A Lower Bound for the Complexity of the Union-Split-Find Problem *

A 07/1986

Kurt Mehlhorn**
Stefan Näher**
Helmut Alt***

**FB10, Informatik
Universität des Saarlandes
D-6600 Saarbrücken
Federal Republic of Germany

***FB Mathematik, WE3
Freie Universität Berlin
D-1000 Berlin 33
Federal Republic of Germany

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 [T79] and Blum [B86]. 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.

* This research was supported by the Deutsche Forschungsgemeinschaft, grant SPP Me 620/6-1

1. Introduction

We consider the following three operations on a linear list $x_1, x_2, \ldots, x_n$ 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 $j = \min\{\ell | \ell \geq i \text{ and } x_\ell \text{ is marked } \}$

Note that the marked items partition the linear list $x_1, \ldots, x_n$ 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 [EKZ77] called it a priority queue problem. He referred to the three operations as Insert, Delete and Successor, and exhibited a $O(\log \log n)$ 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 partioned into collections of records such that each collection corresponds to a currently existing set, ... 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 one views 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 three main results of this paper are:

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 to modify the underlying linear list; the solution does not satisfy the separation assumption.
The lower bound will be proved in section 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 section 3.

3. a) (Tarjan [T79]).*The amortized complexity of the Union-Find problem in the restricted pointer machine model is* $\Theta(\alpha(n))$.

   b) *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*

Tarjan proved his lower bound for the general Union-Find problem. His proof however is actually valid for the Union-Find problem considered here. b) follows immediately from results 1 and 2.

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 2 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 $I = \{x_1^*, x_2^*, \ldots, x_n^*\}$ and a set of output records $O = \{y_1^*, y_2^*, \ldots, y_n^*\}$. (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

1. For all $v \in V$: $\text{outdegree}(v) = 2$
2. There is a set of n input nodes $I = \{x_1, x_2, \ldots, x_n\} \subseteq V$

3. There is a set of n output nodes $O = \{y_1, y_2, \ldots, y_n\} \subseteq V - I$ 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 is 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 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 by 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

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** iff for every input node x there is a path of length at most k from x to the output node y = 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, ... 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 $\{z_1, \ldots, z_L\} \subseteq O$ of output nodes such that no k-structure G' with exactly $z_1, \ldots, z_L$ marked can be constructed from G_0 with less than $(1/2^{2k+4}) \cdot n^{1/2^k}$ edge replacements.

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

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

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

Then the following lemma holds for G_0:

Lemma 2. There is an ℓ, $0 \leq \ell \leq 2^k - 1$, and ℓ distinct nodes $v_1, \ldots, v_\ell$ such that

1. either $\ell = 0$ or $|\bigcap_{1 \leq i \leq \ell} R_{k-1}(v_i)| \geq c_\ell$
2. for each $w \in V - \{v_1, \ldots, v_\ell\}$: $|I^* \cap R_{k-1}(w)| < c_{\ell+1}$
   where $I^* = \bigcap_{1 \leq i \leq \ell} R_{k-1}(v_i)$ ($I^* = I$ if $\ell = 0$)

Proof. Let ℓ_0 be maximal such that there are distinct nodes $v_1, \ldots, v_{\ell_0}$ with $|\cap_{1 \leq i \leq \ell_0} R_{k-1}(v_i)| > c_{\ell_0}$. We only have to show that $\ell_0 < 2^k$. Assume $\ell_0 \geq 2^k$. Then there are 2^k distinct nodes $v_1, \ldots, v_{2^k}$ with $\cap_{1 \leq i \leq 2^k} R_{k-1}(v_i) \neq \emptyset$. 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. ∎

Now let $\ell \geq 0$ and let $v_1, v_2, \ldots, v_\ell$ be nodes satisfying the conditions of lemma 2. We give a sequence of $2^{2k+1} + 1$ SPLIT operations which require at least $(c_\ell/2L - 2^{k+1})/c_{\ell+1}$ edge replacements. First we divide $I^* = \cap_{1 \leq i \leq \ell} R_{k-1}(v_i)$ into $2L$ about equal sized intervals $A_1, A_2, \ldots, A_{2L}$ i.e., 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^*, ...

