A lower bound for read-once-only branching programs

László Babai
University of Chicago and
Eötvös University, Budapest

Péter Hajnal
University of Chicago and
Eötvös University, Budapest

Endre Szemerédi
University of Chicago and
Hungarian Academy of Sciences, Budapest

György Turán
University of Illinois at Chicago and
Hungarian Academy of Sciences, Szeged

Abstract.

We give a C^n lower bound for read-once-only branching programs computing an explicit Boolean function. For $n = \binom{v}{2}$, the function computes the parity of the number of triangles in a graph on v vertices. This improves previous $\exp(c\sqrt{n})$ lower bounds for other graph functions by Wegener and Zák. The result implies a linear lower bound for the space complexity of this Boolean function on "eraser machines", i.e. machines that erase each input bit immediately after having read it.

1. Introduction.

1.1. Branching programs.

A Boolean function in n variables is a mapping from the set of 2^n (0,1) input strings to $\{0, 1\}$. Several models of computation of such functions have been considered in the literature (Turing machine, Boolean circuit, decision tree, Boolean formula, etc.). Branching programs are a model generalizing decision trees. The program is a directed acyclic graph. To avoid confusion we shall use the terms *nodes* and *arcs* to refer to the elements of this digraph. (We shall use branching programs to do computation on graphs; these graphs (input objects) will have *vertices* and *edges*.)

One of the nodes of the branching program is a source (has fan-in zero) and is called START, some other nodes are sinks (fan-out zero) and are called *terminal* nodes. All non-terminal nodes have fan-out two. The two arcs leaving a non-terminal node are labeled 0 and 1. Each non-terminal node is labeled by an input variable and each terminal node is labeled 0 or 1. We may assume without loss of generality that the program is *leveled*, START is on level one and arcs go from each level to the next level only.

Each input string $\alpha = \alpha_1 ... \alpha_n$ defines a unique path from START to a terminal node: the *computation path* determined by α. This path, after entering a nonterminal node labeled x_i, proceeds along the arc labeled α_i. The path ends at a terminal node. The function f computed by this branching program is defined by setting $f(\alpha)$ equal to the label of this terminal node.

The *size* of a branching program is the number of nodes. The *width* of the program is the maximum number of nodes on any level. The *length* is the number of levels. The *multiplicity of reading* is the maximum number of times any particular variable is encountered as a node label along any computation path.

An easy counting argument shows that most Boolean functions require exponential size branching programs. It is desirable to find nontrivial lower bounds for explicit Boolean functions (functions that belong to P or at least to NP).

The only known lower bound for the size of an unrestricted branching program computing an explicit Boolean function is due to Nečiporuk [Ne], [Sa] and is $\Omega(n^2/\log^2 n)$. P. Beame and S. Cook observed [BC] that Nečiporuk's technique actually applies to the "element distinctness" problem in the following sense. Let $x_1, \ldots, x_m$ be m integers between 1 and m^2. Written in binary, they form the input string of length $n = 2m \log m$. Then any branching program deciding whether or not all the x_i are distinct must have size $\Omega(m^2) = \Omega(n^2/\log^2 n)$.

Another approach that has recently gained popularity is proving lower bounds for branching programs with bounds on various "resources" (width, multiplicity of reading). A similar approach to Boolean circuits has been quite successful recently [Ya2], [An], [Ra], [Ha], [AB], [Be].

Our aim is to present a result of this kind.

1.2. Limited reading

A *read-k-times-only* branching program is allowed to encounter each variable at most k times along any computation path. This hierarchy of classes of branching programs was introduced by Masek [Ma]. Wegener [We] conjectures an exponential gap between the levels of this hierarchy and gives candidate Boolean functions computable with polynomial size read-k-times-only programs but conjectured to require exponential size read $(k-1)$-times-only programs.

No superpolynomial lower bounds are known, however, even for read-twice-only branching programs computing an explicit Boolean function, and no such bound will appear in this paper.

In connection with the history of read-once-only branching programs we should mention a paper by Fortune, Hopcroft and Schmidt [FHS]. In the context of program schemes, they gave an $\exp(c\sqrt{n})$ lower bound for computing an explicit function by read-once-only branching programs satisfying the additional restriction that the variables have to be examined in precisely the same order along each computation path. Without this restriction, however, their function is computable by a read-once-only branching program of polynomial size and is indeed defined by such a program.

