# A Note on Predecessor Searching in the Pointer Machine Model ☆

Wolfgang Mulzer

*Department of Computer Science, Princeton University, 35 Olden Street, Princeton, NJ 08540, USA*

**Abstract**

Predecessor searching is a fundamental data structuring problem and at the core of countless algorithms: given a totally ordered universe $\mathcal{U}$ with $n$ elements, maintain a subset $S \subseteq \mathcal{U}$ such that for each element $x \in \mathcal{U}$ its predecessor in $S$ can be found efficiently. During the last thirty years the problem has been studied extensively and optimal algorithms in many classical models of computation are known. In 1988, Mehlhorn, Näher, and Alt [1] showed an amortized lower bound of $\Omega(\log\log n)$ in the pointer machine model. We give a different proof for this bound which sheds new light on the question of how much power the adversary actually needs.

*Key words:* analysis of algorithms, data structures, predecessor searching, lower bound, pointer machine

## 1. Introduction

We consider the classic *predecessor searching* problem in which we are given a totally ordered universe $\mathcal{U} = \{1, \ldots, n\}$ with $n$ elements and where we need to maintain a set $S \subseteq \mathcal{U}$ under the operations `insert`, `delete`, and `pred`. For $x \in \mathcal{U}$, `insert`$(x)$ adds $x$ to $S$, `delete`$(x)$ removes $x$ from $S$, and `pred`$(x)$ finds the largest element in $S$ that is smaller than $x$. This is one of the most fundamental data structuring problems and has been studied extensively in the literature, so that nowadays optimal solutions for a wide variety of models are known. We refer the reader to Beame and Fich [2] and Pătraşcu and Thorup [3] for an overview and some recent results in this area.

We consider one of the most old-fashioned incarnations of the problem, namely, we consider predecessor searching on a pointer machine (this model is explained in detail in the next section). In this setting, there is a classical solution based on van-Emde-Boas trees [4] which achieves $O(\log\log n)$ worst-case time per operation, and this was shown to be optimal by Mehlhorn, Näher, and Alt [1]. Mehlhorn *et al* describe an adversary that locally analyses the data structure and in each step performs the worst possible operation. Our note is motivated by the following question: *how much knowledge does the adversary actually need?* To answer this question, we provide a different proof of the core lemma of Mehlhorn *et al*. Our proof is based on a more global analysis of the data structure and shows that the adversary does not need to be too clever most of the time. Rather, it turns out that if things are set up right, almost every possible predecessor query will be difficult. Furthermore, we believe that our new, more global viewpoint leads to a slightly simpler and more straightforward proof of the lower bound.

☆This work was supported in part by NSF grant CCF-0634958 and NSF CCF 0832797.

*Email address:* wmulzer@cs.princeton.edu (Wolfgang Mulzer)

## 2. The Model

As our model of computation we use the *pointer machine*, which captures the list processing capabilities of a computer [5, 6, 7, 8]. In this setting, the *data structure* is a directed graph $D$ in which every node has out-degree at most 2. The data structure is dynamic and may change throughout the computation. However, we assume that the vertex set $V$ of $D$ stays the same. The data structure contains a set $I = \{i_1, \ldots, i_n\} \subseteq V$ of $n$ distinguished *input nodes* and a set $O = \{o_1, \ldots, o_n\}$ of $n$ distinguished *output nodes* with $I \cap O = \emptyset$. They each correspond to the elements of the universe. All the input nodes have in-degree 0, and all the output nodes have out-degree 0. Every output node can be marked or unmarked, indicating whether the corresponding element is present in the subset $S$ or not. For a given data structure $D$, we call the set of marked output nodes the *elements stored in* $D$, denoted by $S(D)$. Let $x \in \mathcal{U}$ and $i_x \in I$ be the corresponding input node, then $\texttt{pred}_D(i_x)$ denotes the output node $o_y$ such that $y$ is the predecessor of $x$ in $S(D)$. Furthermore, for $v, v' \in V$, we let $d_D(v, v')$ denote the length of a shortest path from $v$ to $v'$, the *distance* of $v'$ from $v$.

We now describe how the operations are carried out. The algorithm receives an input node $i_x$ and a data structure $D$. For an `insert` or `delete` operation, the algorithm is required to return a data structure $D'$ such that $S(D') = S(D) \cup \{x\}$ in case of insertion and $S(D') = S(D) \setminus \{x\}$ in case of deletion. For a `pred` query, the algorithm must return the output node $\texttt{pred}_D(i_x)$ and a data structure $D'$ with $S(D') = S(D)$. The *cost* of an `insert` or `delete` operation is given by $|E(D') \setminus E(D)|$, the cost of a `pred` query is given by $|E(D') \setminus E(D)| + d_D(i_x, \texttt{pred}_D(i_x))$. In words, the cost of an operation is the number of edges the algorithm changes in the data structure plus the length of a shortest path from the input to its predecessor in case of a `pred` query. Note that if $D$ does not contain a path from the input to its predecessor, the `pred` operation has infinite cost.

