem 32 *For any deterministic f_3-V-f_3 branching program solving $RR3(k)$ with layered V-states and at most $k + \sqrt{k}/3$ initial V-states, the total number of V-states is at least $k^2 + k$, for sufficiently large k.*

Proof: This proof does not use the function equivalence class argument used in the previous two proofs.

We start by stating the following definitions and facts, some of them from earlier in this subsection:

$$\begin{aligned}
F &= \{f : [k] \times [k] \to [k]\} && (15)\\
Q &= \{q_1, \ldots, q_{k+u}\} = \text{initial } V\text{-states } (u \text{ small}) && (16)\\
\varphi(v_2, f_3) &= q_i \qquad (\text{initial } V\text{-state from } (v_2, f_3)) && (17)\\
\psi(q_i, \vec{v}) &= p_j \qquad (\text{initial } F_2\text{-state from } (q_i, \vec{v})) && (18)\\
1 \le i \le k+u:\ P_i &= \{\psi(q_i, \vec{v}) \mid \vec{v} \in [k] \times [k]\} && (19)\\
P &= \bigcup_{i=1}^{k+u} P_i = \text{initial } F_2\text{-states} && (20)\\
1 \le i \le k+u:\ W_i &= \{f_3 \mid \exists v_2\ \varphi(v_2, f_3) = q_i\} && (21)\\
\sum_{i=1}^{k+u} |W_i| &= k|F| && (22)\\
Q[f] &= \{\varphi(v_2, f) \mid v_2 \in [k]\} && (23)\\
|Q[f]| &= k, \qquad \forall f \in F && (24)
\end{aligned}$$

Lemma 33 *If $f \in W_i$ and $\psi(q_i, \vec{v}) = \psi(q_i, \vec{w})$ then $f(\vec{v}) = f(\vec{w})$.*

Proof: The initial F_2-state cannot distinguish the outputs $v_2 + f(\vec{v})$ and $v_2 + f(\vec{w})$. ■

Corollary 34 *If $|P_i| < k^2$ then $|W_i| \le |F|/k$*

Proof: There are k^2 possible vectors $\vec{v}$, so by the Lemma if $|P_i| < k^2$ then there exist distinct $\vec{v}, \vec{w}$ so every f in W_i satisfies $f(\vec{v}) = f(\vec{w})$. ■

Corollary 35 *If $u \le k-1$ then there are at least k distinct values of i such that $|P_i| = k^2$.*

Proof: This follows easily from Corollary 34 and (22). ■

Lemma 36 *If $1 \le i < j \le k+u$ and $\psi(q_i, \vec{v}) = \psi(q_j, \vec{w})$ then for all f in $W_i \cap W_j$, $f(\vec{v}) \ne f(\vec{w})$.*

Proof: If $f \in W_i \cap W_j$ and $i \ne j$ then there exists distinct v_2, w_2 such that $\varphi(v_2, f) = q_i$ and $\varphi(w_2, f) = q_j$, but the initial F_2-state $\psi(q_i, \vec{v}) = \psi(q_j, \vec{w})$ cannot distinguish between the outputs $v_2 + f(\vec{v})$ and $w_2 + f(\vec{w})$, so $f(\vec{v}) - f(\vec{w}) = w_2 - v_2 \ne 0$. ■

The next result is immediate from Lemma 36.

Corollary 37 *If f_0 is a constant function, then for all $q_i, q_j \in Q[f_0]$, if $i \ne j$ then $P_i \cap P_j = \varnothing$.*

Note that if $u = 0$ then from Corollary 35 and Corollary 37 we conclude there must be k^3 initial F_2-states and hence at least k^2 non-initial V-states and hence a total of at least $k^2 + k$ V-states, so the Theorem follows in this case.

In the general case we assume $u \leq k-1$ and note that by Corollary 35 there must be at least k sets P_i of cardinality k^2. We divide the remaining u sets P_i into those with at least k^2-a elements and those with fewer than k^2-a elements, for some parameter a to be determined. By renumbering the sets, we may suppose that sets

$$P_1, P_2, \ldots, P_{k+t} \tag{25}$$

have cardinality at least k^2-a for some t with $0 \leq t \leq u$, and sets

$$P_{k+t+1}, \ldots P_{k+u} \tag{26}$$

have cardinality less than k^2-a.

Our plan is to find a function f such that using Lemma 33 we force $f \notin W_i$ when $k+t+1 \leq i \leq k+u$ (so $|P_i|$ is small), and force the k remaining large sets P_i with $q_i \in Q[f]$ to be pairwise disjoint, using Lemma 36.

Consider an undirected graph G with vertices $W = \{1, 2, \ldots k+t\}$ and edges

$$E = \{(i,j) \mid i \neq j \text{ and } P_i \cap P_j \neq \varnothing\}$$

Let M be a subset of the edges E which forms a maximal matching in G.

CLAIM 1: $|M| \leq u$.
For assume $|M| \geq u+1$. By Lemma 36 if $(i,j) \in M$ and f_0 is a constant function, then $f_0 \notin W_i \cap W_j$, so f_0 cannot be in W_i for $u+1$ distinct values of i. But there are only $k+u$ possible values of i and each function f is in exactly k sets W_i. This proves the CLAIM 1.

Since M is a maximal matching, every edge in G has at least one endpoint in common with an edge in M. We expand the set M of edges to form a set M' by adding, for each endpoint i of an edge in M, all edges with an endpoint i except that if there are more than $u+1$ such edges then we only add $u+1$ such edges.

Thus

$$|M'| \leq u + 2u(u+1) = h(u) \tag{27}$$

For each edge $e = (i,j) \in M'$ we know $P_i \cap P_j \neq \varnothing$, so there exists $\vec{v_e}, \vec{w_e}$ such that

$$\psi(q_i, \vec{v_e}) = \psi(q_j, \vec{w_e})$$

Let

$$F' = \{f \in F \mid f(\vec{v_e}) = f(\vec{w_e}), \forall e \in M'\} \tag{28}$$

CLAIM 2: If $f \in F'$ and q_i and q_j are distinct elements of $Q[f]$ then $P_i \cap P_j = \varnothing$.

This is because if $P_i \cap P_j \neq \varnothing$, then G has an edge $e = (i,j)$. But $e \notin M'$ by Lemma 36 and the definition of F'. By definition of M', it follows that one endpoint, say i of e is in the maximal matching M, and there are at least $u+1$ edges in M' which share the endpoint i. If $e' = (i,r)$ is any of these $u+1$ edges then again by Lemma 36 and the definition of F' it follows that $f \notin W_r$. But every f is in k sets W_m and there are only $k+u$ possibilities for m, so this is impossible. This proves CLAIM 2.

Now set the parameter $a = 2h(u)$ as in (27). We can ensure that $f \in F'$ by setting

$$f(\vec{v_e}) = f(\vec{w_e}) = 1, \qquad \forall e \in M'$$

This constrains f on at most a different arguments, and since each of the sets in (26) has at most $k^2 - a - 1$ different elements we can still keep f in F' and use Lemma 33 to set $f(\vec{v}) \neq f(\vec{w})$ for suitable $\vec{v}, \vec{w}$ and make sure $f \notin \{k + t + 1, \ldots k + u\}$. For such f we have $q_i \in Q[f]$ implies $|P_i| \geq k^2 - a$, so by CLAIM 2 there must be at least k pairwise disjoint sets P_i with $|P_i| \geq k^2 - a$. Then by Corollary 35 there are at least $k - u$ sets P_i of size k^2 in this group. Hence we have

$$|P| \geq (k - u)k^2 + u(k^2 - a) = k^3 - uk^2 + uk^2 - ua \geq k^3 - ua$$

By dividing by k we obtain a lower bound of $k^2 - ua/k$ on the number of non-initial V-states, making the total number of V states at least $k + u + k^2 - ua/k$. In order to be sure this lower bound meets the claimed lower bound in the statement of the theorem, we want

$$k + u + k^2 - ua/k \geq k^2 + k$$

or $u \geq ua/k$ or $a \leq k$, or $h(u) \leq k$. Thus it suffices to have $2u^2 + 3u \leq k$, or $u \leq \sqrt{k}/3$, for sufficiently large k. ■

3.2.3 Another FVF proof

Here we give a third lower bound proof (due to Phuong and Steve) for the FVF case. This proof is a little more complicated, but it seems more likely to generalize. The idea is to desribe the set of inputs that reach each state, and use properties of these sets to show that they cannot soon lead to ouput states unless these input sets are small.

Theorem 38 *Any deterministic f_3-V-f_3 branching program solving $RR3(k)$ has at least $k^2(1/2 - O(1/k))$ distinct V-states.*

Proof: Each state in the program is associated with the set of inputs that reach it. The sets that are associated with initial F_2 states form a disjoint partition of all inputs. Each such set must satisfy several constraints, and if there are two few V states (and hence two few initial F_2 states), these constraints will limit the size of the sets, and this will give a contradiction.

Let ℓ be the number of initial V states, and r be the number of initial F_2 states. We will show that either $\ell \geq k^2/2$, or $r \geq k^3/2$. We prove this by contradiction, so assume that both $\ell < k^2/2$ and $r < k^3/2$.

Let F denote the set of all functions $[k] \times [k] \to [k]$. Note that there are in total $k^3|F|$ inputs to the program. Let $q_1, q_2, \ldots, q_\ell$ be the initial V states and $p_1, p_2, \ldots, p_r$ be the initial F_2 states. Below we will use indices i, j, z: $1 \leq i \leq k$ (possible value of v_2), $1 \leq j \leq \ell$ (q_j ranges over all initial V states) and $1 \leq z \leq r$ (p_z ranges over all initial F_2 states). We will also use w for pairs (v_6, v_7) (so $w \in [k]^2$).

For each $1 \leq i \leq k$ let $G_{i,1}, G_{i,2}, \ldots, G_{i,\ell}$ be the disjoint partition of F: $G_{i,j}$ is the set of functions $f \in F$ such that following $v_2 = i$ and f we reach q_j.

