

## A LOWER BOUND ON THE COMPLEXITY OF THE UNION-SPLIT-FIND PROBLEM\*

KURT MEHLHORN<sup>†</sup>, STEFAN NÄHER<sup>†</sup>, AND HELMUT ALT<sup>‡</sup>

**Abstract.** We prove a  $\Theta(\log \log n)$  (i.e., matching upper and lower) bound on the complexity of the Union-Split-Find problem, a variant of the Union-Find problem. Our lower bound holds for **all** pointer machine algorithms and does not require the separation assumption used in the lower-bound arguments of Tarjan [*J. Comput. Systems Sci.*, 18 (1979), pp. 110-127] and Blum [*SIAM J. Comput.*, 15 (1986), pp. 1021-1024]. We complement this with a  $\Theta(\log n)$  bound for the Split-Find problem under the separation assumption. This shows that the separation assumption can imply an exponential loss in efficiency.

**Key words.** complexity, interval splitting, lower bound, pointer machine, priority queue, union

**AMS(MOS) subject classification.** 68

**1. Introduction.** We consider the following three operations on a linear list  $x_1, x_2, \dots, x_n$  of items, some of which are marked:

UNION ( $x_i$ ): given a pointer to the marked item  $x_i$  unmark this item;

SPLIT ( $x_i$ ): given a pointer to the unmarked item  $x_i$  mark this item;

FIND ( $x_i$ ): given a pointer to the item  $x_i$  return a pointer to  $x_j$ ,  
where  $j = \min \{l \mid l \geq i \text{ and } x_l \text{ is marked}\}$ .

Note that the marked items partition the linear list  $x_1, \dots, x_n$  into intervals of unmarked items. Then FIND ( $x_i$ ) returns (a pointer to) the right endpoint of the interval containing  $x_i$ , SPLIT ( $x_i$ ) splits the interval containing  $x_i$ , and UNION ( $x_i$ ) joins the two intervals having  $x_i$  as a common endpoint. We call the problem above the **Union-Split-Find problem**; P. v. Emde Boas et al. [EKZ77] called it a priority queue problem. They referred to the three operations as Insert, Delete, and Successor, and exhibited an  $O(\log \log n)$  solution for it. We will also consider the **Split-Find problem** and the **Union-Find problem** (only operations split, find and union, find, respectively). Note that our Union-Find problem is a restriction of the usual Union-Find problem (here called the general Union-Find problem) because we allow only adjacent intervals to be joined. The Union-Split-Find problem is important for a number of applications, e.g., dynamic fractional cascading [MN86] and computing shortest paths [M84b, p. 47].

We study the complexity of the Union-Split-Find problem in the pointer machine model of computation (Kolmogorov [Ko53], Knuth [Kn68], Schönhage [S73], Tarjan [T79]). A pointer machine captures the list-processing capabilities of computers; its storage consists of records connected by pointers. Previously, lower bounds in a restricted pointer machine model (the term "restricted" is explained below) were obtained by Tarjan [T79] and Blum [B86]. Tarjan proved a  $\Theta(\alpha(n))$  bound on the amortized cost of the general Union-Find problem and Blum proved a  $\Theta(\log n / \log \log n)$  bound on the worst-case cost. Tarjan's and Blum's lower bounds rely heavily on the following **separation assumption** (quoted from Tarjan [T79]):

At any time during the computation, the contents of the memory can be partitioned into collections of records such that each collection corresponds to a currently existing

\* Received by the editors December 31, 1986; accepted for publication (in revised form) December 9, 1987. This research was supported by the Deutsche Forschungsgemeinschaft, grant SPP ME 620/6-1.

<sup>†</sup> FB10, Informatik Universität des Saarlandes, D-6600 Saarbrücken, Federal Republic of Germany.

<sup>‡</sup> FB Mathematik, WE3, Freie Universität Berlin, D-1000 Berlin 33, Federal Republic of Germany.

set,  $\dots$  and no record in one collection contains a pointer to a record in another collection.

Because of this assumption we call their model **restricted**. If we view pointers as undirected edges in a graph, then the separation assumption states that every currently existing set corresponds to a component of the graph.

The main results of this paper are as follows.

(1) The complexity of the Union-Split-Find problem in the pointer machine model is  $\Theta(\log \log n)$ . Here, the upper bound is on the worst-case cost of the three operations and the lower bound is on the amortized cost of the three operations, i.e., there are arbitrarily large  $m$  and sequences of  $m$  SPLIT, FIND, and UNION operations having a total cost of  $\Omega(m \log \log n)$ .