Wegener [We] and Zák [Za] independently prove an $\exp(c\sqrt{n})$ lower bound for read-once-only branching programs computing certain, clique related graph properties. Wegener's property is NP-complete (presence of a clique of size $v/2$ where v is the number of vertices), Zák's is polynomial time decidable (recognizing the graphs that consist of a clique of size $v/2$ and $v/2$ isolated vertices.) We shall improve the lower bound to C^n (for a different function, also a polynomial time decidable graph property) (Section 3).

1.3. Space-complexity: the eraser RAM

It has been noted ([Ma], [BFKLT], [Pu]) that a lower bound $S(n)$ on the size of the smallest branching program computing a Boolean function f_n of n variables implies an $\Omega(\log S(n))$ lower bound on the space complexity of the family $\{ f_n : n = 1, 2, \ldots \}$ on any reasonable model of computation.

The Fortune-Hopcroft-Schmidt result mentioned above corresponds to *on-line space complexity:* the input bits are read once and in a given order only. The [FHS] result provides an $\Omega(\sqrt{n})$ space lower bound for such computation (independently of the given order of input bits).

General read-once-only branching programs suggest the following machine model which we call *eraser RAM.* This is a RAM with a special read-only input tape. The machine decides in the course of the computation in what order to read the input but once an input cell has been read, it is erased. Let us measure the space required by a computation by the number of *bits* stored at any given time on the worktape.

The following is immediate.

Proposition. *If a language L can be recognized by an eraser RAM in space $S(n)$ then the set $L_n = L \cap \{ 0, 1 \}^n$ can be recognized by a read-once-only branching program of size* $\exp(O(S(n)))$.

The results of Wegener and Zák thus imply an $\Omega(\sqrt{n})$ lower bound for the eraser RAM space complexity of their respective Boolean functions. Our result implies a linear lower bound on the same model.

2. Read-once-only branching programs: the result

Let $n = \binom{v}{2}$ and let us fix a bijection between the set $\{ 1, \ldots, n \}$ and the set of pairs from $\{ 1, \ldots, v \}$. Each string $x = x_1 \ldots x_n \in \{ 0, 1 \}^n$ can be thought of as representing a graph $G(x)$ on the vertex set $\{ 1, \ldots, v \}$. The value of each input variable corresponds to the presence or absence of an edge between a given pair of vertices in $G(x)$.

Let $f_n(x)$ denote the number of triangles in $G(x)$ modulo 2.

Theorem. *There exists a positive constant α such that every read-once-only branching program computing f_n has size at least $2^{\alpha n}$.*

First we outline the idea of the proof.

We shall use the term "edge" to mean any of the $\binom{v}{2}$ pairs of vertices. (These are the edges of the complete graph K_v.) Let P be a path in a branching program. We shall say that an arc of P labeled 1 from a node labeled x_e has the effect of *accepting* the edge e; the arc labeled 0 from the same node *rejects* e. The edges accepted by P form the graph $A(P)$, the rejected edges form the graph $R(P)$. The union of these two edge sets constitutes the set $D(P)$ of edges *determined* by P.

Assume f_n is computed by a read-once-only branching program of size less than $2^{\varepsilon n}$ for some appropriately selected small positive constant ε. From this assumption we shall derive

(1) *the existence of a node w in the program, two paths P_0 and P_1 both leading from START to w, and an edge e not determined by either P_i, such that the parity of the number of triangles containing e in the graph $A(P_i) \cup e$ is i.*

The read-once-only property implies that after w, the program follows the same path of computation for input graphs $A(P_0) \cup e$ and $A(P_1) \cup e$ and thus leads to the same terminal node. This means these two graphs have the same number of triangles mod 2; the same observations hold for $A(P_0)$ and $A(P_1)$. This contradicts the choice of the P_i and e.

We proceed to showing how w, e, P_0, and P_1 satisfying (1) are found.

The *depth* of a node is its distance from START.