Clearly, $$|A_i| \geq \frac{c_\ell}{2L} - 1 \quad \text{for all } i.$$

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

Lemma 3. In every A'_j, $j \in \{2, 4, 6, \ldots, 2L\}$, 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 $|\{v \in V | x \in R_k(v)\}| \leq 2^{k+1} - 1$. Thus we conclude

$$\sum_{x \in A_{j-1}} |\{v \in V | x \in R_k(v)\}| \leq |A_{j-1}| \cdot (2^{k+1} - 1)$$
$$= |A'_j| \cdot (2^{k+1} - 1)$$

■

Let $y_j \in A'_j$, $j \in \{2, 4, \ldots, 2L\}$ 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

Lemma 4.

1. $|\{x \in B_{j-1} | v(x) = w\}| \leq c_{\ell+1}$ for every node $w \notin \{v_1, v_2, \ldots, v_\ell\}$ and every j
2. There is a $j \in \{2, 4, 6, \ldots, 2L\}$ such that $v(x) \notin \{v_1, v_2, \ldots, v_\ell\}$ 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 $v_1, \ldots, v_\ell$.
2. For every $v \in \{v_1, v_2, \ldots, v_\ell\}$ we have $|\{y \in O | v \in R_k(y)\}| \leq 2^{k+1}$

   and thus $$\sum_{v \in \{v_1, \ldots, v_\ell\}} |\{y \in O | v \in R_k(y)\}| \leq \ell 2^{k+1}$$
   $$\leq 2^{2k+1} \qquad \text{since } \ell \leq 2^k$$

   i.e., all paths of length at most k using nodes from $\{v_1, \ldots, v_\ell\}$ 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 $v(x) \notin \{v_1, \ldots, v_\ell\}$ for all $x \in B_{j-1}$. ■

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

$$\frac{|B_{j-1}|}{c_{\ell+1}} = \frac{|A_j| - 2^{k+1}}{c_{\ell+1}} = \frac{\frac{c_\ell}{2L} - 1 - 2^{k+1}}{c_{\ell+1}} \geq \frac{\frac{c_\ell}{2^{2k+2}+2} - 1 - 2^{k+1}}{c_{\ell+1}}$$

edges must be changed to obtain G' from G_0.

With $$c_i = n^{1-\frac{1}{2^k}}, \quad 0 \le i \le 2^k$$

we conclude further that

$$\frac{\frac{c_\ell}{2^{2k+2}+2} - (2^{k+1}+1)}{c_{\ell+1}} \ge \frac{n^{\frac{1}{2^k}}}{2^{2k+2}+2} - (2^{k+1}+1)$$

$$\ge \frac{n^{\frac{1}{2^k}}}{2 \cdot (2^{2k+2}+2)} \qquad (\text{since } n \ge 2^{(5k)2^k})$$

$$\ge \frac{n^{\frac{1}{2^k}}}{2^{2k+4}}$$

This completes the proof of the claim and of lemma 1. ∎

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 $n \ge 2^{(5k)2^k}$. Then $k = \Omega(\log\log n)$ and

$$\frac{1}{2^{2k+1}+1} \cdot \frac{1}{2^{2k+4}} n^{\frac{1}{2^k}} \ge \frac{2^{5k}}{2^{2k+4}(2^{2k+1}+1)}$$

$$= \Omega(2^k)$$

$$= \Omega(\log n)$$

$$= \Omega(\log\log n)$$

Let $L = 2^{2k+1}+1$. 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 $n \geq 2^{(5k)2^k}$. Let $L = 2^{2k+1} + 1$ 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 ... . Using this argument s times yields a sequence of m instructions with total cost $\Omega(m \log\log n)$. ∎

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 partioned 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]).

Similarly as in the previous section we define a k-structure.

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

Lemma 5. Let $n \geq (4k)^k/k!$. In any k-structure with output nodes $y_1, \ldots, y_{n-1}$ 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 $\{x_1, \ldots, x_n\}$ 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 $(w_1, v), (w_2, v), \ldots, (w_m, v)$ 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 son of v.)