Fix some j, $1 \leq j \leq \ell$. Then $G_{1,j} \cup G_{2,j} \cup \ldots \cup G_{k,j}$ is the set of all functions that reach q_j (via some possible value of v_2). Observe that $G_{1,j}$, $G_{2,j}$, ..., $G_{k,j}$ are pairwise disjoint, because if

$f \in G_{i,j} \cap G_{i',j}$ for $i \neq i'$, then on inputs (i, w, f) and (i', w, f) (for any w) the program has the same output, which is a contradiction. Let G_j denote the disjoint union

$$G_j = G_{1,j} \cup G_{2,j} \cup \ldots \cup G_{k,j}$$

For each pair of states q_j and p_z, let $W_{j,z}$ denote the set of tuples $w \in [k]^2$ that lead from q_j to p_z. Then $W_{j,1}$, $W_{j,2}$, ..., $W_{j,r}$ is a partition of $[k]^2$. Let $t_{j,z} = |W_{j,z}|$, then it follows that

$$t_{j,1} + t_{j,2} + \ldots + t_{j,r} = k^2 \tag{29}$$

It is helpful to write down the set of inputs that reach an initial F_2 state p_z. This consists of the disjoint union of (we will write $i \times W \times G$ for $\{i\} \times W \times G$):

$$\begin{array}{c}
1 \times W_{1,z} \times G_{1,1}, \;\; 2 \times W_{1,z} \times G_{2,1}, \;\; \ldots, \;\; k \times W_{1,z} \times G_{k,1} \\
1 \times W_{2,z} \times G_{1,2}, \;\; 2 \times W_{2,z} \times G_{2,2}, \;\; \ldots, \;\; k \times W_{2,z} \times G_{k,2} \\
\ldots \\
1 \times W_{\ell,z} \times G_{1,\ell}, \;\; 2 \times W_{\ell,z} \times G_{2,\ell}, \;\; \ldots, \;\; k \times W_{\ell,z} \times G_{k,\ell}
\end{array}$$

So the total number of inputs associated with p_z is

$$\sum_{j=1}^{\ell} t_{j,z} |G_j|$$

Summing up over all states p_z we have the total number of all possible inputs, so:

$$\sum_{z=1}^{r} \sum_{j=1}^{\ell} t_{j,z} |G_j| = k^3 |F| \tag{30}$$

We will prove an upper bound using $\ell, r, |F|$ on the LHS, and this will give the desired lower bounds on ℓ and r.

Notice that if some $W_{j,z}$ contains two different pairs w, w' then for all inputs of the form (v_2, w, f) and (v_2', w', f) that reach p_z (here v_2 and v_2' may not be distinct) the values of $f(w)$ and $f(w')$ are tied together by

$$v_2 + f(w) = v_2' + f(w')$$

because at p_z the program cannot distinguish between the two inputs. As a result, if some state q_j can afford to treat several w in the same way then the set of functions G_j cannot be too large. We have:

Lemma 39 *Let* $1 \leq j \leq \ell$ *and* $t = \max_{1 \leq z \leq r} t_{j,z}$. *Then*

$$|G_j| \leq \frac{|F|}{k^{t-1}}$$

Proof: Let p_z be a state such that $|W_{j,z}| = t_{j,z} = t$, and let $f \in G_j$. Choose i so $f \in G_{i,j}$. Then on all inputs (i, w, f) (for $w \in W_{j,z}$) the program has the same output. Therefore f is constant on $W_{j,z}$, i.e., $f(w_1) = f(w_2) = \ldots = f(w_t)$ where $\{w_1, w_2, \ldots, w_t\} = W_{j,z}$. In other words, the values of f on $t-1$ inputs $w_2, \ldots, w_t$ are determined by $f(w_1)$. So the conclusion follows. ■

We will show that the contribution of small G_j (i.e., $|G_j| \le |F|/k$) to the LHS of (30) is not significant, and therefore we can focus on the G_j such that $t_{j,z} \in \{0,1\}$ for all z. So let $S = \{j \ : \ \max_{1 \le z \le r} t_{j,z} \ge 2\}$ index the small sets G_j and $L = \{1, 2, \ldots, \ell\} - S$ index the large sets. By the lemma we have

$$|G_j| \le |F|/k \quad \text{for all } j \in S \tag{31}$$

The LHS of (30) is

$$\sum_{z=1}^{r} \sum_{j \in S} t_{j,z} |G_j| + \sum_{z=1}^{r} \sum_{j \in L} t_{j,z} |G_j|$$

The first sum can be upper bounded as follows:

$$\begin{aligned} \sum_{z=1}^{r} \sum_{j \in S} t_{j,z} |G_j| &= \sum_{j \in S} |G_j| \sum_{z=1}^{r} t_{j,z} \\ &= \sum_{j \in S} |G_j| k^2 \quad \text{by (29)} \\ &\le \sum_{j \in S} \frac{|F|}{k} k^2 \quad \text{by (31)} \\ &\le \ell k |F| \end{aligned} \tag{32}$$

Now we estimate the second sum. For $t \in \{0,1\}$ and a set G we use the notation tG to denote $\varnothing$ (the empty set) if $t = 0$ and G if $t = 1$. Then for each z $(1 \le z \le r)$ the inclusion-exclusion principle gives us:

$$|\bigcup_{j \in L} t_{j,z} G_j| \ge \sum_{j \in L} t_{j,z} |G_j| - \sum_{j < j' \in L} t_{j,z} t_{j',z} |G_j \cap G_{j'}|$$

Therefore

$$\begin{aligned} \sum_{j \in L} t_{j,z} |G_j| &\le |\bigcup_{j \in L} t_{j,z} G_j| + \sum_{j < j' \in L} t_{j,z} t_{j',z} |G_j \cap G_{j'}| \\ &\le |F| + \sum_{j < j' \in L} t_{j,z} t_{j',z} |G_j \cap G_{j'}| \end{aligned}$$

Hence,

$$\begin{aligned} \sum_{z=1}^{r} \sum_{j \in L} t_{j,z} |G_j| &\le r|F| + \sum_{z=1}^{r} \sum_{j < j' \in L} t_{j,z} t_{j',z} |G_j \cap G_{j'}| \\ &= r|F| + \sum_{j < j' \in L} |G_j \cap G_{j'}| \sum_{z=1}^{r} t_{j,z} t_{j',z} \end{aligned} \tag{33}$$

It remains to upper bound the sum on the RHS. We need the following two lemmas.

Lemma 40

$$\sum_{1\le j<j'\le \ell} |G_j \cap G_{j'}| = k(k-1)|F|/2$$

Proof: For each pair (j, j'), $|G_j \cap G_{j'}|$ is the number of functions $f \in F$ that are in both G_j and $G_{j'}$. Each function f appears exactly k sets among $G_1, G_2, \ldots, G_\ell$. This is because for each value i of v_2, the set F is partitioned into disjoint union

$$G_{i,1} \cup G_{i,2} \cup \ldots \cup G_{i,\ell}$$

and each G_j is the disjoint union

$$G_j = G_{1,j} \cup G_{2,j} \cup \ldots \cup G_{k,j}$$

So each f is counted exactly $k(k-1)/2$ times in the sum. As a result,

$$\sum_{1\le j<j'\le \ell} |G_j \cap G_{j'}| = \sum_{f\in F} k(k-1)/2 = k(k-1)|F|/2$$

■

The above lemma provides a good upper bound on

$$\sum_{j<j'\in L} |G_j \cap G_{j'}| \sum_{z=1}^{r} t_{j,z}t_{j',z}$$

provided that for all $j < j'$

$$\sum_{z=1}^{r} t_{j,z}t_{j',z}$$

is small ($\mathcal{O}(k)$). The next lemma is to deal with the pairs j, j' where this sum is large.

Notice that for each pair $q_j, q_{j'}$ (where $j < j' \in L$) the sum

$$\sum_{z=1}^{r} t_{j,z}t_{j',z}$$

is precisely the total number of states p_z such that there are paths from both $q_j, q_{j'}$ to p_z. The following lemma is proved in the same way as Lemma 39:

Lemma 41 *Suppose $j < j' \in L$. Let*

$$t = \sum_{z=1}^{r} t_{j,z}t_{j',z}$$

Then

$$|G_j \cap G_{j'}| \le \frac{|F|}{k^{t-1}} \tag{34}$$

Proof: By the above remark, there are precisely t states p_z such that there are paths from q_j and $q_{j'}$ to p_z. Without loss of generality let these states be $p_1, p_2, \ldots, p_t$. Then for each p_z ($1 \le z \le t$) there are values w_z, w'_z of (v_6, v_7) that lead q_j, $q_{j'}$ to p_z, respectively (i.e., $W_{j,z} = \{w_z\}$, $W_{j',z} = \{w'_z\}$). Observe that $w_1, w_2, \ldots, w_t$ are pairwise distinct (and similarly for $w'_1, w'_2, \ldots, w'_t$).

Suppose $f \in G_j \cap G_{j'}$. Then for distinct $i, i' \in [k]$ we have $f \in G_{i,j} \cap G_{i',j'}$. Thus for each z the program produces the same output on (i, w_z, f) and (i', w'_z, f). Thus (recall $+$ and $-$ refer to addition and subtraction mod k)

$$i + f(w_z) = i' + f(w'_z)$$

so

$$f(w_z) = i' - i + f(w'_z)$$

Since $i \neq i'$ it follows that $w_z \neq w'_z$. Now for each $y \in [k]$ let

$$H_y = \bigcup_{i,i' \in [k], i - i' = y} (G_{i,j} \cap G_{i',j'})$$

Then for every $f \in H_y$ we have

$$f(w_z) = y + f(w'_z)$$

In other words, the values of f on t distinct inputs $w_1, w_2, \ldots, w_t$ can be determined from the values on other inputs. Therefore

$$|H_y| \le |F|/k^t$$

It follows that

$$|G_j \cap G_{j'}| = \sum_{y \in [k]} |H_y| \ \le k|F|/k^t = |F|/k^{t-1}$$

■

For each $j < j' \in L$ let

$$T_{j,j'} = \sum_{z=1}^{r} t_{j,z} t_{j',z}$$

Note that $T_{j,j'} \le r$ for all j, j'. For the derivation below, we use Lemma 40 for the pairs (j, j') where $T_{j,j'} \le 5$ and Lemma 41 for the other pairs. We have

$$\begin{aligned}
\sum_{j<j' \in L} |G_j \cap G_{j'}| \sum_{z=1}^{r} t_{j,z} t_{j',z} &= \sum_{j<j' \in L} |G_j \cap G_{j'}| T_{j,j'} \\
&= \sum_{T_{j,j'} \le 5} |G_j \cap G_{j'}| + \sum_{T_{j,j'} \ge 6} |G_j \cap G_{j'}| T_{j,j'} \\
&\le 5 \sum |G_j \cap G_{j'}| + \binom{\ell}{2} r 2 |F|/k^5 \\
&= \mathcal{O}(k^2)|F| \ \text{ by Lemma 40 and because } \ell < k^2, r < k^3
\end{aligned}$$

Substituting this into (33) we have

$$\sum_{z=1}^{r}\sum_{j\in L} t_{j,z}|G_j| \le r|F| + \mathcal{O}(k^2)|F| \tag{35}$$

From (30), (32) and (35) we get

$$(\ell k + r + \mathcal{O}(k^2))|F| \ge k^3|F|$$

If s is the total number of V-states then $s \ge \ell$ and $sk \ge r$ so $2sk + O(k^2) \ge k^3$ and

$$s \ge k^2/2 - O(k)$$

from which the Lemma follows. ■

3.2.4 The FVFV case

Theorem 42 [1] *Any deterministic f_3-V-f_3-V branching program solving $RR3(k)$ has at least $k^2/5$ V-states, for sufficiently large k.*

Proof: Let B be a deterministic BP with layers F_1-V_1-F_2-V_2 solving $RR3(k)$. We begin as in **Step 1** in the proof of Theorem 28. Thus there is some set F_* of functions $f : [k] \times [k] \to [k]$ and some set $\{q_1, \ldots, q_k\}$ of initial V_1-states such that for every input (v_2, v_6, v_7, f) to B, if $f \in F_*$ then the input reaches state q_{v_2}.