The upper bound can be found in [EKZ77], [Ka84], and [MN86]. The solution of [MN86] supports two additional operations ADD and ERASE which allow us to modify the underlying linear list; the solution does not satisfy the separation assumption. The lower bound will be proved in § 2 of this paper.

(2) In the restricted pointer machine model the worst-case complexity of the Split-Find problem is  $\Theta(\log n)$  and the amortized complexity of the Union-Split-Find problem is  $\Theta(\log n)$ .

This will be shown in § 3.

**2. The lower bound.** In this section we will show that each solution for the Union-Split-Find problem on a pointer machine requires  $\Omega(\log \log n)$  computational steps, even in the amortized sense. More precisely, we show that there are arbitrarily large  $m$  and sequences of  $m$  UNION, FIND, SPLIT operations having a total cost of  $\Omega(m \log \log n)$ .

Our machine model is a pointer machine as described in [T79]. Its memory  $M$  consists of an unbounded collection of records, each containing two pointers to other records and an arbitrary amount of additional information. Thus  $M$  can be regarded as a directed graph with outdegree 2. We assume that the set of items  $S$  is realized by two sets of records in  $M$ , a set of input records  $I = \{x_1^*, x_2^*, \dots, x_n^*\}$ , and a set of output records  $O = \{y_1^*, y_2^*, \dots, y_n^*\}$ . (The distinction between input and output records is merely a notational convenience.)

FIND( $x$ ) is executed as follows. The machine starts with a pointer to input Record  $x^*$  (corresponding to item  $x \in S$ ) in some register  $r$ . It stops with a pointer to the output record  $y^*$  in  $r$  which corresponds to item  $y = \text{FIND}(x)$ . Since it can access records in  $M$  only by pointers contained in one of its registers, there must be a path  $p$  formed by records and pointers in  $M$  starting in  $x^*$  and ending in  $y^*$  such that pointers to all records on  $p$  have been loaded into a register during the execution of the FIND. In addition to traversing a path from  $x^*$  to  $y^*$  the FIND operation may change some number of pointers. The cost of the FIND operation is certainly bounded from below by the length of a shortest path from  $x^*$  to  $y^*$  in  $M$  plus the number of pointers changed.

When executing the operation SPLIT( $y$ ) the machine starts with a pointer to record  $y^*$  in some register. After the operation record  $y^*$  is reachable via pointers in  $M$  for all records  $x^*$  corresponding to items  $x$  with  $\text{FIND}(x) = y$ . To achieve this the machine changes certain pointers in  $M$ . The number of pointers changed is the cost of SPLIT( $y$ ). The cost of UNION( $y$ ) is defined similarly.

In order to model the actions of the pointer machine on its memory  $M$  in a more abstract way we consider FIND and SPLIT as operations on a directed graph  $G$  as follows:

Let  $G = (V, E)$  be a directed graph with the following:

- (1) For all  $v \in V$ :  $\text{outdegree}(v) = 2$ .
- (2) There is a set of  $n$  input nodes  $I = \{x_1, x_2, \dots, x_n\} \subseteq V$ .
- (3) There is a set of  $n$  output nodes  $O = \{y_1, y_2, \dots, y_n\} \subseteq V - I$  which may be marked (we call  $y_i$  the output node corresponding to input node  $x_i$ ,  $1 \leq i \leq n$ ).

FIND ( $x_i$ ),  $x_i \in I$ , returns output node  $y_j$  such that  $j \geq i$ , and  $j$  is minimal with  $y_j$  marked. In addition, it may replace some edges of  $G$  by new edges. The complexity of FIND ( $x_i$ ) is the length of the shortest path from  $x_i$  to  $y_j$  in  $G$  plus the number of edges replaced.

SPLIT ( $y$ ),  $y \in O$ , marks output node  $y$  and replaces some edges of  $G$  by new edges such that every marked output node  $y$  is reachable from all input nodes  $x$  with  $\text{FIND}(x) = y$ . The complexity of SPLIT ( $y$ ) is the number of edges replaced.

UNION ( $y$ ),  $y \in O$ , unmarks output node  $y$  and replaces some edges of  $G$  with new edges. The complexity of UNION ( $y$ ) is the number of edges replaced.