## 3. The Lower Bound

Our proof follows the general strategy of Mehlhorn *et al*. We fix $\sigma := \frac{1}{3}\log\log n - 1$ and $\tau := 2^{\sigma+1} = \sqrt[3]{\log n}$. For a given data structure $D$, we say that the input node $i_x$ is *satisfied* in $D$ if $d_D(i_x, \texttt{pred}_D(i_x)) \leq \sigma$, and *unsatisfied* in $D$ otherwise. At the heart of the argument is the following lemma:

**Lemma 3.1.** *Let $D_0$ be any data structure with $S(D_0) = \emptyset$. There exists a nonempty set $S \subseteq \mathcal{U}$, such that in every data structure $D$ with $S(D) = S$ and $|E(D) \setminus E(D_0)| \leq \sigma|S|$ at least $(1 - 2/\tau)n$ input nodes are unsatisfied.*

We call a set $S \subseteq \mathcal{U}$ as in Lemma 3.1 a *difficult set* for $D_0$. Mehlhorn *et al* prove a similar lemma with a weaker guarantee. While they show that there exists a difficult set such that at least one input node is unsatisfied, we prove that almost all input nodes are unsatisfied. Following Mehlhorn *et al*, it is now easy to prove the lower bound:

**Theorem 3.2.** *Let $\mathcal{U} = \{1, \ldots, n\}$ be a totally ordered universe of size $n$ and $D_0$ a data structure with $S(D_0) = \emptyset$. Any algorithm that solves the predecessor searching problem on $\mathcal{U}$ has amortized complexity $\Omega(\log\log n)$ per operation. More precisely, for every $m \in \mathbb{N}$ there exists a sequence $O$ of at least $m$ operations such that the total cost of performing $O$ with initial data structure $D_0$ is at least $\sigma|O|/3$, where $|O|$ denotes the length of $O$.*

*Proof.* We describe how to construct the operation sequence $O$: it consists of several *rounds*, each of which starts with an empty data structure. A round is constructed as follows: let $D_0$ be the initial data structure, and let $S_0 \subseteq \mathcal{U}$ be a difficult set for $D_0$ as in Lemma 3.1. There are $3|S_0|$ operations: in the first $|S_0|$ steps, all elements from $S_0$ are inserted into the data structure, and in the last $|S_0|$ steps, the elements from $S_0$ are deleted. The remaining $|S_0|$ operations are predecessor queries that are determined as follows: at any point, if an unsatisfied element exists, query it. Otherwise, query an arbitrary element in the universe. It now follows that the round has cost at least $\sigma|S_0|$: either, we query $|S_0|$ unsatisfied elements, or at some point all elements in the current data structure are satisfied, which means, by Lemma 3.1, that the current data structure differs from $D_0$ by at least $\sigma|S_0|$ edges. We can now obtain a difficult operation sequence of length more than $m$ by adding new rounds until the total number of operations exceeds $m$.

Note that the proof also goes through if in each round we pick the predecessor queries uniformly at random, because by Lemma 3.1 almost all input nodes are unsatisfied. □

It remains to prove the lemma.

*Proof of Lemma 3.1.* Let $v \in V$. We define the *group* of $v$ in $D_0$ as the set of input nodes in $D_0$ that can reach $v$ on a path of length at most $\sigma$. It is called $A_v$:

$$A_v := \{i_x \in I \mid d_{D_0}(i_x, v) \leq \sigma\}.$$

Since $D_0$ has out-degree at most 2, for each $i_x \in I$ there can be at most $\tau$ groups $A_v$ with $i_x \in A_v$.

For $0 \leq k \leq \tau^2$, let $d_k := n^{k/\tau^2}$. We let $B_k$ denote the set of input nodes that are in a group whose size is in the interval $(d_{k-1}, d_k]$:

$$B_k := \{(i_x, v) \mid v \in V, i_x \in A_v \text{ and } d_{k-1} < |A_v| \leq d_k\}.$$

**Claim 3.3.** *There exists an $1 \leq \beta \leq \tau^2$ such that $|B_\beta| \leq n/\tau$.*

*Proof.* Since the $B_k$ are disjoint, we have

$$\begin{aligned}\sum_{k=1}^{\tau^2} |B_k| &= \left|\bigcup_{k=1}^{\tau^2} B_k\right| \\ &\leq |\{(i_x, v) \mid v \in V, i_x \in A_v\}| \\ &= \sum_{x \in \mathcal{U}} |\{v \in V \mid i_x \in A_v\}| \leq \tau n,\end{aligned}$$

since each $i_x$ appears in at most $\tau$ groups $A_v$. Therefore, there is at least one $B_\beta$ of size at most $n/\tau$. □