So in effect we can ignore the initial layer F_1 and assume that the BP has only three layers V_1-F_2-V_2, provided we only consider input functions $f \in F_*$. The only thing we need to know about F_* is that it is big enough; namely

$$|F_*| > k^{k^2-2k} \tag{36}$$

Recall that in the proof of the V_1-F-V_2 case (Theorem 15) we argued that if there are fewer than k^3 initial F-states (which is implied by fewer than k^2 V_1-states) then two distinct input triples (v_2, v_6, v_7) and (v_2', v_6', v_7') would reach the same initial F-state, and this would necessitate at least k^2 initial V_2-states. We cannot make the same argument now, because F_* may not include the right functions. However we can still argue that if there are fewer than $k^3/5$ initial F_2-states then there must be more than k^2 initial V_2-states.

Let $P = \{p_1, \ldots, p_r\}$ be the set of initial F_2-states. Define $\psi : [k]^3 \to P$ by

$$\psi(i, \vec{v}) = p_j$$

where p_j is the initial F_2 state reached when the input $(v_2, v_6, v_7) = (i, \vec{v})$ (here the choice of $f \in F_*$ is irrelevant).

Let Q_2 be the set of initial V_2-states. For each $f \in F_*$ define $\rho_f : P \to Q_2$ by $\rho_f(p_i)$ is the initial V_2-state reached starting at state p_i when the input function is f.

[1] This theorem has been replaced by Theorem 45, and can be omitted, although it has an interesting argument involving uniform hypergraphs.

Lemma 43 *Let $P' \subseteq P$ be such that for every $\vec{v} \in [k]^2$ there is $i \in [k]$ such that $\psi(i, \vec{v}) \in P'$. Then $\rho_f \upharpoonright_{P'}$ (the restriction of ρ_f to P') uniquely determines $f \in F_*$.*

Proof: Let P' be as in the lemma, and let f be any member of F_*. Let $\vec{v}$ be any member of $[k]^2$, and let $i \in [k]$ be such that $p_j \in P'$, where $p_j = \psi(i, \vec{v})$. Then the initial V_2-state $q = \rho_f(p_j)$ together with $\vec{v}$ determines the output $v_2 + f(\vec{v})$ of B, and this together with $v_2 = i$ determines $f(\vec{v})$. ■

Claim: If $|P| \le k^3/5$ then there is $P' \subseteq P$ with $|P'| \le (139/300)k^2$ such that P' satisfies the conditions of Lemma 43.

The theorem follows from the Claim,[2] since if $|P| > k^3/5$ then there are more than $k^2/5$ V_2-states (since states have fanout at most k), so we may assume $|P| \le k^3/5$. Hence we may apply Lemma 43 with $|P'| \le (139/300)k^2$. If $m = |Q_2|$ then there are at most $m^{(139/300)k^2}$ choices for $\rho_f \upharpoonright_{P'}: P' \to Q_2$, so by (36)

$$m^{(139/300)k^2} \ge |F_*| \ge k^{k^2-2k}$$

Raising both sides to the power $300/(139k^2)$ we obtain $|Q_2| = m > k^2$ for sufficiently large k, giving us the required lower bound.

The Claim follows from the next lemma, where $W = [k]^2$, P is the set of initial F_2-states, and each node $\vec{v} \in W$ is connected to the k nodes $\psi(i, \vec{v}), 1 \le i \le k$. (Note that these k states must be distinct, since the output $v_2 + f(\vec{v})$ with $i = v_2$ depends on v_2.)

Lemma 44 [3] *Let G be a bipartite graph with left vertex set W satisfying $|W| = k^2$ and right vertex set P satisfying $|P| \le k^3/5$, and suppose that each vertex in W has degree k. Then there is a subset $P' \subseteq P$ with $|P'| \le (139/300)k^2$ which covers W, in the sense that every node in W is adjacent to some node in P'.*

Proof: For each subset $P' \subseteq P$ let

$$W(P') = \{w \in W \mid w \text{ is adjacent to some element of } P'\}$$

Our goal is to find some P' with $|P'| \le (139/300)k^2$ and $W(P') = W$. We will form P' as the disjoint union of sets P_5, P_4, P_3, P_2, P_1 using a greedy algorithm, and define

$$t_i = |P_i|,\ s_i = |W(P_i) \setminus \bigcup_{j=i+1}^{5} W(P_j)|,\ 1 \le i \le 5$$

Motivated by the fact that the average degree of elements of P is at least 5, we start by building P_5, initializing P_5 to be $\varnothing$ and successively adding elements to P_5 such that each element added increases $|W(P_5)|$ by at least 5, until no more such elements can be added. Since $s_5 = |W(P_5)|$ we have

$$s_5 \ge 5t_5$$

[2]For this all we need know about 139/300 is that it is less than 1/2.

[3]Dustin says this is really about vertex covers for k-uniform hypergraphs. He worked out the following generalization: If H is a k-uniform hypergraph with m edges and average degree at most d, then H has a vertex cover of size less than mH_d/d, where H_d is the dth harmonic number. Lemma 44 is obtained by setting $m = k^2$ and $d = 5$.

By definition no node in P_5 is adjacent to any node in $W \setminus W(P_5)$, and since no more nodes can be added to P_5 it follows that any node in $P \setminus P_5$ can be adjacent to at most 4 nodes in $W \setminus W(P_5)$. Since each node in $W \setminus W(P_5)$ has degree k we have

$$4(k^3/5 - t_5) \geq k(k^2 - s_5)$$

so

$$s_5 > k^2/5$$

Now we augment P_5 by initializing $P_4 = \varnothing$, and successively adding elements of $P \setminus P_5$ to P_4 so that each element added increases $W(P_5) \cup W(P_4)$ by at least 4, until no more such elements can be added. Arguing as before we have

$$s_4 \geq 4t_4$$

and

$$3(k^3/5 - t_4 - t_5) \geq k(k^2 - s_4 - s_5)$$

so

$$s_4 + s_5 > (2/5)k^2$$

We define P_3 and P_2 similarly, so

$$\begin{aligned} s_3 &\geq 3t_3 \\ s_2 &\geq 2t_2 \end{aligned}$$

and

$$\begin{aligned} 2(k^3/5 - t_3 - t_4 - t_5) &\geq k(k^2 - s_3 - s_4 - s_5) \\ k^3/5 - t_2 - t_3 - t_4 - t_5 &\geq k(k^2 - s_2 - s_3 - s_4 - s_5) \end{aligned}$$

so

$$\begin{aligned} s_3 + s_4 + s_5 &> (3/5)k^2 \\ s_2 + s_3 + s_4 + s_5 &> (4/5)k^2 \end{aligned}$$

Finally we choose P_1 to cover one by one the $k^2 - s_2 - s_3 - s_4 - s_5$ remaining nodes of W, so

$$t_1 = s_1 = k^2 - s_2 - s_3 - s_4 - s_5$$

Setting $P' = P_5 \cup P_4 \cup P_3 \cup P_2 \cup P_1$ we have from the above inequalities

$$\begin{aligned} |P'| &= t_5 + t_4 + t_3 + t_2 + t_1 \\ &\leq s_5/5 + s_4/4 + s_3/3 + s_2/2 + (k^2 - s_5 - s_4 - s_3 - s_2) \\ &= k^2 - (4/5)s_5 - (3/4)s_4 - (2/3)s_3 - (1/2)s_2 \\ &= k^2 - (1/2)(s_5 + s_4 + s_3 + s_2) - (1/6)(s_5 + s_4 + s_3) - (1/12)(s_5 + s_4) - (1/20)s_5 \\ &< k^2 - (1/2)(4/5)k^2 - (1/6)(3/5)k^2 - (1/12)(2/5)k^2 - (1/20)(1/5)k^2 \\ &= (139/300)k^2 \end{aligned}$$

■

■

Now we present an improved version of Theorem 42.

Theorem 45 *Any deterministic f_3-V-f_3-V branching program solving $RR3(k)$ has more than $k^2/3$ V-states, for sufficiently large k.*

Proof: Let B be a deterministic BP with layers F_1-V_1-F_2-V_2 solving $RR3(k)$. We begin as in **Step 1** in the proof of Theorem 28. Thus there is some set F_* of functions $f : [k] \times [k] \rightarrow [k]$ and some set $\{q_1, \ldots, q_k\}$ of initial V_1-states such that for every input (v_2, v_6, v_7, f) to B, if $f \in F_*$ then the input reaches state q_{v_2}.

So in effect we can ignore the initial layer F_1 and assume that the BP has only three layers V_1-F_2-V_2, provided we only consider input functions $f \in F_*$. The only thing we need to know about F_* is that it is big enough; namely

$$|F_*| > k^{k^2-2k} \tag{37}$$

Recall that in the proof of the V_1-F-V_2 case (Theorem 15) we argued that if there are fewer than k^3 initial F-states (which is implied by fewer than k^2 V_1-states) then two distinct input triples (v_2, v_6, v_7) and (v_2', v_6', v_7') would reach the same initial F-state, and this would necessitate at least k^2 initial V_2-states. We cannot make the same argument now, because F_* may not include the right functions. However we can still argue that if there are fewer than $k^3/3$ initial F_2-states then there must be more than k^2 initial V_2-states.

Let $P = \{p_1, \ldots, p_r\}$ be the set of initial F_2-states. Assume to get a contradiction that

$$r = |P| \leq k^3/3 \tag{38}$$

Define $\psi : [k]^3 \rightarrow P$ by

$$\psi(v, \vec{w}) = p_i$$

where p_i is the initial F_2 state reached when the input $(v_2, v_6, v_7) = (v, \vec{w})$ (here the choice of $f \in F_*$ is irrelevant).