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

$$p: \quad x = v_0 \longrightarrow v_1 \longrightarrow \cdots \longrightarrow v_\ell = y_n$$

be the path from x to the root y_n in T. For any v_i, $1 \leq i \leq \ell$, on this path let $w_1^i, \ldots, w_j^i = v_{i-1}, \ldots, w_m^i$ be the sons 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 FIND(a) $\neq$ 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 SPLIT(x) requires at least

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

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

The total cost for SPLIT(x) is

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

We first 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

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

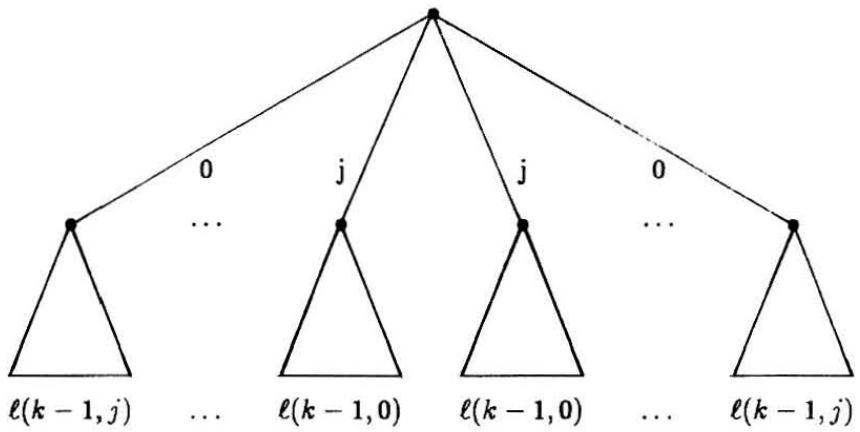


Figure 1

Proof. Define $\ell(k,j)$ 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

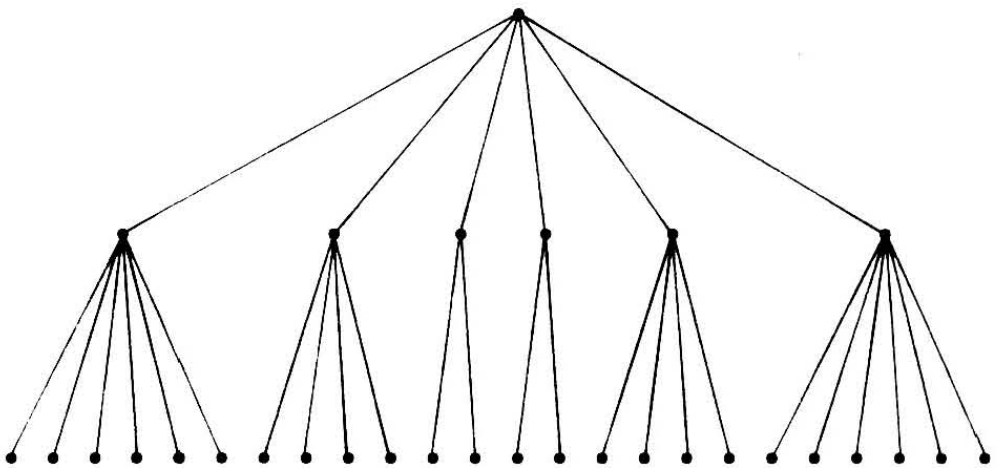
Then we have $\quad \ell(0,j) = 1 \quad \text{for } j \geq 0$

and (cf. figure 1) $\quad \ell(k,j) = 2 \cdot \sum_{i=0}^{j} \ell(k-1, j-i)$

$$= 2 \cdot \sum_{i=0}^{j} \ell(k-1,i) \quad \text{for } k \geq 1$$

The following table gives some values of $\ell(k,j)$. Figure 2 shows a tree of height $k = 2$ with the maximal number of leaves such that all paths have at most cost $j = 2$.

$\ell(k,j)$	0	1	2	3	4	5	6	7	...	k
0	1	2	4	8	16	32	64	128	...	
1	1	4	12	32	80	192	448	...		
2	1	6	24	80	240	672	...			
3	1	8	40	160	560	...				
4	1	10	60	280	...					
5	1	12	84	...						
6	1	14	...							
7	1	...								
$\vdots$										
j										



$\ell(2,2) = 24$

Figure 2

A different interpretation of $\ell(k,j)$ is as follows. Associate with every edge $e = (v,w)$ of T the label $c(e) = \min\{i-1, \text{indegree}(w) - i | v \text{ is the } i\text{-th son of } w\}$. Then we have $0 \leq c(e) \leq j$ for every edge e and for every path $p = v_0 \longrightarrow v_1 \longrightarrow \cdots \longrightarrow v_k$

$$c(p) = \sum_{i=0}^{k-1} c(v_i, v_{i+1})$$

Thus $\ell(k,j) \leq 2^k \cdot$ (number of possible ways to write j as the sum of

k terms from $\{0, 1, \ldots, j\}$)