LEMMA 1. *Let  $k$  be any integer, let  $L = 2^{2k+1} + 1$ , let  $n \geq 2^{(5k)2^k}$ , and let  $G_0 = (V, E)$  be a directed graph with all output nodes unmarked. Then there is a sequence of  $L$  SPLIT operations followed by  $L$  FIND operations, followed by  $L$  UNION operations such that we have the following:*

- (1) *The total cost of the sequence is at least*

$$\min \left( k \cdot L, \frac{1}{2^{2k+4}} n^{1/2^k} \right).$$

- (2) *All output nodes are unmarked after executing the sequence.*

*Proof.* We call a graph  $G$  a  **$k$ -structure** if and only if for every input node  $x$  there is a path of length at most  $k$  from  $x$  to the output node  $y = \text{FIND}(x)$ .

The idea of the proof is as follows. We first execute a sequence of  $L$  SPLIT instructions (this sequence is constructed below) which leaves us with a data structure  $G_1$ . We next execute a hardest FIND on  $G_1$ , i.e., an operation FIND ( $x$ ) where the shortest path from  $x$  to FIND ( $x$ ) has maximal length. Execution of this FIND instruction yields the data structure  $G_2$ . Again we perform a hardest FIND,  $\dots$ . In this way we perform a total of  $L$  FINDs, each FIND being a hardest FIND in the present data structure. Finally, we perform a sequence of UNIONS which undo all the SPLITS. This yields a data structure  $G'_0$  (generally different from  $G_0$ ) in which all output nodes are unmarked.

In order to estimate the cost of this sequence of instructions we distinguish two cases. Assume first that no  $G_i$ ,  $1 \leq i \leq L$ , is a  $k$ -structure. Then each FIND costs at least  $k$  time units for a total cost of  $k \cdot L$  time units. Assume next that some  $G_i$  is a  $k$ -structure. We will show that it takes at least  $(1/2^{2k+4}) \cdot n^{1/2^k}$  edge changes to construct  $G_i$  from  $G_0$ ; more precisely, we show the following claim.

CLAIM. *Let  $G_0$  be an arbitrary data structure with all output nodes unmarked. Then there is a set  $\{z_1, \dots, z_L\} \subseteq O$  of output nodes such that no  $k$ -structure  $G'$  with exactly  $z_1, \dots, z_L$  marked can be constructed from  $G_0$  with fewer than  $(1/2^{2k+4}) \cdot n^{1/2^k}$  edge replacements.*

*Proof.* For all  $v \in V$  and  $l \geq 1$  let  $R_l(v)$  denote the set of all input nodes  $w$  such that  $v$  is reachable from  $w$  on a path of length at most  $l$ .

Let  $c_0, c_1, \dots, c_{2^k}$  be a sequence of positive constants with

$$n = c_0 \geq c_1 \geq \dots \geq c_{2^k-1} \geq c_{2^k} = 1.$$

Then the following lemma holds for  $G_0$ .

LEMMA 2. *There is an  $l$ ,  $0 \leq l \leq 2^k - 1$ , and  $l$  distinct nodes  $v_1, \dots, v_l$  such that we have the following:*

- (1) *Either  $l = 0$ , or  $|\bigcap_{1 \leq i \leq l} R_{k-1}(v_i)| \geq c_l$ .*

(2) For each  $w \in V - \{v_1, \dots, v_l\}$ :  $|I^* \cap R_{k-1}(w)| < c_{l+1}$ , where  $I^* = \bigcap_{1 \leq i \leq l} R_{k-1}(v_i)$  ( $I^* = I$  if  $l = 0$ ).

*Proof.* Let  $l_0$  be maximal such that there are distinct nodes  $v_1, \dots, v_{l_0}$  with  $|\bigcap_{1 \leq i \leq l_0} R_{k-1}(v_i)| > c_{l_0}$ . We only have to show that  $l_0 < 2^k$ . Assume  $l_0 \geq 2^k$ . Then there are  $2^k$  distinct nodes  $v_1, \dots, v_{2^k}$  with  $\bigcap_{1 \leq i \leq 2^k} R_{k-1}(v_i) \neq \emptyset$ . But for every  $x \in I$  there are at most  $2^k - 1$  nodes  $v \in V$  with  $x \in R_{k-1}(v)$ , a contradiction.  $\square$