Notice that if $(v, \vec{w})$ and $(v', \vec{w}')$ are distinct elements of $\psi^{-1}(p_i)$ then $\vec{w} \neq \vec{w}'$, because for $v \neq v'$ the two outputs $v + f(\vec{w})$ and $v' + f(\vec{w})$ are different but cannot be distinguished by any query.

For $1 \leq i \leq r$ we order the elements $(v, \vec{w})$ of $\psi^{-1}(p_i)$ by the lexicographic ordering of the $\vec{w}$ component to form the ordered sets

$$\vec{x}_i = \langle (v_i^1, \vec{w}_i^1), \ldots, (v_i^{t_i}, \vec{w}_i^{t_i}) \rangle, \ 1 \leq i \leq r$$

where $t_i = |\psi^{-1}(p_i)|$. Notice that by (38) the average value of t_i is at least 3.

We define the unordered set W_i to consist of the $\vec{w}$ components of $\vec{x}_i$. Thus

$$W_i = \{\vec{w}_i^1, \ldots, \vec{w}_i^{t_i}\} \tag{39}$$

As observed above, the $\vec{w}$ elements are distinct, so

$$|W_i| = t_i, \ 1 \leq i \leq r$$

For $f \in F_*$ and $1 \leq i \leq r$ define $\vec{a}_i^f \in [k]^{t_i}$ to be the ordered set of outputs determined by $\vec{x}_i$ and f. Thus

$$\vec{a}_i^f = \langle v_i^1 + f(\vec{w}_i^1), \dots v_i^{t_i} + f(\vec{w}_i^{t_i})\rangle$$

Let Q_2 be the set of initial V_2-states. We may as well assume

$$|Q_2| \leq k^2 \tag{40}$$

Define $\rho : P \times F_* \to Q_2$ by $\rho(p_i, f)$ is the initial V_2-state reached starting at state p_i when the input function is $f \in F_*$.

Lemma 46 *For $1 \leq i \leq r$ and $f, g \in F_*$, if $\rho(p_i, f) = \rho(p_i, g)$ then $\vec{a}_i^f = \vec{a}_i^g$.*

Proof: The V_2 states determine the output, but no V_2-query can distinguish f from g. ■

For $1 \leq i \leq r$ let

$$S_i = \{\vec{a}_i^f \mid f \in F_*\} \tag{41}$$

By (40) and Lemma 46 we have

$$|S_i| \leq k^2, \ 1 \leq i \leq r \tag{42}$$

Notice that if in the definition (41) we had allowed f to range over the set F of all functions $f : [k]^2 \to [k]$ instead of over F_*, then $|S_i| = k^{t_i}$. Since the average value of t_i is at least 3, equation (42) puts a considerable contraint on the cardinality of F_*. We use this idea to derive a contradiction from (37) and our assumption (38).

We will give an algorithm to define a sequence $p_{i_1}, p_{i_2}, \dots, p_{i_d}$ of states in P so that when $W'_1, \dots, W'_d$ are defined recursively by (see (39))

$$W'_1 = W_{i_1} \tag{43}$$

$$W'_{j+1} = W_{i_{j+1}} \setminus \bigcup_{\ell=1}^{j} W'_\ell \tag{44}$$

then the sets $W'_1, \dots, W'_d$ are pairwise disjoint and each has at least 3 elements.

Let

$$u_j = |W'_j| \geq 3, \ 1 \leq j \leq d$$

For $f \in F$ the number of choices for $f \upharpoonright W'_j$ (the restriction of f to W'_j) is k^{u_j}, but by (42) the number of choices for $f \upharpoonright W'_j$ when $f \in F_*$ is at most k^2. Since the sets W'_j are pairwise disjoint, each j cuts down the ratio of choices for $f \in F_*$ to choices of $f \in F$ by a factor of at least $k^{u_j - 2}$. Since by (37) we know $|F|/|F_*| < k^{2k}$, to reach a contradiction it suffices to have

$$\prod_{j=1}^{d} k^{u_j - 2} \geq k^{2k}$$

which is equivalent to

$$\sum_{j=1}^{d} u_j \geq 2k + 2d \tag{45}$$

Here is the algorithm to choose the sequence $p_{i_1}, p_{i_2}, \ldots, p_{i_d}$ of states in P. We use the abbreviation

$$Y_i = W_i \setminus \bigcup_{\ell<j} W_{i_\ell}, \qquad 1 \leq i \leq r$$

```
$j \leftarrow 1$
        while $j \leq 2k$ and $\max\{|Y_i| : 1 \leq i \leq r\} \geq 3$
        choose $i_j$ so $|Y_{i_j}| \geq 3$
$j \leftarrow j + 1$
end while
$d \leftarrow j - 1$
```

Notice that the resulting sets $W'_1, \ldots, W'_d$ are the sets $Y_{i_1}, \ldots, Y_{i_d}$, and hence each has at least 3 elements (and the sets are pairwise disjoint) as required. Also

$$d \leq 2k \tag{46}$$

To prove the requirement (45) there are two cases.

If the while loop halts because $j > 2k$, then $d = 2k$, and since each $u_i \geq 3$ it follows that $\sum_{j=1}^{d} u_j \geq 3d = 6k = 2k + 2d$.

Otherwise the while loop halts because $|Y_i| \leq 2$ for $1 \leq i \leq r$. Observe that each $\vec{w} \in [k]^2$ occurs in exactly k different sets W_i (once for each choice of $v_2 \in [k]$). Thus if each $|Y_i| \leq 2$, it must be that the ratio of the number of occurrences of all remaining vectors $\vec{w}$ in $W_1, \ldots, W_r$ after removing all occurrences in $\bigcup_{\ell=1}^{d} W'_\ell$, to $r \leq k^3/3$, is at most 2. Thus

$$\frac{k^3 - k\sum u_j}{k^3/3} \leq 2$$

so by (46) we have $\sum u_j \geq k^2/3 > 2k + 2d$ for sufficiently large k, as required.

■

3.2.5 The VFVF case

At this point we have been unable to prove an interesting lower bound for the VFVF case in solving $RR3(k)$ (Definition 7). Here we summarize the work on this by Steve and Phuong in the summer of 2010. This is based on generalizing the proof of Theorem 38.

Each input to our branching program B has the form (v_2, v_6, v_7, f) where $f = f_3 \in F$ where

$$F = \{f : [k]^2 \rightarrow [k]\}$$

We assume each V-state can query any property of v_6, v_7 (as long as the fanout is at most k) so we abbreviate v_6, v_7 by w. Thus each input to B has the form

$$(v, w, f) = (v_2, v_6, v_7, f_3)$$

The corresponding output of B is $v + f(w)$. Note that there are exactly $k^3|F|$ different inputs.

The states of B are divided sequentially into the four layers

$$V_1, F_1, V_2, F_2$$

where the states of V_1 and V_2 query w and have fanout k, and the states of F_1 and F_2 query f and have arbitrary fanout. (See Theorem 48 to see that we can get a lower bound in this general setting for the height 2 case.)

Layer V_1 has k inputs, one for each value of $v = v_2$.
Layer F_1 has m initial states $P^1 = \{p_1^1, \ldots, p_m^1\}$.
Layer V_2 has ℓ initial states $Q^2 = \{q_1, \ldots, q_\ell\}$.
Layer F_2 has r initial states $P^2 = \{p_1^2, \ldots, p_r^2\}$.

Layer V_1 computes a transformation $\varphi_{V_1} : [k^3] \to P^1$.
Layer F_1 computes a transformation $\varphi_{F_1} : P^1 \times F \to Q^2$.
Layer V_2 computes a transformation $\varphi_{V_2} : Q^2 \times [k]^2 \to P^2$.
Layer F_2 computes a transformation $\varphi_{F_2} : P^2 \times F \to [k]$.

For $1 \leq i \leq m$ let $U^i = \varphi_{V_1}^{-1}(p_i^1)$ be the set of triples (v, w) reaching state p_i^1. Thus the sets $U_1, \ldots, U_m$ form a partition of $[k]^3$. We write

$$U^i = \{(v_1^i, w_1^i), \ldots, (v_{u_i}^i, w_{u_i}^i)\}$$

where $u_i = |U^i|$.

FACT: For $1 \leq i \leq m$, the elements of $\{w_1^i, \ldots, w_{u_i}^i\}$ are all distinct.

Otherwise there would be distinct values v, v' of v_2 such that for some w, both of the triples (v, w) and (v', w) occur in U^i. But then for any f, the inputs (v, w, f) and (v', w, f) would reach the same output state of B even though $v + f(w) \neq v' + f(w)$.

For each state s in B we let I_s denote the set of inputs reaching state s. Thus

$$I_{p_i^1} = U^i \times F$$

For $1 \leq i \leq m$ and $1 \leq j \leq \ell$ let

$$G_j^i = \{f : \varphi_{F_1}(p_i^1, f) = q_j\}$$

Note that for each i, the sets $\{G_1^i, \ldots, G_\ell^i\}$ form a partition of F. For $1 \leq j \leq \ell$ we have

$$I_{q_j} = \bigcup_{i=1}^{m} U^i \times G_j^i \text{ (disjoint union)}$$

For $1 \leq j \leq \ell, 1 \leq z \leq r$ let

$$W_{j,z} = \{w : \varphi_{V_2}(q_j, w) = p^2_z\}$$

For $1 \leq z \leq r$ we have

$$I_{p^2_z} = \bigcup_{j=1}^{\ell} \bigcup_{i=1}^{m} \left(U^i \cap ([k] \times W^{j,z})\right) \times G^i_j$$

Thus $I_{p^2_z}$ is the disjoint union over all pairs (p^1_i, q_j) of inputs which follow the path (p^1_i, q_j) to p^2_z.

Let us briefly give an intuitive argument. We are trying to prove a lower bound of $\Omega(k^2)$ on the total number of V-states in B. So lets assume otherwise; say there are at most $k^2/10$ V-states. Then on average $|U^i| \geq 10$, so U_i has 10 distinct values of w, and the restriction of f to these 10 values has k^{10} possibilities. But there are only $\ell \leq k^2/10$ initial V_2-states, so on average a state q_j receiving the input set $U^i \times G^i_j$ cannot know much about the value of a function in G^i_j on a w in U^i. For each such w^i_a, q_j sends the entire input set $w^i_a \times G^i_j$ to some initial F_2-state p^2_z. Since there are only $r \leq k^3/10$ such states, on average each p^2_z receives at least $10|F|$ of the $k^3|F|$ possible inputs, so that makes an average of 10 possible w values for each function f it receives. Yet p^2_z must determine the output $v + f(w)$ based solely on f.