Proposition 1. *Let P be a path from START to a terminal node. If three edges are undetermined by this path, they cannot form a triangle.*

Proof. Suppose, to the contrary, that the edges e_1, e_2, e_3 of a triangle are left undetermined by P. Then the parity of the number of triangles in each graph $A(P) \cup e_i$ must agree with the parity of the number of triangles in $A(P)$. But then adding all the three edges at once will change the parity, a contradiction. $\blacksquare$

Corollary 2. *The depth of each terminal node is at least $v(v-2)/4$.*

Proof: by Turán's Theorem in graph theory (cf. [Lo, Probl.10.30,34]). Any path of length less than $v(v-2)/4$ leaves more than $v^2/4$ edges undetermined, forcing the graph of undetermined edges to contain a triangle. $\blacksquare$

It follows that for any constant $c < 1/4$, there are precisely 2^{cn} computation paths of length cn beginning at START. Since the branching program has size less than $2^{\varepsilon n}$ there exists a node w such that at least $2^{(c-\varepsilon)n}$ paths of length cn connect START to w.

Let us fix c at a quite small value; any $c \leq 10^{-5}$ will be safe. Then, ε must be even smaller; let us set $\varepsilon = c^{3/2}$. At the same time we assume that v is sufficiently large.

Using w as a "checkpoint", we shall classify the edges according to their status at the time various computation paths pass through w. We shall see that these classes exhibit a strong structure.

Let D denote the set of edges determined by at least one path from START to w. Let U denote the set of the remaining (undetermined) edges; $|D| + |U| = n$.

Proposition 3. *Let P be any path from START to w. It is impossible that three edges e_1, e_2, e_3 form a triangle, where $e_1 \in D - D(P)$, $e_2, e_3 \notin D(P)$.*

Proof. The proof is similar to that of Proposition 1. Suppose the contrary. The read-once-only property implies that e_1 is not tested along any path starting at w and therefore the parity of the number of triangles in $A(P)$ and $A(P) \cup \{ e_1 \}$ is the same. In other words, e_1 is contained in an even number of triangles in $A(P) \cup \{ e_1 \}$. Similarly we infer that the number of triangles containing e_1 in the graph $A(P) \cup \{ e_1, e_2, e_3 \}$ is even. But this number is precisely one greater than the number just shown to be even, a contradiction. $\blacksquare$

Let AR denote the set of those edges which are accepted along some path from START to w and are rejected along some other. Clearly, $AR \subseteq D$.

Proposition 4. *There is no triangle e_1, e_2, e_3 with $e_1 \in AR$, $e_2, e_3 \in U$.*

Proof: The proof is a parity argument similar to the proofs of Propositions 1 and 3. Suppose the contrary. Let be P be any path from START to w, accepting e_1 and let Q

be some other path from START to w, rejecting e_1. Then the parity of the number of triangles in $A(P)$ must agree with the parity of the number of triangles in $A(Q)$. Similarly the parity of the number of triangles in $A(P) \cup \{ e_2 \}$ and in $A(Q) \cup \{ e_2 \}$ is the same. In other words the number of triangles containing e_2 in $A(P) \cup \{ e_2 \}$ and the number of triangles containing e_2 in $A(Q) \cup \{ e_2 \}$ have the same parity. The same holds for e_3. Clearly the number of triangles in $A(P) \cup \{ e_2, e_3 \}$ and the number of triangles in $A(Q) \cup \{ e_2, e_3 \}$ are the same mod 2. One can divide the triangles in $A(Q) \cup \{ e_2, e_3 \}$ into three classes, namely the triangles in $A(Q)$, the triangles containing e_2 in $A(Q) \cup \{ e_2 \}$ and the triangles containing e_3 in $A(Q) \cup \{ e_3 \}$. In the case of the triangles in $A(P) \cup \{ e_2, e_3 \}$ one must add the triangle $\{ e_1, e_2, e_3 \}$ to the corresponding classes. This contadicts the parity arguments above. ▮

One can deduce from Proposition 3 that most edges determined along any path between START and w are actually determined along P, i.e. the set $D - D(P)$ is small. Moreover, most edges determined by some path to w are both accepted and rejected along paths to w, i.e. $D - AR$ is a small set. More specifically:

Lemma 5. (a) $|D - D(P)| \leq 3c^{3/2}n$.
(b) $|U| > (1 - c - 3c^{3/2})n$.
(c) $|AR| \geq (c - \varepsilon)n$.
(d) $|D - AR| \leq 4c^{3/2}n$.

Proof. For a set A of edges, let $\deg_A(p)$ denote the degree of p with respect to the graph formed by A.