Now let  $l \geq 0$  and let  $v_1, v_2, \dots, v_l$  be nodes satisfying the conditions of Lemma 2. We give a sequence of  $2^{2k+1} + 1$  SPLIT operations which require at least  $(c_l/2L - 2^{k+1})/c_{l+1}$  edge replacements. First we divide  $I^* = \bigcap_{1 \leq i \leq l} R_{k-1}(v_i)$  into  $2L$  intervals  $A_1, A_2, \dots, A_{2L}$  of about equal size.  $A_1$  consists of the  $\lfloor I^*/2L \rfloor$  smallest indexed items in  $I^*$ ,  $A_2$  consists of the next  $\lfloor I^*/2L \rfloor$  smallest indexed items in  $I^*$ , and so forth. Clearly,

$$|A_i| \geq \frac{c_l}{2L} - 1 \quad \text{for all } i.$$

Let  $A'_1, A'_2, \dots, A'_{2L}$  be the corresponding intervals of output nodes.

LEMMA 3. In every  $A'_j$ ,  $j \in \{2, 4, 6, \dots, 2L\}$ , there is an output node  $y_j$  that is reachable in  $G_0$  from at most  $2^{k+1}$  input nodes in  $A_{j-1}$  on a path of length  $\leq k$ .

*Proof.* For all  $x \in A_{j-1}$  we have  $|\{v \in V \mid x \in R_k(v)\}| \leq 2^{k+1} - 1$ . Thus we conclude

$$\begin{aligned} \sum_{x \in A_{j-1}} |\{v \in V \mid x \in R_k(v)\}| &\leq |A_{j-1}| \cdot (2^{k+1} - 1) \\ &= |A'_j| \cdot (2^{k+1} - 1). \end{aligned} \quad \square$$

Let  $y_j \in A'_j$ ,  $j \in \{2, 4, \dots, 2L\}$  be defined as in Lemma 3. Then there exists for each  $j$  a set  $B_{j-1} \subseteq A_{j-1}$  such that  $|B_{j-1}| \geq |A_j| - 2^{k+1}$  and such that for each  $x \in B_{j-1}$  there is no path of length at most  $k$  from  $x$  to  $y_j$  in  $G_0$ . Next consider any  $k$ -structure  $G'$  in which exactly the  $y_j$ 's defined above are marked. In  $G'$  there must be a path of length at most  $k$  from any  $x \in B_{j-1}$  to  $y_j$ . Let  $p(x)$  be any such path from  $x$  to  $y_j$  and let  $e(x)$  denote the first new edge on  $p(x)$ . Let the edge  $e(x)$  start in vertex  $v(x)$ . We have the following lemma.

LEMMA 4. (1)  $|\{x \in B_{j-1} \mid v(x) = w\}| \leq c_{l+1}$  for every node  $w \notin \{v_1, v_2, \dots, v_l\}$  and every  $j$ .

(2) There is a  $j \in \{2, 4, 6, \dots, 2L\}$  such that  $v(x) \notin \{v_1, v_2, \dots, v_l\}$  for all  $x \in B_{j-1}$ .

*Proof.* (1) If  $v(x) = w$  then  $w$  was reachable from  $x$  in the data structure  $G_0$  in at most  $k - 1$  steps. Thus the claim follows immediately from the definition of  $v_1, \dots, v_l$ .

(2) For every  $v \in \{v_1, v_2, \dots, v_l\}$  we have  $|\{y \in O \mid v \in R_k(y)\}| \leq 2^{k+1}$  and thus

$$\begin{aligned} \sum_{v \in \{v_1, \dots, v_l\}} |\{y \in O \mid v \in R_k(y)\}| &\leq l2^{k+1} \\ &\leq 2^{2k+1} \quad \text{since } l \leq 2^k. \end{aligned}$$

That is, all paths of length at most  $k$  using nodes from  $\{v_1, \dots, v_l\}$  cannot lead to more than  $2^{2k+1} = L - 1$  output nodes. Since we marked  $L$  nodes in  $O$  there must be one  $y_j$  such that  $v(x) \notin \{v_1, \dots, v_l\}$  for all  $x \in B_{j-1}$ .  $\square$