The proof of the next result shows that in order to get a lower bound of $\Omega(k^3)$ on r (and hence a lower bound of $\Omega(k^2)$ on the number of V-states in B), we may as well assume that the number of edges in B reaching each state p^2_z is $\omega(\sqrt{k})$, since any state p^2_z with fan-in $O(\sqrt{k})$ can handle only $O(|F|)$ distinct inputs.

Theorem 47 *Suppose that the number of input sets $(v^i_a, w^i_a) \times G^i_j$ reaching any fixed state p^2_z is $O(\sqrt{k})$. Then $r = \Omega(k^3)$. (Here $r = |P^2|$ is the number of initial states in the F_2 layer.)*

Proof: Suppose that the input sets reaching p^2_z are $(x_1 \times G_1), \ldots, (x_n \times G_n)$, where $x_1, \ldots, x_n$ are distinct triples in $[k]^3$, and $G_1, \ldots, G_n$ are subsets of F. Note that for any $i < j$, any f in $G_i \cap G_j$ must give the same output for both the triples x_i and x_j, and hence

$$|G_i \cap G_j| \leq |F|/k$$

Hence we can bound the number of inputs reaching p^2_z using the inclusion-exclusion principle as follows:

$$\begin{aligned} |I_{p^2_z}| = \sum_{i=1}^{n} |G_i| &\leq |\bigcup_{i=1}^{n} G_i| + \sum_{i<j} |G_i \cap G_j| \\ &\leq |F| + \binom{n}{2}|F|/k \\ &= O(|F|) \text{ assuming } n = O(\sqrt{k}) \end{aligned}$$

Since the total number of inputs is $k^3|F|$, it follows that the number of states p^2_z is $r = \Omega(k^3)$. ■

Now we will try using the method in the proof of Theorem 38 to get a lower bound of $\Omega(k^2)$ on the total number of V-states in B.

Let $t_{ijz} = |U^i \cap ([k] \times W_{j,z})|$ = the number of triples sent to state p^2_z via states p^1_i and q_j. As in Lemma 39 it is easy to show

$$|G^i_j| \le |F|/k^{t_{ijz}-1}$$

As before, when $\max_{1 \le z \le r} \ge 4$ (or some small number like 4), $|G^i_j|$ is small enough to ignore. For simplicity, let us assume that

$$t_{ijz} \le 1$$

By the inclusion/exclusion principle we have

$$\begin{aligned} |I_{p^2_z}| &= \sum_{i,j} t_{ijz}|G^i_j| \\ &\le |\bigcup_{i,j} t_{ijz}G^i_j| + \sum_{i<i',j,j'} |G^i_j \cap G^{i'}_{j'}| t_{ijz}t_{i'j'z} \end{aligned}$$

Using an upper bound of $|F|$ on the first term in the second line, and the fact that there are a total of $k^3|F|$ inputs, it follows that

$$k^3|F| = \sum_{z=1}^{r} |I_{p^2_z}| \le r|F| + \sum_{i<i',j,j'} (|G^i_j \cap G^{i'}_{j'}| \cdot \sum_z t_{ijz}t_{i'j'z}) \tag{47}$$

Let $S_{ij} = \{z : t_{ijz} = 1\}$. Then

$$\sum_z t_{ijz}t_{i'j'z} = |S_{ij} \cap S_{i'j'}|$$

Note that $\sum_{i<i',j,j'} |G^i_j \cap G^{i'}_{j'}| = \binom{m}{2}|F|$, because each f contributes once to the sum for each pair $i < i'$. Since $|G^i_j \cap G^{i'}_{j'}|$ is small when $|S_{ij} \cap S_{i'j'}| \ge c$ for a suitable constant c, we obtain

$$k^3 \le r + \binom{m}{2}c$$

However this is not good enough to get a good lower bound on $r + m$. In order to show this, we must show that $S_{ij} \cap S_{i'j'}$ is empty for most tuples i, j, i', j'.

3.2.6 The 'simple' (height 2) case

By the *simple* case we mean the Tree Evaluation Problem for binary trees of height 2. There is an easy proof in [CMW+10] showing that any k-way BP solving this must have at least k states that query the leaves v_2, v_3, even when the root function is fixed to be addition mod k. However this proof sheds no light on our attempts to prove lower bounds in this section (the sequential height 4 case) since the problem becomes trivial when f_3 is addition mod k. In fact our approach in this section assumes that the states in the BP are divided into layers such as VFVF. This amounts to trying to prove lower bounds for a much stronger kind of BP, in which the V layers allow an arbitrary transformation depending on v_6, v_7 with some fixed (but large) number of inputs and outputs, and similarly for the F layers.

Hence it is worth studying proofs for this simple height 2 case assuming that the BP is divided into V and F layers which have general access to the inputs (v_2, v_3) and f_1, respectively. In fact a simple counting argument in the style of Neciporuk gives us the desired lower bound for this case, and more generally. Unfortunately this counting argument does not work for solving the problem $RR3(k)$, although perhaps it might prove helpful in some way.

Theorem 48 *Let B be a k-way branching program solving the Tree Evaluation Problem for binary trees of height 2, and suppose that we allow any V-state in B to make an arbitrary query concerning the pair of leaf values (v_2, v_3) (as long as there are at most k edges leaving the state) and we allow any F-state to make an arbitrary query concerning the table of values defining the root function f_1 (here we can allow any number of out-edges from the F-state). Then B has at least k different V-states.*

Proof: Because we are interested in the layered case in this subsection we will assume that the states of B are divided into alternate V and F layers $V_1, F_1, V_2, F_2, \ldots, V_t, F_t$. However a standard Neciporuk-style argument proves the theorem even without this assumption.

Suppose layer V_i has ℓ_i V-states, $1 \leq i \leq t$. By the fanout assumption it follows that each F-layer F_i has at most $k\ell_i$ inputs from the preceeding V-layer V_i. Thus for each possible root function $f : [k]^2 \to [k]$ we can describe the effect of layer F_i as a map from $k\ell_i$ inputs to at most ℓ_{i+1} inputs of layer V_{i+1}, and the effect of the final layer F_t as a map from $k\ell_t$ inputs to the k outputs of the branching program B. Since we are trying to prove that there are at least a total of k V-states we may assume $\ell_i \leq k$. Hence for $1 \leq i \leq t$ and each function f we can describe the effect of layer F_i by a map

$$\varphi_i^f : [k\ell_i] \to [k]$$

There are at most $k^{k\ell_i}$ choices for φ_i^f, and for each distinct $f : [k]^2 \to [k]$ the sequence $\varphi_1^f, \ldots, \varphi_t^f$ must be distinct (since any two distinct root functions f will give a different output for some pair v_2, v_3). Hence

$$\prod_{i=1}^{t} k^{k\ell_i} \geq k^{k^2}$$

so $\sum_{i=1}^{t} \ell_i \geq k$, as required. ■

Remark: The above proof shows a stronger result, namely that even with no limit on the fan-out of the V-states, the total number of initial F-states is at least k^2.

3.2.7 Lower bound for semi-thrifty programs for the simple case

We have not been able to prove the lower bound for the VFVF case even when the branching program satisfies the condition that the sets G_j^i are determined by querying only $f(w)$ where w appears in U_i. Here we will give a simple argument showing lower bound for VFVF semi-thrifty branching programs that solves the simple (height 2) case.

Theorem 49 *Let B be a VFVF semi-thrifty branching program that solves the Tree Evaluation Problem for binary trees of height 2. Then either it has k^2 many initial V-states, or it has k^2 many initial F-states.*

Proof: As usual, we denote the initial states in the first F layer by $p^1_1, p^1_2, \ldots, p^1_m$; the initial states in the second V layer by $q_1, q_2, \ldots, q_\ell$; and the initial states in the last F layer by $p^2_1, p^2_2, \ldots, p^2_r$. Suppose that $\ell < k^2$, we will show that $m + r \geq k^2$.

Let U_i denote the set of w that reach p^1_i. So $U_1, U_2, \ldots, U_m$ is a partition of $[k]^2$. In particular, U_i and $U_{i'}$ are disjoint for any $i \neq i'$.

We say that (w, G^i_j) "meets" $(w', G^{i'}_{j'})$ if for some state p^2_z, w labels an edge from q_j to p^2_j and w' labels an edge from $q_{j'}$ to p^2_z.

Claim: Suppose that (w, G^i_j) meets $(w', G^{i'}_{j'})$. Then G^i_j is single-valued on w, that is, for any two functions $f, f' \in G^i_j$, $f(w) = f'(w)$.

Proof: Suppose for a contradiction that there are functions $f, f' \in G^i_j$ such that $f(w) \neq f'(w)$. We view f and f' as function on U_i. Because U_i is disjoint from $U_{i'}$, and $G^{i'}_{j'}$ is defined based on values of functions on $U_{i'}$, there are extensions $\widehat{f}$ and $\widehat{f'}$ of f, f' such that $\widehat{f}|_{U_{i'}} = \widehat{f'}|_{U_{i'}}$, and that both $\widehat{f}$ and $\widehat{f'}$ belong to $G^{i'}_{j'}$. As a result, both $\widehat{f}$ and $\widehat{f'}$ belong to the intersection

$$G^i_j \cap G^{i'}_{j'}$$

Therefore

$$\widehat{f}(w) = \widehat{f}(w'), \qquad \widehat{f'}(w) = \widehat{f'}(w')$$

This is a contradiction, because $\widehat{f}(w) = f(w) \neq f'(w) = \widehat{f'}(w)$, while $\widehat{f}(w') = \widehat{f'}(w')$. ■

It follows from the claim that if there are distinct $w_i, w'_i \in U_i$ such that (w_i, G^i_j) and (w'_i, G^i_j) each meets at least some other pair $(w', G^{i'}_{j'})$, then $|G^i_j| \leq |F|/k^2$. This is because in this case G^i_j must be single-valued on both w_i and w'_i.

Fix i, $1 \leq i \leq m$. Because $\ell < k^2$, there must be some j so that $|G^i_j| > |F|/k^2$. By the above observation, there is at most one value $w_i \in U_i$ such that the pair (w_i, G^i_j) meets some other pair $(w', G^{i'}_{j'})$. Consequently, for each of the other $|U_i| - 1$ many $w \in U_i$ the pair (w, G^i_j) takes up on distinct state p^2_z. As a result, the total number of initial F_2-states is at least

$$\sum_{i=1}^{m} (|U_i| - 1) = k^2 - m$$

In other words, $r + m \geq k^2$. ■

4 Optimizing Branching Programs from Pebbling

In his MSc research paper Dustin shows that when the standard black pebbling algorithm for the binary tree of height h is implemented by a k-way BP, the resulting BP has exactly $(k+1)^h - k$ states (not counting output states). Although we know that the standard pebbling algorithm uses the least possible number of pebbles, it is not immediately clear that its implementation by a BP uses the least possible number of states (among all pebbling algorithms). Here we show that it does. We also consider minimizing leaf queries.