(a) Let $e = pq$ be any edge in $D - D(P)$. By Proposition 3, every vertex is adjacent in $D(P)$ to at least one end of e. Therefore,

$$\deg_{D(P)}(p) + \deg_{D(P)}(q) \geq (v - 2).$$

Adding up these inequalities for all $pq \in D - D(P)$ we obtain

$$\sum_p \deg_{D-D(P)}(p) \deg_{D(P)}(p) \geq (v-2)|D - D(P)|. \tag{2}$$

On the other hand, also by Proposition 3, the neighborhood in $D - D(P)$ of any vertex p induces a clique in $D(P)$. Therefore

$$\binom{\deg_{D-D(P)}(p)}{2} \leq |D(P)| = cn = c\binom{v}{2}.$$

Consequently,

$$\deg_{D-D(P)}(p) \leq 1 + c^{1/2}v. \tag{3}$$

Combining (2) and (3),

$$|D - D(P)| \leq \frac{1 + c^{1/2}v}{v-2} \sum_p \deg_{D(P)}(p) = \frac{2 + 2c^{1/2}v}{v-2} |D(P)| \leq 3c^{3/2}n.$$

(b) follows immediately from (a) since $|U| = n - |D|$.

(c) Clearly, the logarithm of the number of START-to-w paths is a lower bound for $|AR|$.

(d) By (a), $|D| = |D - D(P)| + |D(P)| \le 3c^{3/2}n + cn$. Combining this inequality with (c) we obtain $|D - AR| \le (\varepsilon + 3c^{3/2})n = 4c^{3/2}n$. ▮

Lemma 5(b) implies that the graph U has a vertex p_0 of degree greater than $d = (1 - c - 4c^{3/2})v$. Let S be a set of precisely d neighbors of p_0 in U and let T be the complement of S ($|T| + |S| = v$).

Proposition 4 implies that no edge in AR has both of its endpoints in S. From this, it follows that AR is "mostly" bipartite, with bipartition (S, T). We can actually deduce even more structure: most vertices in T are adjacent in AR to either almost all or to almost no vertices in S (about half of the vertices will satisfy each alternative). More precisely, let us divide T into three classes, T_0, T_1, T_2. We shall refer to a moderately large constant K, $20 \le K \le 1/(8c^{1/2})$.

Let T_0 consist of those $p \in T$ which have more than $Kc^{1/2}v$ neighbors in S in the graph $D - AR$. We put $p \in T - T_0$ into T_1 or T_2 according to whether p has more AR-neighbors in S than U-neighbors or not. Let $\deg^S_{AR}(p)$ denote the number of AR-neighbors of p in S and analogously for other classes.

Lemma 6. (a) $|T_0| < 2cv/K$.
(b) *For each* $p \in T_1$, $\deg^S_U(p) \le 5c^{3/2}v$.
(c) *For each* $p \in T_2$, $\deg^S_{AR}(p) \le 5c^{3/2}v$.

Proof. By Lemma 5(d),

$$|T_0|Kc^{1/2}v \le |D - AR| \le 4c^{3/2}n.$$

Claim (a) is now immediate.

To prove (b) and (c), let $p \in T - T_0$. Let N_1 and N_2 denote the sets of U-neighbors and AR-neighbors of p in S, resp.; let $n_i = |N_i|$. Since $p \notin T_0$, we have

$$n_1 + n_2 \ge |S| - Kc^{1/2}v > 6v/7. \tag{4}$$

On the other hand, by Proposition 4, all edges between N_1 and N_2 belong to $D - AR$. By Lemma 5(d) it follows that $n_1 n_2 \le 4c^{3/2}n < 2c^{3/2}v^2$. Consequently,

$$\min\{n_1, n_2\} \le \frac{2n_1n_2}{n_1 + n_2} < 5c^{3/2}v. \quad ▮$$

Let X denote the set of AR-edges between T_1 and S.

Corollary 7. (a) $(1 - \frac{8}{K})cv/2 \le |T_1| \le (1 + 4c^{1/2})cv/2$.
(b) $|AR - X| < \frac{3c}{K}v^2$.

Proof. We begin with (b). Clearly,

$$|AR - X| < |T|^2 + |T_2| \max_{p \in T_2} \deg^S_{AR}(p) + |T_0||S|.$$

By definition, $|T_2| \le |T| \le (c + 4c^{3/2})v$. We use Lemma 6(c) to estimate the second term and Lemma 6(a) and the fact $|S| < v$ for the last term.