Now consider any  $j \in \{2, 4, \dots, 2L\}$  satisfying condition (2) of Lemma 4; i.e.,  $v(x) \notin \{v_1, \dots, v_l\}$  for all  $x \in B_{j-1}$ . Next observe that by (1) of Lemma 4 for any node  $w \notin \{v_1, \dots, v_l\}$  there are at most  $c_{l+1}$  nodes  $x \in B_{j-1}$  with  $v(x) = w$ . Thus

$$\frac{|B_{j-1}|}{c_{l+1}} = \frac{|A_j| - 2^{k+1}}{c_{l+1}} \geq \frac{c_l/2L - 1 - 2^{k+1}}{c_{l+1}} \geq \frac{c_l/(2^{2k+2} + 2) - 1 - 2^{k+1}}{c_{l+1}}$$

edges must be changed to obtain  $G'$  from  $G_0$ . With

$$c_i = n^{1-i/2^k}, \quad 0 \leq i \leq 2^k$$

we conclude further that

$$\begin{aligned} \frac{c_i / (2^{2k+2} + 2) - (2^{k+1} + 1)}{c_{i+1}} &\geq \frac{n^{1/2^k}}{2^{2k+2} + 2} - (2^{k+1} + 1) \\ &\geq \frac{n^{1/2^k}}{2 \cdot (2^{2k+2} + 2)} \quad \text{since } n \geq 2^{(5k)2^k} \\ &\geq \frac{n^{1/2^k}}{2^{2k+4}}. \end{aligned}$$

This completes the proof of the claim and of Lemma 1.  $\square$

**THEOREM 1.** (a) *The single operation worst-case complexity of the Split-Find problem is  $\Omega(\log \log n)$ .*

(b) *The amortized complexity of the Union-Split-Find problem is  $\Omega(\log \log n)$ ; i.e., there are arbitrarily large  $m$  and sequences of  $m$  UNION, SPLIT, and FIND operations having a total cost of  $\Omega(m \log \log n)$ .*

*Proof.* (a) Let  $k$  be maximal such that  $n \geq 2^{(5k)2^k}$ . Then  $k = \Omega(\log \log n)$  and

$$\begin{aligned} \frac{1}{2^{2k+1} + 1} \cdot \frac{1}{2^{2k+4}} n^{1/2^k} &\geq \frac{2^{5k}}{2^{2k+4}(2^{2k+1} + 1)} \\ &= \Omega(2^k) \\ &= \Omega(\log n) \\ &= \Omega(\log \log n). \end{aligned}$$

Consider the sequence of  $2L$  SPLITS and FINDS constructed in the proof of Lemma 1. The cost of this sequence is

$$\Omega\left(\min\left(k \cdot L, \frac{1}{2^{2k+4}} \cdot n^{1/2^k}\right)\right) = \Omega(L \cdot \log \log n),$$

and hence at least one operation of the sequence has cost  $\Omega(\log \log n)$ .

(b) Let  $k$  be maximal such that  $n \geq 2^{(5k)2^k}$  and let  $m = 3 \cdot L \cdot s$  for some  $s$ . Let  $G_0$  be any data structure with all output nodes unmarked. By part (a) and Lemma 1 there is a sequence of  $3 \cdot L$  instructions such that

- (1) the total cost of the sequence is  $\Omega(L \log \log n)$ ;
- (2) all output nodes are unmarked after executing the sequence.

Thus again by part (a) and Lemma 1 there is another sequence of  $3 \cdot L$  instructions  $\dots$ . Using this argument  $s$  times yields a sequence of  $m$  instructions with total cost  $\Omega(m \log \log n)$ .  $\square$

**3. A lower bound in the restricted model.** In this section we prove a  $\Omega(\log n)$  lower bound for the worst-case complexity of the Split-Find problem and a  $\Omega(\log n)$  lower bound for the amortized complexity of the Union-Split-Find problem in the restricted pointer machine model. We consider pointer machine algorithms satisfying the following **separation condition**:

The memory can be partitioned into subgraphs such that each subgraph corresponds exactly to a current interval. There exists no edge from a node in such a subgraph to a node outside the subgraph [B86], [T79].

As in the previous section we define a  $k$ -structure.

DEFINITION. Let  $G = (V, E)$  be a directed graph with input nodes  $I = \{x_1, \dots, x_n\}$  and output nodes  $O = \{y_1, \dots, y_n\}$  some of which may be marked.  $G$  is called a  $k$ -structure if and only if for every  $x \in I$  there is a path of length at most  $k$  to  $\text{FIND}(x)$  and  $G$  fulfills the separation condition; i.e., for every  $x_i, x_j \in I$  with  $\text{FIND}(x_i) \neq \text{FIND}(x_j)$  there is no  $v \in V$  that is reachable from both  $x_i$  and  $x_j$ .  $\square$

LEMMA 5. Let  $n \geq (4k)^k/k!$ . In any  $k$ -structure with output nodes  $y_1, \dots, y_{n-1}$  unmarked and  $y_n$  marked satisfying the separation condition there is a SPLIT operation requiring at least  $k/12 \cdot n^{1/k}$  edge replacements.