4.1 Minimizing All States

Let Π be a black pebbling algorithm for $T = T_2^h$ (the binary tree of height h). Let $States(k, \Pi)$ be the number of non-output states in the k-way BP that implements Π in order to solve $FT^h(k)$.

In general we assume $h \geq 1$ and $k \geq 2$.

We can define $States(k, \Pi)$ as follows. For $1 \leq i \leq t$ let p_i be the number of pebbles on the tree before the ith pebble placement in Π (here t is the total number of placements). Note that the pebble placement at step i requires k^{p_i} states in the corresponding BP; one state for each of the possible values of the pebbled nodes. Thus

$$States(k, \Pi) = \sum_i k^{p_i}$$

Now let $SS(k, h)$ (SS stands for Standard States) be $States(k, \Pi)$, where Π is the standard pebbling algorithm for T^h.

Theorem 50

$$SS(k, h) = (k + 1)^h - k$$

Proof: We use induction on h. For $h = 1$ just one pebble is required, so the base case follows. For the induction step, the standard pebbling algorithm first pebbles the left principal subtree, keeps a pebble on its root, then pebbles the right principal subtree, and then with a pebble on each child of node 1, finally pebbles node 1. Thus

$$\begin{aligned} SS(k, h + 1) &= SS(k, h) + k \cdot SS(k, h) + k^2 && (48)\\ &= (k + 1)[(k + 1)^h - k] + k^2 && (49)\\ &= (k + 1)^{h+1} - k && (50) \end{aligned}$$

■

For each k, h let $MinStates(k, h)$ be the minimum of $States(k, \Pi_h)$ for all pebbling algorithms Π_h for T^h.

Theorem 51 *For all* k, h

$$MinStates(k, h) = SS(k, h) = (k + 1)^h - k$$

Proof: Obviously $MinStates(k, h) \leq SS(k, h)$. We prove the reverse inequality by induction on h. The base case $h = 1$ is clear. For the induction step, (from h to $h + 1$) consider the last step in Π_{h+1} before the root is pebbled that one of the principal subtrees (say the left one) of T^{h+1} is empty. From this point on until node 2 is pebbled there is at least one pebble on the right subtree, so (by the Induction Hypothesis) the pebble placements on the left subtree contribute at least $k \cdot SS(k, h)$ states which query nodes in this subtree. Also node 3 must be pebbled at some point, so by the I.H. there must be at least $SS(k, h)$ states which query nodes in the right subtree. Finally there must be at least k^2 states which query node 1. Referring to (48) we see that $MinStates(k, h + 1) \geq SS(k, h + 1)$. ■

4.2 Minimizing Leaf Queries

Let $Leaf(k,\Pi)$ be the number of states which query leaves in the BP corresponding to the pebbling algorithm Π. As before, let p_i be the number of pebbles in the tree before the pebble is placed at step i of Π. Let $leaf$ be the set of all i such that a pebble is placed on a leaf at step i of Π. Then

$$Leaf(k,\Pi) = \sum_{i \in leaf} k_i^p$$

For each k,h let $MinLeaf(k,h)$ be the minimum of $Leaf(k,\Pi_h)$ for all pebbling algorithms Π_h for T^h.

Theorem 52

$$MinLeaf(k,h) = (k+1)^{h-1}$$

Proof: It is easy to see by induction on h that the standard pebbling algorithm achieves the stated bound. Hence it suffices to prove the lower bound by induction on h. The base case $h = 1$ is obvious. For the induction step we argue as in the proof of Theorem 51. Starting from the last time before the root is pebbled that one principal subtree is empty we note that the leaves on that subtree must be pebbled while the other subtree has at least one pebble, and the leaves of the other subtree must also be pebbled. Hence

$$MinLeaf(k,h+1) \geq k(k+1)^{h-1} + (k+1)^{h-1} = (k+1)^h$$

■

Conjecture: Every deterministic thrifty BP which solves $FT^h(k)$ has at least $MinLeaf(h,k) = (k+1)^{h-1}$ states which query leaves.

Finally we consider sequential BPs (see Section 3). We are interested in the number of leaf states required for a sequential BP to solve $FT^3(k)$ assuming that it has k initial states, one for each possible value of v_2, and it is not allowed to query v_2 later. (This is the problem $RR3(k)$ described in Definition 7.)

The corrsponding pebbling problem is to pebble T^3 with a permanent pebble placed on node 2 throughout. It is easy to see that the minimum number of states in any BP which corresponds to such a pebbling algorithm is $k(k+1) = k^2 + k$. This is slightly larger than the lower bound of k^2 stated in Theorem 30. (See the comment at the end of the proof.)

5 Read-Once Lower Bound

We give two lower bound proofs for deterministic read-once BPs solving FT_2^h. The first one is due to James. The second one is an improvement, due to Siuman. Corollary 63 to the second Theorem states that deterministic read-once BPs solving $FT_2^h(k)$ have $\Omega(k^h)$ states, showing that black-pebbling gives the optimal read-once algorithm.

5.1 First Theorem

Theorem 53 *Let $k = p^{2r}$, where p is prime and $r \geq 1$. Let $h \in \{1, \dots, \sqrt{k}\}$. Then any deterministic read-once branching program which solves the tree evaluation problem FT_2^h must have at least $k^{(h-1)/2} = p^{r(h-1)}$ states which query leaves.*

We need the following definitions:

5.1.1 The Internal Nodes

Let $\mathbf{F} = \mathbf{F}_{\sqrt{k}}$ be the field of order $\sqrt{k}$. For $a, b \in \mathbf{F}_{\sqrt{k}}$, let $[a, b] = a + b\sqrt{k} \in [k]$, so that $[\cdot, \cdot]$ is a bijection between $\mathbf{F}^2$ and $[k]$.

We consider trees in which every internal node is labeled with the function $\varphi : [k] \times [k] \to [k]$, defined by

$$\varphi([a_0, b_0], [a_1, b_1]) = [a_0 a_1, a_0 b_0 + a_1 b_1].$$

The internal nodes being fixed, henceforth an *input* ℓ shall be an assignment of values to the leaves of the tree $\ell = (v_{2^{h-1}}, \dots, v_{2^h - 1})$.

5.1.2 i-Siblings

Node v's 0-parent is v itself, and in general v's $i+1$-parent is the parent of v's i-parent. i-children are defined oppositely.

Node v's only 0-sibling is v itself. In general, an *i-sibling* of v is any i-child of v's i-parent which is not already a j-sibling of v for any $j < i$. For example, 1-siblings are ordinary siblings, and 2-siblings are cousins.

5.1.3 The Final-Leaf Function

Let $i \in \{2^{h-1}, 2^{h-1} + 1, \dots, 2^h - 1\}$ be the last leaf queried by some computation that queries each leaf exactly once. Let $\ell = (v_{2^{h-1}}, \dots, v_{2^h - 1})$ be a tuple assigning a value to every leaf. Then the *final-leaf function* $f_{i,\ell} : [k] \to [k]$ is defined as follows: for any $x \in [k]$, $f_{i,\ell}(x)$ is the value of the root of the tree if the leaf i is assigned value x, the other leaves are assigned their values from ℓ, and the internal nodes compute the function defined in Section 5.1.1.

5.2 Proof of Theorem 53

Throughout this section, B shall be any read-once branching program which correctly computes the tree-evaluation problem, and S shall be the set of states of B.

We first show in Lemma 54 that at the time B makes its last query to a leaf, it must remember the entire final-leaf function, so it must have at least as many states as there are possible functions at that point. Then, we show in Lemma 55 that there is at least one distinct possible final-leaf function for every degree-$h-1$ polynomial over the field $\mathbf{F}_{\sqrt{k}}$. These two lemmas complete the proof.

Lemma 54 *There exists a function $G : S \to [k] \to [k]$ such that for any leaf assignment $\ell = (v_{2^{h-1}}, \ldots, v_{2^h-1})$, if i is the last leaf queried by B on input ℓ and $s \in S$ is the state which makes that query, then $G(s) = f_{i,\ell}$. That is, the state s encodes the entire final-leaf function for the last leaf queried by B.*

Proof: The values of the internal nodes are all predetermined, so the computation of B from state s to the output state depends only on the value of the leaf i. Since the program is read-once, the state s does not depend on the value x of i. Hence the value of $f_{i,\ell}(x)$ is completely determined by s and x (independent of ℓ). ■

Lemma 55 *Call a function $g : [k] \to [k]$* attainable *if there is some input $\ell = (v_{2^{h-1}}, \ldots, v_{2^h-1})$ which forces B's final-leaf function to be g. There are at least $k^{(h-1)/2}$ attainable functions.*

5.2.1 Proof of Lemma 55

Using the operation $[\cdot,\cdot]$ from Section 5.1.1, for every leaf i with value v_i, let $[a_i, b_i] = v_i$.

Let $i \in \{2^{h-1}, \ldots, 2^h - 1\}$ be the index of any leaf. Then for $j \in \{1, \ldots, h-1\}$, let $C_j(i)$ be the set of all j-siblings of i, and let $\Sigma_j(i) = \sum_{m \in C_j(i)} b_m$.

Lemma 56 *Let i be the index of any node. If the value of every leaf below i is of the form $[1, b]$ (that is, its first component is 1), then i has the value $[1, b']$, where b' is the sum of all the b-values of those leaves.*

Proof: Observe that the internal node function given in Section 5.1.1, on inputs $[1, b_0]$ and $[1, b_1]$, assumes the value $[1, b_0 + b_1]$. ■

Lemma 57 *Let i be a leaf index and suppose that for all leaves $j \neq i$, $a_j = 1$. Then the final function $f_{i,\ell}$, when restricted to inputs of the form $[a, 0]$, is*

$$f_{i,\ell}([a,0]) = \left[a, \sum_{j=1}^{h-1} \Sigma_j(i) a^{h-j-1} \right].$$

Note that the second component of $f_{i,\ell}$ is the degree-$h-2$ polynomial in a with coefficients $\Sigma_j(i)$: in particular, every choice of values (Σ_j) gives a different final function. (Recall that Theorem 53 assumes $h \leq \sqrt{k}$, so a degree $h-1$ polynomial cannot be identically 0 over $\mathbf{F}$.)

Proof: Assume without loss of generality that i is the leftmost leaf: $i = 2^{h-1}$. Let $x_1 = i, x_2, \ldots, x_{h-1}, x_h = 1$ be the indices of the nodes on the path from leaf i to the root of the tree, with values $v_{x_j} = [a_{x_j}, b_{x_j}]$.