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

$$\leq 2^k \cdot \binom{k+j-1}{k-1}$$

$$\leq 2^k \cdot \frac{(k+j-1)^{k-1}}{(k-1)!}$$

$$\leq 2^k \cdot \frac{(k+j-1)^k}{k!}$$

$$\leq 2^k \cdot \frac{(k+j)^k}{k!}$$

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

$$n \leq 2^k \cdot \frac{(c(p)+k)^k}{k!}$$

$$\leq 2^k \cdot \frac{(2 \cdot c(p))^k}{k!} \quad \text{since } n \geq (4k)^k/k! \text{ and hence } c(p) \geq k$$

and finally

$$c(p) \geq \frac{1}{2}\left(\frac{n \cdot k!}{2^k}\right)^{1/k}$$

$$\geq \frac{k}{12} \cdot n^{1/k} \quad \text{since } (k!)^{1/k} \geq \frac{k}{3} \text{ if } k \geq 6$$

This completes the proof of lemma 5. ∎

Theorem 2. In the restricted pointer machine model

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 $\Omega(\log n)$.

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. ∎

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. One only has to represent each interval by a balanced tree; cf. e.g. [M84a, section III.5.3.1].

We close this section with our third result.

Theorem 3.

a) (Tarjan [T79]). The amortized complexity of the Union-Find problem in the restricted pointer machine model is $\Theta(\alpha(n))$.

b) 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.

a) Although Tarjan stated his result for the general Union-Find problem his proof actually works for our restricted version of it.

b) This follows immediately from theorem 1 and 2. ∎

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 for 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.

The following table 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.

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

There are still several open problems in both models. We have no lower bounds for the amortized complexity of Union-Find and Split-Find and for the worst-case complexity of Union-Find in the general model. In the restricted model it remains an open problem whether the Split-Find algorithm of Hopcroft/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 have amortized complexity $\Theta(1)$ on random access machines. This was shown by

Garbow/Tarjan [GT83] for the Union-Find problem and by Imai/Asano [IA84] for the Split-Find problem.

5. References

[**B86**] N. Blum: "On the Single-Operation Worst-Case Time Complexity of the Disjoint Set Union Problem", SIAM J. Comput., Vol. 15, No. 4, 1986, 1021-1024

[**EKZ77**] P. v. Emde Boas, R. Kaas, E. Zijlstra: "Design and Implementation of an Efficient Priority Queue", Math. Systems Theory 10, 1977, 99-127

[**GT83**] H.N. Gabow, R.E. Tarjan: "A linear-time algorithm for a special case of disjoint set union", Proc. 15-th Ann. SIGACT Symp., 1983, 246-251

[**IA84**] T. Imai, T. Asano: "Dynamic Segment Intersection with Applications", 25th FOCS, 1984, 393-402

[**HU73**] J.E. Hopcroft, J.D. Ullman: "Set Merging Algorithms", SIAM J. Comput., Vol. 2, 1973, 294-304

[**Ka84**] R.G.Karlsson:"Algorithms in a Restricted Universe", Report CS-84-50, Department of Computer Science, University of Waterloo, 1984

[**Kn68**] D.E. Knuth: "The Art of Computer Programming", Vol. 3, "Fundamental Algorithms", Addison-Wesley, Reading, Mass., 1968.

[**Ko53**] A.N. Kolmogorov: "On the notion of Algorithm", Uspehi Mat. Nauk. 8 (1953), 175-176

[**M84**] K. Mehlhorn: "Data Structures and Algorithms", Springer Publ. Comp., 1984

a) Vol. 1: Sorting and Searching
b) Vol. 2: Graph-Algorithms and NP-Completeness
c) Vol. 3: Multidimensional Searching and Computational Geometry

[MN86] K. Mehlhorn, S. Näher: "Dynamic Fractional Cascading", TR 06/1986, FB10, Universität des Saarlandes, Saarbrücken, 1986

[S73] A. Schönhage: "Storage Modification Machines", SIAM J. Comput., Vol. 9, 1980, 490-508

[T79] R.E. Tarjan: " A Class of Algorithms which Require Nonlinear Time to Maintain Disjoint Sets", J. Comp. Sys. Sci. 18, 1979, 110-127