Proof. Since  $y_n$  is the only marked output node there is, for each input node  $x$ , a path of length at most  $k$  to  $y_n$ . Let  $T$  be the subgraph of  $G$  formed by all shortest paths from input nodes to  $y_n$ . Then  $T$  is a tree with root  $y_n$ , leaves  $\{x_1, \dots, x_n\}$ , and height at most  $k$ . For any node  $v$  in  $T$  let  $I(v)$  denote the set of input nodes which are the leaves of the subtree rooted at  $v$ . At each internal node  $v$  of  $T$  the incoming edges can be ordered  $(w_1, v), (w_2, v), \dots, (w_m, v)$  such that the minimal index of any input node in  $I(w_i)$  is smaller than the minimal index of any input node in  $I(w_j)$  for all  $1 \leq i < j \leq m$ . (We call  $w_i$  the  $i$ th child of  $v$ .)

Now assume that  $\text{SPLIT}(x)$  is executed for some  $x \in I$ . Let

$$p: x = v_0 \rightarrow v_1 \rightarrow \dots \rightarrow v_l = y_n$$

be the path from  $x$  to the root  $y_n$  in  $T$ . For any  $v_i, 1 \leq i \leq l$ , on this path let  $w_1^i, \dots, w_j^i = v_{i-1}, \dots, w_m^i$  be the children of  $v_i$  in the order defined above. Before the split operation  $v_i$  is reachable from all input nodes of both of the following sets:

$$L_i = \{u \mid u \text{ is the node with minimal index in } I(w_k^i), 1 \leq k \leq j-1\},$$

$$R_i = \{u \mid u \text{ is the node with minimal index in } I(w_k^i), j+1 \leq k \leq m\}.$$

After the split  $\text{FIND}(a) \neq \text{FIND}(b)$  for all  $a \in L_i, b \in R_i$  and by the separation condition  $v_i$  is not reachable from any node of at least one of the sets  $L_i$  or  $R_i$ . Thus the execution of  $\text{SPLIT}(x)$  requires at least

$$\min(|L_i|, |R_i|) = \min\{j-1, \text{indegree}(v_i) - j \mid v_{i-1} \text{ is the } j\text{th child of } v_i\}$$

edge replacements for each node  $v_i, 1 \leq i \leq l$ , on path  $p$ .

The total cost for  $\text{SPLIT}(x)$  is at least

$$c(p) = \sum_{i=1}^l \min\{j-1, \text{indegree}(v_i) - j \mid v_{i-1} \text{ is the } j\text{th child of } v_i\}.$$

We prove a lower bound for  $c(p)$ .

LEMMA 6. Let  $T$  be any tree of height  $k$  with  $n$  leaves and root  $r$ . Then there is a leaf  $x$  in  $T$  such that the path  $p$  from  $x$  to  $r$  has cost

$$c(p) \geq \frac{k}{12} \cdot n^{1/k}.$$

Proof. Define  $l(k, j)$  as the maximal number of leaves in any tree  $T$  of height  $k$ , such that  $c(p) \leq j$  for every path  $p$  from a leaf to the root of  $T$ .