Notice that every node x_j has as its right child the root of the subtree whose leaves are the set $C_{j-1}(i)$: it follows by Lemma 56 that node x_j has as its right input the value $[1, \Sigma_{j-1}(i)]$. Then the b-value of the root is the polynomial in a stated in the Lemma, evaluated by Horner's Rule by climbing the path from i to the root. To see this, prove by induction that for every $m \in \{1, \ldots, h\}$,

$$v_{x_m} = \left[a, \sum_{j=1}^{m-1} \Sigma_j(i) a^{m-j-1} \right].$$

■

Lemma 58 *For any choice of values $s_1, \ldots, s_{h-1} \in \mathbf{F}_{\sqrt{k}}$, there exists an input $\ell = (v_{2^{h-1}}, \ldots, v_{2^h-1})$ such that the following two properties hold. Let i be the index of the final leaf queried by B on input ℓ. Then*

- *For all $j \neq i$, $a_j = 1$.*
- *For all $j \in \{1, \ldots, h-1\}$, $\Sigma_j(i) = s_j$.*

Proof: Begin with all the leaf values unspecified, and run the branching program B. Every time B queries a leaf (except for the last time) decide the value to return according to the following algorithm:

1. Let m be the index of the leaf being queried.
2. If m's 1-sibling is still unspecified, let $j = 1$. Otherwise, let j be the smallest integer such that m has at least one j-sibling that has not yet been specified.
3. Let D be the set of all leaves which are r-siblings of m for any $r < j$, excluding m itself: $D = \cup_{r=1}^{j-1} C_r(m)$.
4. Specify the b-value of leaf m to be

$$b_m = s_j - \sum_{x \in D} b_x.$$

After $2^h - 1$ leaf queries have happened, the only leaf b-value that is still unspecified is that of the final leaf, b_i. It suffices to show that the following invariant holds at every step of the interaction.

- Let i be the index of any leaf whose b-value has not yet been specified. Then for every $j \in \{1, \ldots, h-1\}$, the b-value of $\Sigma_j(i)$ is either undetermined (due to some leaf b-value in $C_j(i)$ still being unspecified) or is equal to s_j.

The invariant is true at the beginning of the interaction simply because $\Sigma_j(i)$ is undetermined for every i and j. Now, assume the invariant holds at one point in time, and the b-value of a new leaf m is specified. Consider the b-value $\Sigma_{j'}(i)$ for any j' and any leaf $i \neq m$ whose b-value has not yet been specified. If the b-value of $\Sigma_{j'}(i)$ was determined before leaf m was specified, it must already be correct, so the only case we are interested in is that $\Sigma_{j'}(i)$ was not determined before and is determined now. j' must be equal to the j-value chosen in the algorithm: for if j were less than j', then m's unspecified j-sibling would leave $\Sigma_{j'}(i)$ undetermined, and if j were greater than j', then the value of i would already have been determined. Therefore, the set $C_{j'}(i) = C_j(i)$ is equal to $D \cup \{m\}$ (taking the D used in the algorithm), so

$$\Sigma_{j'}(i) = \sum_{x \in D} b_x + b_m = \sum_{x \in D} b_x + s_j - \sum_{x \in D} b_x = s_j.$$

■

Lemmas 57 and 58 together show that the final function can be forced to be any degree-$h-1$ polynomial over $\mathbf{F}_{\sqrt{k}}$, which completes the proof of Lemma 55.

5.3 Second Theorem

Theorem 59 *Any deterministic read-once BP which solves FT_2^h has $\Omega(k^{h-1})$ states which query leaves.*

Proof: The proof is similar to that of the previous theorem, but now we use a different field, and the node values are field elements, rather than pairs of field elements. Fix a deterministic read-once branching program B which solves FT_2^h.

Let d be an odd positive integer. We let the field F be $GF(2^d)$, and set $k = 2^d$. In general the nonzero elements of the field $GF(q)$ form a cyclic group of order $q-1$, so if e is relatively prime with $q-1$ then the polynomial x^e is a permutation polynomial of the field $GF(q)$. (A permutation polynomial of a field is a polynomial that permutes the elements of the field.) Since $2^d - 2$ is divisible by 3 when d is odd, it follows that $gcd(3, 2^d-1) = 1$, so x^3 is a permutation polynomial of $F = GF(2^d) = GF(k)$.

We assume $k > 3^{h-1}$, so that distinct polynomials over F of degree 3^{h-1} represent distinct functions on F.

We fix the function f_i assigned to each internal node of T_2^h to be

$$f_i(a,b) = a^3 + b^3$$

[NOTE: Alternatively we could take $f_i(a,b) = (a+b)^3$]

With this assignment of functions to internal nodes, the following lemma is easily proved by induction on the height of node r, using the fact that x^3 is a permutation polynomial.

Lemma 60 *For each internal node r of T_2^h and element $a \in F$ and each leaf i in the subtree rooted at r and each assignment of elements in F to the leaves other than i, there is a value $v_i \in F$ for i which makes $v_r = a$, where v_r is the value of node r.*

From this lemma it follows that, for each possible tuple $(s_1, s_2, \ldots, s_{h-1})$ of $h-1$ field elements and each leaf i, there is an assignment of field elements to the leaves other than i such that the values of the siblings of the nodes on the path from the root to the node i (excluding the root itself) form the sequence $(s_1, s_2, \ldots, s_{h-1})$.

Thus if x is the value of the leaf i then value of the root is

$$f_1(x) = s_1 + (s_2 + \ldots + (s_{h-2} + (s_{h-1} + x^3)^3)^3 \ldots)^3 \tag{51}$$

Lemma 61 *For each distinct sequence $(s_1, s_2, \ldots, s_{h-1})$ of field elements the polynomial $f_1(x)$ in (51) is distinct.*

Proof: The proof is by induction on h. If we write the polynomial $f_1(x)$ as a sum of monomials $a_i x^i$ then the highest degree monomial is $x^{3^{h-1}}$ and the second-highest degree monomial is the same as the second highest one in $(s_{h-1} + x^3)^{3^{h-2}}$, namely $3^{h-2} s_{h-1} x^{(3^{h-1}-3)}$. Since F has characteristic 2, the coefficient $3^{h-2} s_{h-1}$ is the same as s_{h-1} in F. Hence the value of s_{h-1} is determined by the polynomial $f_1(x)$. If we set $y = s_{h-1} + x^3$ (note that this map from x to y is a bijection) then (51) becomes

$$f_1(x) = s_1 + (s_2 + \ldots + (s_{h-2} + y^3)^3 \ldots)^3$$

so $s_1, \ldots, s_{h-2}$ are determined by the induction hypothesis. ■

It follows from the above lemma that there are k^{h-1} distinct polynomials of the form (51). Since we assume k exceeds their degree 3^{h-1}, there are k^{h-1} distinct functions $f_1(x)$ of the form (51).

To finish the proof of the theorem we need an analog of Lemma 58 in the proof of Theorem 53. This will show that for each tuple $s_1, \ldots, s_{h-1} \in F$ there is an assignment of elements to the leaves of T_2^h such that if x is the value of the last leaf read by the program, then the value of the root is $f_1(x)$ as given in (51). Hence there must be at least k^{h-1} distinct states in B which read the last leaf in some computation.

Recall from the proof of Theorem 53 that for $j \in \{1, \ldots, h-1\}$, $C_j(i)$ is the set of all j-siblings of i. Then we define $root_j(i)$ to be the root of the subtree whose leaves are the set $C_j(i)$, and we define $S_j(i)$ to be the value of $root_j(i)$.

Lemma 62 *For any choice of values $s_1, \ldots, s_{h-1} \in F$ there exists an input $\ell = (v_{2^{h-1}}, \ldots, v_{2^h-1})$ such that if i is the last leaf queried by B on input ℓ then for all $j \in \{1, \ldots, h-1\}$, $s_{h-j} = (S_j(i))^3$.*

Proof: The proof is similar to the proof of Lemma 58. Steps 1. and 2. of the algorithm to evaluate a leaf are the same as before, but the last two steps are replace by

$3'$. The value v_m of leaf m is that which according to Lemma 60 makes $v_r = (s_{h-j})^{1/3}$, where $r = root_j(i)$.

Now the invariant becomes

- Let i be any leaf whose value has not yet been specified. Then for every $j \in \{1, \ldots, h-1\}$, $S_j(i)$ is either undetermined (due to some leaf in $C_j(i)$ still being unspecified) or is equal to $(s_{h-j})^{1/3}$.

The proof of the invariant consists in observing that whenever the algorithm fixes the value of any node of height j, that value is $(s_{h-j})^{1/3}$. ■

■

Corollary 63 *Any deterministic read-once BP solving $FT_2^h(k)$ has $\Omega(k^h)$ states.*

Proof: This follows from Theorems 59 and 1. Difficult instances of $FT_2^h(k)$ are obtained by assigning functions to the nodes in the top $h-2$ layers according to the proof of Theorem 59 when the overall tree height is $h-1$. The functions assigned to layer 2 from the bottom are those described in the proof of Theorem 1 (i.e. zero everywhere except at some specific pair r, r' of child values). The values of the sibling pairs of the leaves are r, r'. ■

6 Thrifty lower bound - alternative proof

Theorem 64 *Any deterministic thrifty BP solving $FT_2^h(k)$ has at least k^h states.*

Proof: The definition of critical states used in this proof is the same as in the original proof, but the way of assigning pebbling sequences to inputs is slightly different, so to ensure consistency we'll start by repeating the definition of the critical state sequence of an input.

Fix an input I, and let $P = P_I$ be its computation path. We will choose n states on P, one for each node of the tree, as critical states for I. The critical state q_1^I for the root is the last state on P that queries the root. In general for internal nodes i, the critical states q_{2i}^I and q_{2i+1}^I are the last states on P before q_i^I that query nodes $2i$ and $2i+1$ (the thrifty condition ensures that these states exist). Note that if j is a descendant of i then q_j^I occurs before q_i^I on P.

The pebbling algorithm associated with the computation path P is slightly different from before, in that now a node i is pebbled (and the pebbles on the children removed if i is not a leaf) at the state q' following the critical state q_i rather than at q_i itself. The intuitive reason for this is (assuming i is not the root) by the thrifty property q' 'knows' the value of v_i of node i, since the parent of i will be queried before i is again queried.