For the upper bound in (a), we obtain from Lemma 6(b) that

$$|T_1| \le \frac{|D|}{\min_{p \in T_1} \deg^S_D(p)} \le \frac{|D|}{|S| - 5c^{3/2}v}.$$

Lemma 5(a) provides the bound $|D| \le (c + 3c^{3/2})n$. By the definition of S (after the proof of Lemma 5), $|S| = \lfloor (1 - c - 4c^{3/2})v \rfloor$. A combination of these estimates yields the desired upper bound.

For the lower bound we first observe that $|X| > (c - \varepsilon - 7c/K)v^2/2 > (1 - 8/K)cv^2/2$. This follows from Lemma 5(c) and part (b) of this Corollary. On the other hand, trivially, $|T_1| \ge |X|/v$. ▌

The structural consequence of Lemma 6 and Corollary 7 for the AR graph is that the subgraph X induced between T_1 and S is a nearly complete bipartite graph, and X contains almost all edges of AR.

In order to focus on X, let us make a decision on the value of each input variable (edge) in $AR - X$. There are $2^{|AR-X|} < 2^{(3/K)cv^2}$ possible outcomes (by Corollary 7(b)). Let us choose the one that is the most frequent among the START to w computation paths. Having fixed these values,we still have at least

$$2^{(c-\varepsilon)n - (3/K)cv^2} > 2^{\frac{c}{2}v^2(1-8/K)} \tag{5}$$

computation paths left. Let Π denote the set of these paths:

$$\log |\Pi| \ge \frac{c}{2} v^2 (1 - 8/K). \tag{6}$$

(The base of the log is 2.)

Let $t = |T_1|$ and $s = |S|$. We see, that $\log |\Pi|$ is nearly ts. In order to complete the proof, we show, that, unless situation (1) arises, the number of subgraphs of X arising from paths $P \in \Pi$ must be substantially less than 2^{ts}: only about $2^{ts/2}$. This is impossible because different paths define different subgraphs of X. (This in turn is true since the possible branchings on variables in $AR - X$ have been eliminated.)

The proof is based on a linear algebra counting lemma for $GF(2)$.

Let A, B, C be $(0,1)$-matrices of the same dimensions.

We shall say that $A \equiv C \bmod B$ if for every i, j, $B[i,j] = 0$ implies $A[i,j] = C[i,j]$.

Lemma 8. *Let $A_1, \ldots, A_N$ be different $t \times s$ matrices over the two-element field $GF(2)$. Furthermore, let B and C be $s \times s$ matrices over $GF(2)$. Let β be the number of 1's in B. Assume that $A_i^T A_i \equiv C \bmod B$ for every i. Then*

$$\log N < \beta + \frac{t}{2}(s + t + \log s). \tag{7}$$

Proof. First we estimate the number of $t \times s$ matrices of rank $\leq t/2$ over $GF(2)$. There are less than $2^{t^2/2}$ possible choices of the column space. Given the column space of dimension $\leq t/2$, there are $\leq 2^{t/2}$ choices for each column, giving a total of $\leq 2^{t(s+t)/2}$ matrices.

Next, we estimate the number of those A_i having rank $> t/2$. Such a matrix has a set of $t/2$ linearly independent columns; they are positioned in any of $\binom{s}{t/2} < s^{t/2}$ ways. Let us fix their positions, say columns $1, \ldots, t/2$, and decide their entries. Let us estimate, how many ways the remaining columns can be filled. For each pair (i, j) where $1 \leq j \leq t/2 < i \leq s$ and $B[i, j] = 0$, we have a linear condition $\sum_{k=1}^{t} x_{ik} A[k, j] = C[i, j]$ for the prospective entries x_{ik}. All these equations are linearly independent and their number is $\geq t(2s - t)/4 - \beta$. This reduces the number of candidates (2^{ts}) by a factor of $2^{-t(2s-t)/4+\beta}$. The number of those A_i of rank $> t/2$ is thus

$$< s^{t/2} 2^{ts-t(2s-t)/4+\beta} = 2^{\beta+\frac{t}{2}(s+\frac{t}{2}+\log s)}. \tag{8}$$

Add the bound $2^{t(s+t)/2}$ on the number of low rank matrices to this; the figure in (7) is a generous overestimate of logarithm of the sum. ∎