Then we have

$$l(0, j) = 1 \quad \text{for } j \geq 0$$



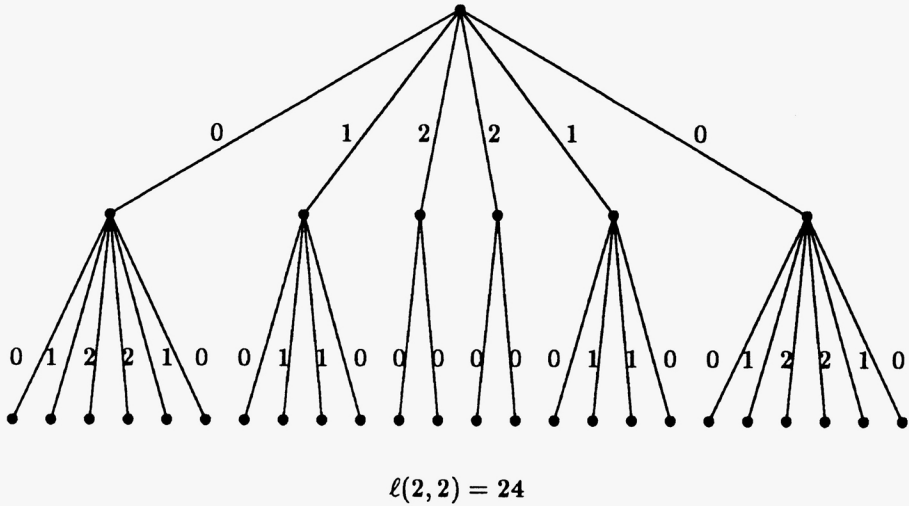


FIG. 2

The term  $2^k$  accounts for the fact that in every vertex each label can be used twice:

$$\begin{aligned}
 l(k, j) &\leq 2^k \cdot \binom{k+j-1}{k-1} \\
 &\leq 2^k \cdot \frac{(k+j-1)^{k-1}}{(k-1)!} \\
 &\leq 2^k \cdot \frac{(k+j-1)^k}{k!} \\
 &\leq 2^k \cdot \frac{(k+j)^k}{k!}.
 \end{aligned}$$

We conclude that in any tree of height  $k$  with  $n$  leaves there exists a path  $p$  from some leaf to the root such that

$$\begin{aligned}
 n &\leq 2^k \cdot \frac{(c(p) + k)^k}{k!} \\
 &\leq 2^k \cdot \frac{(2 \cdot c(p))^k}{k!} \quad \text{since } n \geq \frac{(4k)^k}{k!} \text{ and hence } c(p) \geq k,
 \end{aligned}$$

and finally

$$\begin{aligned}
 c(p) &\geq \frac{1}{2} \left( \frac{n \cdot k!}{2^k} \right)^{1/k} \\
 &\geq \frac{k}{12} \cdot n^{1/k} \quad \text{since } (k!)^{1/k} \geq \frac{k}{3} \text{ if } k \geq 6.
 \end{aligned}$$

This completes the proof of Lemma 5.  $\square$

**THEOREM 2.** *In the restricted pointer machine model we have the following:*

- (a) *The single-operation worst-case complexity of the Split-Find problem is  $\Omega(\log n)$ .*
- (b) *The amortized complexity for the Union-Split-Find problem is  $\Omega(\log n)$ ; i.e., there are arbitrarily large  $m$  and sequences of  $m$  UNION, SPLIT, FIND operations having a total cost of  $\Omega(m \log n)$ .*



*Proof.* (a) Let  $k$  be maximal such that  $n \geq (4k)^k/k!$ . Then  $k = \Omega(\log n)$ . Let  $G$  be any data structure with all output nodes except  $y_n$  unmarked. By Lemma 5 there is either a FIND operation which costs more than  $k$  time units or there is a SPLIT operation having a cost of

$$\frac{k}{12} \cdot n^{1/k} \geq \frac{\log n}{12} \cdot n^{1/\log n} = \frac{\log n}{6} = \Omega(\log n).$$

(b) Let  $G$  be any structure with all output nodes except  $y_n$  unmarked. Define  $k$  as in part (a). If  $G$  is not a  $k$ -structure then we perform a hardest FIND in  $G$ ; this FIND has cost  $k = \Omega(\log n)$  and leaves us with a structure  $G'$  where all output nodes except  $y_n$  are unmarked. If  $G$  is a  $k$ -structure then there is a SPLIT operation of cost  $k/12 \cdot n^{1/k} = \Omega(\log n)$ . We perform it and immediately undo it by the corresponding UNION operation. This leaves us with a structure  $G'$ , where all output nodes except  $y_n$  are unmarked. At this point we are in the initial situation and we have forced the algorithm to spend  $\Omega(\log n)$  time units on at most two operations. Part (b) follows.  $\square$

We want to point out that  $O(\log n)$  is clearly also an upper bound for the complexity of the Union-Split-Find problem in the restricted model. We only have to represent each interval by a balanced tree (cf., e.g., [M84a, § III.5.3.1]).

We close this section with the following corollary.