This revised pebbling algorithm assigns a pebbling configuration to each state on P, such that the set of pebbled nodes in each configuration is a minimal cut of the tree or a subset of some minimal cut, and any two adjacent configurations are either identical, or else the later one follows from the earlier one by valid pebbling moves. This assignment can be described inductively by starting with the last state in P and working backwards. The pebbling configuration for the output state has just a black pebble on the root. Assume we've defined the pebbling configurations for q and every state following q on P. Let q' be the state before q on P. If q' is not critical, then its pebbling configuration is the same as that of q. If q' is critical then it must query a node i that is pebbled in q. The pebbling configuration for q' is obtained from the configuration for q by removing the pebble from i and adding pebbles to $2i$ and $2i+1$ (if i is an internal node - otherwise you only remove the pebble from i). This is the main property of the pebbling sequences that we will use directly:

Fact 65 *If non-root node i is pebbled at critical state q_j^I, then there is a later critical state $q_{i'}^I$ of I that queries the parent i' of i, and there is no state (critical or otherwise) between q_j^I and $q_{i'}^I$ on P that queries i.*

Define the supercritical state q_c^I of I to be the last critical states of I on P whose associated pebbling configuration is a bottleneck. Hence by the black pebbling lower bound for binary trees of height h, there are at least h nodes pebbled at q_c^I.

Remark Similiar to the original proof, at this point we could restrict the set of inputs to be the k^n inputs whose functions f_i are zero everywhere except possibly when the arguments are the values of the children of node i. We would still get a lower bound of k^h states. However, doing so does not simplify this proof, and might make it less robust in the following sense: Possibly this proof could be adapted to work for subsets of the inputs that satisfy some statistical properties, but have no obvious "simple" description, in contrast to the set of k^n inputs just mentioned.

Let Q_c be the set of states that are supercritical for some input. Let E be the set of inputs and let $N_E := (2^{h-1} - 1)k^2 + 2^{h-1}$ be the number of k-valued input variables. Thus $|E| = k^{N_E}$. Let A (for 'Advice') be the set of tuples of $[k]$ of length $N_A := N_E - h$. We will now define an onto function $G : Q_c \times A \to E$, which implies $|Q_c| * |A| \geq |E|$ and so $|Q_c| \geq k^h$, which completes the proof. G will have the property that for every $I \in E$ there is some advice $a_I \in A$ such that $G(q_c^I, a_I) = I$.

To be precise, we will define G using a deterministic algorithm M which takes as input an element of $Q_c \times A$. On input $\langle q_c, a\rangle$, algorithm M constructs a path $q_0, e_0, q_1, \ldots, e_{m-1}, q_m$ through the branching program, where $q_0 = q_c$, the output state is q_m, and e_t is an edge from q_t to q_{t+1}. M maintains a partial assignment v^* of values in $[k]$ to nodes. Initially $v^*(i)$ is undefined for all nodes i.

The idea is that M learns the values of the children of each node queried, because of the thrifty property. It uses this information to answer a query to a node i which is the child of a previously-queried node. Otherwise it uses the next element in the advice string a as the value of i. There are at least h pebbled nodes in the bottleneck configuration, and M will query each of their parents before they are queried, and hence learn their values. This is why the advice a can have h fewer elements than there are input variables.

Back to the formal description: M indexes a using the variable l, which is initially zero. The main loop of M follows. At state q_t, M does the following:

1. If q_t is an output state, set $v^*(1)$ to its label.[4] Let $i_1 < i_2 < \cdots < i_z$ be the entries of v^* that are still undefined. Set $v^*(i_1) := a[l], v^*(i_1 + 1) := a[l + 1], \ldots, v^*(i_z) := a[l + z - 1]$.[5] Then exit this loop.
2. Let X be the input variable queried by q_t, and i the corresponding node of the tree.
3. If $X = f_i(y, y')$, then set $v^*(2i) := y$ and $v^*(2i + 1) := y'$.
4. If $v^*(i)$ is defined, then take the edge e_t with that label (i.e. set q_{t+1} to the state e_t points to).
5. Otherwise $v^*(i)$ is undefined. Take the edge e_t with label $a[l]$. Set $v^*(i) := a[l]$. Then increment l by 1.

Before giving the remainder of the algorithm M, we will recursively define the first $n - h$ elements of the advice a_I for I. Suppose we've defined the first m elements a'_I of a_I for some non-negative integer $m < n - h$. Consider the computation of M on $\left\langle q_c^I, a'_I a''\right\rangle$, for arbitrary a'' of length $N_A - m$. If the condition of (1) in the main loop is satisfied before M looks at a'', then define the next $n - h - m$ elements of a_I to be the values I takes on the undefined entries of v^*

[4]This isn't actually necessary: if the advice is good then $v^*(1)$ will already be defined by now

[5]We will argue later that $l + z - 1$ will never exceed the length of a when $\langle q_c, a\rangle = \left\langle q_c^I, a_I\right\rangle$ for some I

(in increasing order of node indices). Otherwise, M reaches a state q_t where the condition of (5) in the main loop is satisfied, and so far the computation does not depend on a''. Let i be the node queried by q_t. Define the next element of a_I to be v^I_i.

Let p be the number of pebbles in the bottleneck configuration of I. Let a'_I be the first $n-h$ elements of a_I, defined above. Let a'' be an arbitrary tuple of elements of $[k]$ of length $N_A-(n-h)=N_E-n$. It follows from the earlier stated fact about the assigned pebbling sequences that M on input $\langle q^I_c, a'_I a''\rangle$ will use at most the first $n-p \leq n-h$ elements of a'_I before satisfying the condition of (1). Also v^* will be defined for all p of the pebbled nodes.

After the main loop, v^* is defined for all nodes. Hence there are exactly $|E|/k^n = k^{N_E-n}$ inputs consistent with v^* (and I is one of them), and at most that many which also have q^I_c as their supercritical state. Hence we can use the remaining N_E-n undefined elements of a_I to uniquely specify I. So after exiting the main loop, M outputs the input I coded by the pair of v^* and the last N_E-n elements of a.

■

7 $(k{+}1)^h{-}k$ is exact for BPs that are both thrifty and read-once

This section is most-related to Section 4. We write TE$^h(k)$ as an abbreviation for "BT$^h_2(k)$ or FT$^h_2(k)$". We show that a BP solving TE$^h(k)$ has minimum depth (defined below) if and only if it is both thrifty and read-once (Fact 68), and that the upper bound of $(k+1)^h-k$ non-output states for FT$^h_2(k)$ mentioned in Section 4 is the exact minimum for these (very restricted) BPs (Theorem 69).

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 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 TE$^h(k)$ is **min-depth** if it has depth 2^h. We will use the following results from [Weh10]:

Lemma 66 *Every min-depth BP solving TE$^h(k)$ is thrifty.* [6]

Lemma 67 *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 69 is the goal. We will not use the next fact in the proof, but it may be worth noting: Lemmas 66 and 67 charaterize the min-depth BPs solving TE$^h(k)$, in the following sense:

Fact 68 *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 66 and 67. 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). ■

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

Theorem 69 *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 70 the theorem follows if we can show for arbitrary such B that:

$$\begin{aligned} &|\mathsf{States}(B,1)| \geq (k+1)^{h-1} \\ \text{and}\quad &|\mathsf{States}(B,l)| \geq k^2(k+1)^{h-l} \quad \text{for } 2 \leq l \leq h \end{aligned} \tag{52}$$

Lemma 70

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

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 71 *If $|\mathsf{States}(B,2)| \geq k^2(k+1)^{h-2}$ for every h and B, then (52) holds for every h and B.*

Proof: Assume the hypothesis holds. Let B be a min-depth BP that solves BT$_2^h(k)$ (the proof is the same for 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 BT$_2^{h+1}(k)$. Replace each state that queries a leaf variable with a copy of the BP for 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 66). 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 72 *$Q^2_{t_1} \cap Q^2_{t_2} = \emptyset$ 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 67 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 73 *Let #ones(t) be the number of 1s in the binary representation of $t \geq 0$. Then $\vec{z}_2(t) = 2 + \#\mathit{ones}(t-1)$ for $t \geq 1$.*

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

Lemma 74 $\sum_{t=1}^{2^{h-l}} k^{\vec{z}_l(t)} = k^2(k+1)^{h-l}$ *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 66), 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 66 and 67, 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 76.

Lemma 75 *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 73. 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 73 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 76 *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 75, 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 72, 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 76 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 74 (for $l = 2$), we finish the proof:

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

■

8 DAG Half-Pebble Lower Bound

Theorem 14 (due to Rahul) in the arXiv version of our journal paper on tree evaluation (15 May 2010) states

'If a DAG D has a fractional pebbling using p pebbles, then it has a black-white pebbling using at most 2p pebbles.'

The proof idea is simple. We adapt the fractional pebbling procedure FRAC to the B/W procedure WHOLE so that if at time t FRAC has a half or more black pebble on node v, then WHOLE has a whole black pebble on v at time t. If FRAC has less than half a black pebble, then WHOLE has no black pebble.

Similarly for white pebbles, but for consistency we need change 'half or more' to 'more than half', and 'less than half' to 'half or less'.

This is easy to implement. The crucial point is that if at time t FRAC has a combined black-white weight of 1 on node v, then at time t WHOLE has either a whole black or a whole white pebble on v at time t.

It is obvious that for each node v and each time t, WHOLE has at most twice the weight of FRAC on v at time t.

We can use the same idea to prove the following:

Theorem 77 *[Steve] If a DAG D has a fractional pebbling using p pebbles, then it has a fractional pebbling using at most 1.5p pebbles such that the only proper pebble fraction allowed is $1/2$.*

Proof: We modify the above proof as follows. We choose parameters r, s such that

$$0 < r < 1/2 < s < 1$$

and

$$r + s = 1 \tag{53}$$

Now given a fractional pebbling FRAC we modify it to form a pebbling HALF as follows. If at time t node v has a fraction b of black pebble, then HALF has black pebble weight given by the following table:

FRAC	HALF
$0 \le b < r$	0
$r \le b < s$	$1/2$
$s \le b \le 1$	1

Similarly, if v has a fraction w of white pebble at time t, then HALF has white pebble weight given by

FRAC	HALF
$0 \le w \le r$	0
$r < w \le s$	$1/2$
$s < w \le 1$	1

By the constraints on r and s, it is easy to check that for any pair b, w for node v at time t in FRAC such that $0 \leq b + w \leq 1$, the total pebble weight for v at time t in HALF never exceeds 1. Further, if $b + w = 1$, then the total pebble weight in HALF is 1.

The ratio of the weight of v at time t of HALF/FRAC is at most

$$\max\{.5/r, 1/s\}$$

This is minimized (subject to the constraints on r, s) when $r = 1/3$ and $s = 2/3$. The maximum blow-up ratio for this r, s is thus 3/2. ■

References

[CMW+10] Stephen Cook, Pierre McKenzie, Dustin Wehr, Mark Braverman, and Rahul Santhanam. Pebbles and branching programs for tree evaluation, 2010. arXiv:1005.2642.

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