JOURNAL OF COMPUTER AND SYSTEM SCIENCES 18, 86–91 (1979)

On the Complexity of Computations under Varying Sets of Primitives*

DAVID P. DOBKIN AND RICHARD J. LIPTON†

Department of Computer Science, Yale University, New Haven, Connecticut 06520

Received December 15, 1975; revised July 9, 1978

1. INTRODUCTION

The goal of the current research is the study of lower bounds on the complexity of a set of searching problems under various restrictions on the nature of the primitive operation used to determine each branch in a search tree. The model, first studied by Rabin [6], Reingold [7], and Spira [9] and to be described in more detail in the next section, has programs consisting of two types of statements. Query statements are of the form

$$L_k\text{: } \textit{if } f(x) \mathscr{R}\ 0 \textit{ then goto } L_m \textit{ else goto } L_n$$

where $\mathscr{R}$ is one of the relations ($>$ or $=$) and f is a function of restricted form on the input of x. An output statement of the form

$$L_s\text{: } \textit{accept} \text{ (or } \textit{reject}\text{)}$$

occurs for each possible outcome of the problem.

The problems we consider all involve searching a set of geometric objects in Euclidean space to determine in which region of their partition of space a given point lies or whether the point lies in any of the given regions. Among the new results obtained are exponential lower bounds on searching for solutions to a knapsack problem, viewed as a hyperplane search problem, using various models involving restrictions on the primitive operations allowed. A nonlinear (in the number of hyperplanes) lower bound is given for a generalized hyperplane search problem along with an $O(n \log n)$ bound for a problem in the plane.

2. BASIC MODEL

The model of computation used here is based on the notion of a *search program.* A search program P with input $(x_1, ..., x_n)$ is a finite list of instructions of the following three types:

* An earlier version of this paper was presented at the 2nd GI-Professional Conference of Automata Theory and Formal Language, at Kaiserslautern, Germany, in May 1975.

† The first author's research was supported in part by the Office of Naval Research under Grant N00014-75-C-0450, the second author's in part by National Science Foundation Grant MCS76-81486.

(1) L_k: *if* $f(x_1, \ldots, x_n) \mathcal{R}\, 0$ *then goto* L_m ($\mathcal{R} \in \{>, =\}$
else goto L_p

(2) L_k: *accept*

(3) L_k: *reject.*

Control initially starts at the first instruction. An instruction of type (1) determines whether the indicated test is true: if it is true, control passes to the statement with label L_m; otherwise, control passes to the statement with label L_p. An instruction of type (2) denotes that the program has halted and it has accepted the input. Correspondingly, an instruction of type (3) denotes that the program has halted and it has rejected the input.

We will restrict search programs in two ways. The functions allowed in instructions of type (1) are called *primitives.* Often we will restrict the class of allowed primitives. We will also restrict at times the relations $\mathcal{R}$ allowed in instructions of type (1). Thus an equality search program can have $\mathcal{R}$ equal to $\{=\}$. On the other hand, a linear search program can have only functions f that are linear.

The complexity measure we will use on our search programs is "time." Each possible input $(x_1, \ldots, x_n)$ determines a computation through the search program. The length of this computation is the number of steps associated with the input $(x_1, \ldots, x_n)$. We are always interested in the worst-case behavior, i.e. the maximum number of steps required by a given search program.

3. Restricted Linear Programs

In this section we will investigate the n-dimensional knapsack problem (KS_n). We can view this problem as follows: Given a point $(x_1, \ldots, x_n, b) \in E^{n+1}$ we are to determine whether there exists an index set I such that

$$\sum_{i \in I} x_i - b = 0.$$

The first question we ask is: if we restrict our search programs to queries of the form

$$\sum_{i \in I} x_i \gtreqless b$$

can we show that they must require an exponential number of queries? The answer is yes:

Theorem 1. *Any search program having as its primitive operation functions of the form*

$$\sum_{i \in I} x_i - b$$

for some index set I and any tests $>$, $=$, and $<$ must require $O(2^n)$ primitive steps to solve the n-dimensional Knapsack Problem.

Proof. We adopt an adversary approach and provide a set of data such that if fewer than $\binom{n}{n/2}$ primitive operations are executed the data can be altered so as to make it possible for the solution of the problem to change without changing previous results.

Our adversary will return answers to queries according to the following plan:

(i) if $|I| < n/2$, *then* $\sum_{i \in I} x_i < b$

(ii) if $|I| > n/2$, then $\sum_{i \in I} x_i > b$

(iii) if $|I| = n/2$ and fewer than $\binom{n}{n/2} - 1$ tests on index sets of exactly $n/2$ elements have been done, then $\sum_{i \in I} x_i > b$.