Let now $s = |S|$, $t = |T_1|$ and for each $P \in \Pi$ let A_P be the $t \times s$ adjacency matrix of the bipartite subgraph of X defined by P. (This graph is the restriction to $T_1 \times S$ of $A(P)$.) Let B be the $s \times s$ adjacency matrix of the induced subgraph of $D - AR$ on S. (Recall that the complement, relative to S, of this graph belongs entirely to U by Proposition 4.) Observe that the entries of $A_P^T A_P$ count modulo 2 the number of common neighbors of each pair of vertices in S. The falsity of (1) implies the statement that all the $A_P^T A_P \equiv C \bmod B \bmod 2$ for some fixed $s \times s$ matrix C. The number of 1's in B is $\beta = 2|D - AR| \leq 8c^{3/2}n < 4c^{3/2}v^2$ by Lemma 5(b). Using the upper bound of Lemma 7(a) for t we now infer from Lemma 8 that

$$\log |\Pi| < \beta + \frac{t}{2}(s + t + \log s) < \beta + \frac{t}{2}(v + \log v) < \frac{c}{4}v^2(1 + 20c^{1/2} + \frac{2\log v}{v}). \tag{9}$$

This contradicts (6) for large v, completing the proof of the Theorem. ∎

References

[AB] N. Alon and R. Boppana, The monotone circuit complexity of Boolean functions, *Combinatorica*, to appear

[An] A. E. Andreev, On a method of obtaining lower bounds for the complexity of individual monotone functions (in Russian), *Dokl. Akad. Nauk SSSR* **282/5** (1985), 1033-1037

[Be] P. Beame, Limits on the power of concurrent-write parallel machines, *Proc. 18th ACM STOC*, Berkeley CA 1986

[BC] P. Beame and S. Cook, 1985, private communication

[BFKLT] A. Borodin, M.J. Fischer, D.G. Kirkpatrick, N.A. Lynch and M. Tompa, A time-space tradeoff for sorting on nonoblivious machines, *J.C.S.S.* **22** (1981) 351-364.

[FHS] S. Fortune, J. Hopcroft, E. M. Schmidt, The complexity of equivalence and containment free single variable program schemes, Cornell Univ. TR 77-310

[GRS] R. L. Graham, B. Rothschild and J. Spencer, *Ramsey Theory*, Wiley, New York 1980.

[Ha] J. Hastad, Improved lower bounds for small depth circuits, *Proc. 18th ACM STOC*, Berkeley CA 1986

[Le] C. Y. Lee, Representation of switching functions by binary decision programs, *Bell Syst. Tech. Journal* **38** (1959), 985-999.

[Lo] L. Lovász, *Combinatorial Problems and Exercises*, North-Holland 1979.

[Ma] W. Masek, A fast algorithm for the string editing problem and decision graph complexity, M.Sc. Thesis, MIT 1976

[Ne] E. I. Nečiporuk, On a Boolean function, *Dokl. Akad. Nauk SSSR* **169** No. 4 (1966), 765-766. English translation: *Soviet Math. Doklady* **7** (1966), pp. 999-1000.

[Pu] P. Pudlák, A lower bound on complexity of branching programs, *Proc. Conf. on the Mathematical Foundations of Computer Science 1984*, Springer Lecture Notes in Computer Science **176** (1984), 480–489.

[Ra1] A. A. Razborov, Lower bounds for the monotone complexity of some Boolean functions (in Russian), *Dokl. Akad. Nauk SSSR* **281** (1985), 798-801.

[Ra2] A. A. Razborov, A lower bound for the monotone network complexity of the logical permanent (in Russian), *Matematicheskie Zametki* **37/6** (1985)

[Sa] J. E. Savage, *The Complexity of Computing*, Wiley 1976

[We] I. Wegener, On the complexity of branching programs and decision trees for clique functions, Universität Frankfurt, Fachbereich Informatik, Int. Rept. 5/84, 1984

[Ya1] A. C. Yao, Lower bounds by probabilistic arguments, *Proc. 24th IEEE FOCS*, 1983, pp. 420–428.

[Ya2] A. C. Yao, Separating the polynomial-time hierarchy by oracles, *Proc. 26th IEEE FOCS*, Portland OR 1985, pp. 1–10.

[Za] S. Zák, An exponential lower bound for one-time-only branching programs, *Proc. Conf. on Mathematical Foundations of Computer Science 1984*, Springer Lecture Notes in Computer Science **176** (1984), 562-566.