**COROLLARY 1.** *Pointer machines obeying the separation assumption are exponentially weaker than pointer machines without the separation assumption. This is true for the worst-case complexity and for the amortized complexity.*

*Proof.* This follows immediately from Theorems 1 and 2.  $\square$

**4. Conclusions and open problems.** In this paper we proved several new lower bounds for the Union-Split-Find problem. In particular, we presented an  $\Omega(\log \log n)$  lower bound on the amortized complexity of the Union-Split-Find problem valid for **all** pointer machine algorithms and an  $\Omega(\log n)$  lower bound for restricted (in the sense of Tarjan [T79] and Blum [B86]) pointer machine algorithms. Our lower bounds match known upper bounds. Thus our results reveal that the separation assumption of Tarjan and Blum can imply an exponential loss in efficiency.

Table 2 summarizes all known bounds for the complexity of pointer machine algorithms for the Union-Split-Find problem on intervals including the results of this paper.

There are still several open problems in both models. We have no lower bounds on the amortized complexity of Union-Find and Split-Find and for the worst-case

TABLE 2

Problem	General model		Restricted model	
Union-Find worst case amortized	$O(\log \log n)$ —	[EKZ77]	$O(\log n / \log \log n)$ $\Theta(\alpha(n))$	[B86] [T79]
Split-Find worst case amortized	$\Theta(\log \log n)$ —	<b>new</b>	$\Theta(\log n)$ $O(\log^* n)$	<b>new</b> [HU73]
Union-Split-Find worst case amortized	$\Theta(\log \log n)$ $\Theta(\log \log n)$	<b>new</b> <b>new</b>	$\Theta(\log n)$ $\Theta(\log n)$	<b>new</b> <b>new</b>

complexity of Union-Find in the general model. In the restricted model it remains an open problem to determine whether the Split-Find algorithm of Hopcroft and Ullman [HU73], whose amortized running time is  $O(\log^* n)$ , is optimal. We suppose that Blum's lower bound proof for the general Union-Find problem cannot be modified to work also for the interval problem.

We want to point out that the Union-Find problem and the Split-Find problem considered in this paper have amortized complexity  $\Theta(1)$  on random access machines. This was shown by Gabow and Tarjan [GT83] for the Union-Find problem and by Imai and Asano [IA84] for the Split-Find problem.

## REFERENCES

- [B86] N. BLUM, *On the single-operation worst-case time complexity of the disjoint set union problem*, SIAM J. Comput., 15 (1986), pp. 1021–1024.
- [EKZ77] P. V. EMDE BOAS, R. KAAS, AND E. ZIJLSTRA, *Design and implementation of an efficient priority queue*, Math. Systems Theory, 10 (1977), pp. 99–127.
- [GT83] H. N. GABOW AND R. E. TARJAN, *A linear-time algorithm for a special case of disjoint set union*, in Proc. 15th Annual SIGACT Symposium, 1983, pp. 246–251.
- [IA84] T. IMAI AND T. ASANO, *Dynamic segment intersection with applications*, in Proc. 25th Annual IEEE Symposium on the Foundations of Computer Science, Singer Island, FL, 1984, pp. 393–402.
- [HU73] J. E. HOPCROFT AND J. D. ULLMAN, *Set merging algorithms*, SIAM J. Comput., 2 (1973), pp. 294–304.
- [Ka84] R. G. KARLSSON, *Algorithms in a restricted universe*, Report CS-84-50, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1984.
- [Kn68] D. E. KNUTH, *The Art of Computer Programming*, Vol. 3, Fundamental Algorithms, Addison-Wesley, Reading, MA, 1968.
- [Ko53] A. N. KOLMOGOROV, *On the notion of algorithm*, Uspekhi Mat. Nauk., 8 (1953), pp. 175–176.
- [M84a] K. MEHLHORN, *Data Structures and Algorithms. Vol. 1. Sorting and Searching*, Springer-Verlag, Berlin, New York, 1984.
- [M84b] ———, *Data Structure and Algorithms. Vol. 2. Graph-Algorithms and NP-Completeness*, Springer-Verlag, Berlin, New York, 1984.
- [M84c] ———, *Data Structures and Algorithms. Vol. 3. Multidimensional Searching and Computational Geometry*, Springer-Verlag, Berlin, New York, 1984.
- [MN86] K. MEHLHORN AND S. NÄHER, *Dynamic fractional cascading*, TR 06/1986, FB10, Universität des Saarlandes, Saarbrücken, Federal Republic of Germany, 1986.
- [S73] A. SCHÖNHAGE, *Storage modification machines*, SIAM J. Comput., 9 (1980), pp. 490–508.
- [T79] R. E. TARJAN, *A class of algorithms which require nonlinear time to maintain disjoint sets*, J. Comput. Systems Sci., 18 (1979), pp. 110–127.