Suppose under this adversary strategy there is a query $\sum_{i \in I} x_i - b$ with $|I| = n/2$ which is not tested. Then, we might be dealing with either of the following two sets of data (where $0 < \epsilon < (1/n^2)b$).

(i) $x_i = (2/n)b + \epsilon, \forall_i$;

(ii) $x_i = (2/n)b$ for $i \in I$ and $x_i = (2/n)b + \epsilon$, otherwise.

Clearly (i) should yield a "no" answer while (ii) a "yes" answer to the problem. A contradiction.

The result of this theorem is that any polynomial-time algorithm for solving the knapsack problem must use comparisons to hyperplanes not in the original set but generated from the original set. While such an algorithm is possible, it is unlikely to exist as a general procedure but might rather exist as a set of procedures $\{P_i\}_{i=1}^{\infty}$ such that solving the n-dimensional knapsack problem involves using procedure P_n to generate new hyperplanes and solving the $n+1$-dimensional knapsack problem involves using (possibly different) procedure P_{n+1} to generate new hyperplanes.

4. Linear Programs

Next we will study linear programs. That is, we will allow any test of the form

$$f(x_1, \ldots, x_n) \gtreqqless 0$$

where f is a linear function. The next theorem allows us to obtain lower bounds for the complexity of various membership problems.

THEOREM 2. *Any linear search tree that solves the membership problem for a disjoint union of a family $\{A_i\}_{i \in I}$ of open subsets of R^n requires at least* $\log_2 |I|$ *queries in the worst case.*

Proof. We prove that any such search tree T with leaves $D_1, \ldots, D_r$ has $r \geqslant |I|$ and hence a path of depth $\geqslant \log_2 |I|$. The leaves partition R^n and, for each j, D_j is an accepting leaf if $D_j \subseteq \bigcup_{i \in I} A_i$ and a rejecting leaf otherwise. The theorem now follows from the observation that for each l, $1 \leqslant l \leqslant r$, there is at most one i such that $D_l \cap A_i$ is nonempty. If we choose points $x \in D_l \cap A_i$ and $y \in D_l \cap A_j$ then all points on the line joining x and y belong to D_l by the convexity of D_l. However, since A_i and A_j are disjoint, there is a point on this line that does not belong to $\bigcup_{i \in I} A_i$. Hence D_l can be neither an accepting nor a rejecting leaf and the theorem holds. ∎

Let us now generalize the knapsack problem (KS_n) to the generalized knapsack problem (GKS_n): we are given 2^n hyperplanes $H_1, \dots, H_{2^n}$ in E^{n+1} space that form a simple arrangement, i.e. no $n+2$ hyperplanes have a common point. For each new point x we are to determine whether x lies in any of these hyperplanes. Note that we do *not* insist that the search tree determine which hyperplane x lies in; it must determine only whether or not x lies in some hyperplane.

From this result we obtain the following corollaries:

COROLLARY 1. *The membership problem for GKS_n takes at least $O(n^2)$ queries for any search tree.*

Proof. Since the hyperplanes of this problem form a simple arrangement, we can find a family $\{A_i\}_{i\in I}$ of open subsets of R^n such that

$$x \in \bigcup_{i\in I} A_i \leftrightarrow x \notin \mathrm{GKS}_n$$

and $|I| \geqslant O(2^{n^2})$ [4]. The corollary then follows from the theorem. ▮

This result improves a lower bound of $O(n)$ due to Spira [9]. Further extensions of this result appear in [1, 3]. In [3] it is shown that the KS_n problem requires $\frac{1}{2}n^2$ queries in any search tree.

COROLLARY 2 (Element Uniqueness Problem). *Let E_n be the set of points in R^n that have two coordinates equal; then any algorithm for determining membership in E_n requires at least $O(n \log n)$ queries.*

Proof. Solving the membership problem of E_n corresponds to solving the membership problem for the family

$$\bigcup_{\pi\in S_n} \{A_\pi\}$$

where

$$A_\pi = \{(x_1, \dots, x_n) \in R^n \mid x_{\pi(1)} < x_{\pi(2)} < \cdots < x_{\pi(n)}\}$$

and S_n is the set of permutations on n objects. The result then follows from $|S_n| = n!$. ▮

Applications of this result may be found in [8].

5. EQUALITY PROGRAMS

In the previous section, we considered the problem of determining whether a point belonged to the union of a family of open sets allowing linear search programs. Here, we extend our methodology to the problem of determining whether a point belongs to the union of a family of varieties allowing search programs that determine at each step whether the point is the root of an irreducible polynomial. Before proceeding, we state some results from algebraic geometry [5] that will be necessary to our development.

DEFINITION. A *variety* $V(f_1,\dots,f_m)$ is a subset of R^n defined by $V(f_1,\dots,f_m) = \{(x_1,\dots,x_n) \in R^n \mid f_1(x_1,\dots,x_n) = \cdots = f_m(x_1,\dots,x_n)\} = 0$ for polynomials $f_1,\dots,f_m$.

DEFINITION. The polynomials f and g are said to be *equivalent* iff there exists a non-zero constant λ such that $f = \lambda g$.

Fact 1. If the dimension of $V(f_1,\dots,f_n)$ is denoted by $\dim(V(f_1,\dots,f_n))$ then

(i) $\dim(A) = 0$ if and only if A is empty;

(ii) if $R^n = \bigcup_{i=1}^{k} V(f_i)$, then one of the polynomials f_i is trivial;

(iii) if f and g are non-trivial irreducible polynomials that are not equivalent, then $\dim(V(f, g)) < \dim(V(f))$;

(iv) $\dim(\bigcup_j V(h_j)) = \max_j \dim(V(h_j))$.

THEOREM 3. *If $f_1,\dots,f_m$ are irreducible polynomials of n real variables that are not equivalent, tuen any equality searrh program for*

$$\bigcup_{i=1}^{m} V(f_i)$$

using only irreducible polynomials requires at least m queries.

Proof. Let T be a search program of depth k that determines for any $x \in R^n$ whether or not $x \in \bigcup_{i=1}^{m} V(f_i)$. Select the path in T that always takes the NO branch; moreover, let $g_1(x) = 0,\dots, g_l(x) = 0$ be the queries on this path ($l \leqslant k$). Define

$$F = V(f_1) \cup \cdots \cup V(f_m)$$

and, for $1 \leqslant i \leqslant l$, $G_i = [V(g_1) \cup \cdots \cup V(g_i)]^c$ (where A^c = the complement of the set A). We now assert that if $l < m$ then

(1) $G_l \cap F \neq \phi$ and

(2) $G_l \cap F^c \neq \phi$.

This will be a contradiction since $x \in G_k$ implies that x takes this path; hence, whether the leaf of this path is an ACCEPT or a REJECT we have a contradiction with either (1) or (2).

If (1) is false, i.e. if $G_l \cap F = \phi$, then there is some i such that for all j, $V(f_i) \neq V(g_j)$. This follows since $l < m$. Fix this i. Then $G_l \cap V(f_i) = \phi$, and hence $V(g_1) \cup \cdots \cup V(g_l) \cup V(f_i)^c = R^n$. Thus $(V(g_1) \cap V(f_i)) \cup \cdots \cup (V(g_l) \cap V(f_i)) = V(f_i)$. But $\dim(V(f_i)) = \max_{1 \leqslant j \leqslant l} \dim(V(g_j) \cap V(f_i))$ and $\dim(V(g_j) \cap (V(f_i)) < \dim(V(f_i))$ by facts (iii) and (iv). A contradiction.

Now assume that (2) is false, i.e. that $G_l \cap F^c = \phi$. This is equivalent to

$$V(g_1) \cup \cdots \cup V(g_l) \cup V(f_1) \cup \cdots \cup V(f_m) = R^n$$

which is impossible by Fact (ii). Hence $l \geqslant m$. ▌

COROLLARY 3. *Any equality search program for KS_n that uses only irreducible polynomials requires at least 2^n queries.*

REFERENCES

1. D. DOBKIN, A nonlinear lower bound on linear search tree programs for solving knapsack problems, *J. Comput. System Sci.* **13** (1976), 69–73.
2. D. DOBKIN AND R. J. LIPTON, On some generalizations of binary search, *ACM Symposium on the Theory of Computing*, Seattle, Wash., May 1974.
3. D. DOBKIN AND R. J. LIPTON, A lower bound of $\frac{1}{2}n^2$ on linear search programs for the knapsack problem, *J. Comput. System Sci.* **16** (1978), 413–417.
4. B. GRÜNBAUM, "Convex Polytopes," Interscience, New York, 1967.
5. S. LEFSCHETZ, "Algebraic Geometry," Princeton Univ. Press, Princeton, N.J., 1953.
6. M. RABIN, Proving the simultaneous positivity of linear forms, *J. Comput. System Sci.* **13** (1972), 639–650.
7. E. REINGOLD, Computing the maximum and the median, 12*th Annual Symposium on Switching and Automata Theory* (1971), pp. 216–218.