Let $a := 2\beta - 1$. We subdivide the universe into $n^{1-a/2\tau^2}$ *intervals* of size $b := n^{a/2\tau^2}$: interval $I_1$ consists of elements $1, \ldots b$, interval $I_2$ consists of elements $b+1, \ldots, 2b$, and so on. Let $I_{2l}$ be an interval with even index and $x \in I_{2l}$. The output node $o_x$ is *good*, if $|Q_l \cap A_{o_x}| \leq \tau^2$, where $Q_l$ is the set of input nodes corresponding to the elements in $I_{2l} \cup I_{2l+1} \cup I_{2l+2}$ (we define $I_{2l+1}, I_{2l+2} = \emptyset$, if necessary):

$$Q_l := \{i_x \mid x \in I_{2l} \cup I_{2l+1} \cup I_{2l+2}\}.$$

The output node $o_x$ is called *bad* otherwise. The reason for this definition is that if we mark a good output node $o_x$, at most $\tau^2$ input nodes in the relevant intervals have distance less than $\sigma$ from $o_x$, so all but $\tau^2$ input nodes for the succeeding elements in $I_{2l} \cup I_{2l+1} \cup I_{2l+2}$ will be unsatisfied. An element $x \in \mathcal{U}$ is called *good* or *bad* depending on whether its corresponding output node is good or bad.

**Claim 3.4.** *Every interval $I_{2l}$ contains at least one good element.*

*Proof.* We have

$$\begin{aligned}\sum_{x \in I_{2l}} |Q_l \cap A_{o_x}| &= \sum_{i_y \in Q_l} \left|\{o_x \mid x \in I_{2l}, d_{D_0}(i_y, o_x) \leq \sigma\}\right| \\ &\leq \sum_{i_y \in Q_l} \tau = \tau|Q_l| \leq 3\tau b,\end{aligned}$$

since $|Q_l| \leq 3b$. It follows that there can be at most $3b/\tau$ elements $x \in I_{2l}$ with $|Q_l \cap A_{o_x}| > \tau^2$. Since $3/\tau < 1$, the claim follows. □

Pick a good node from every interval $I_{2l}$ and call the resulting set $S$. Let $D_0'$ be the data structure derived from $D_0$ by marking exactly the output nodes $o_x$ with $x \in S$.

**Claim 3.5.** *At most $\frac{2}{3}\tau^2 n^{1-1/2\tau^2}$ nodes in $D_0'$ are satisfied.*

*Proof.* At most $2b \leq \frac{1}{6}n^{1-a/2\tau^2}$ input nodes are satisfied for elements in $I_1, I_2$. By construction, for each $x \in S$ there are at most $\tau^2$ satisfied input nodes $i_y$ such that $\texttt{pred}_{D'}(i_y) = o_x$. The set $S$ contains at most $\frac{1}{2}n^{1-a/2\tau^2}$ elements, and since $a \geq 1$, the claim follows. □

Let $D$ be any data structure with $S(D) = S$ and $|E(D) \setminus E(D_0)| \leq \sigma|S|$. Let $i_x \in I$ be an input node that is satisfied in $D$, but in $D_0$ we have $d_{D_0}(i_x, \texttt{pred}_D(i_x)) > \sigma$. Let $(v, v')$ be the first edge on a shortest path from $i_x$ to $\texttt{pred}_D(i_x)$ in $D$ that is not in $E(D_0)$. We say that *$v$ satisfies $i_x$*. Note that it follows that $d_{D_0}(i_x, v) \leq \sigma$.

**Claim 3.6.** *Let $v \in V$. Then*

$$|\{\texttt{pred}_D(i_x) \mid i_x \in I \text{ is satisfied by } v\}| \leq \tau,$$

*that is, the input nodes satisfied by $v$ have at most $\tau$ different predecessors.*

*Proof.* Assume $v$ satisfies $i_x$. By definition, $d_D(v, \texttt{pred}_D(i_x)) \leq \sigma$. Since $D$ has out-degree 2 and since every element node is a sink, there can be at most $\tau$ element nodes at distance at most $\sigma$ from $v$. The claim follows. □

Call a node $v \in V$ *big*, if $|A_v| > d_\beta$, *small*, if $|A_v| \leq d_{\beta-1}$, and *medium*, if $d_{\beta-1} < |A_v| \leq d_\beta$. Let $I^{\text{sat}}$ be the set of satisfied input nodes in $D$. Recall that for every node in $I^{\text{sat}}$ that is not satisfied in $D'$, there exists a node $v \in V$ that satisfies it. We have

$$I^{\text{sat}} = I^{\text{sat}}_{\text{initial}} \cup I^{\text{sat}}_{\text{big}} \cup I^{\text{sat}}_{\text{medium}} \cup I^{\text{sat}}_{\text{small}}, \tag{1}$$

where $I^{\text{sat}}_{\text{initial}}$ denotes the input nodes that are already satisfied in $D'_0$, and $I^{\text{sat}}_{\text{big}}, I^{\text{sat}}_{\text{medium}}, I^{\text{sat}}_{\text{small}}$ denote the input nodes satisfied by a big, medium, or small node, respectively. By Claim 3.5, we have $\left|I^{\text{sat}}_{\text{initial}}\right| \leq 2\tau^2 n^{1-1/2\tau^2}/3$. Furthermore, by Claim 3.3, the number of input nodes $i_x$ such that $d_{D_0}(i_x, v) \leq \sigma$ for a medium node $v$ is at most $n/\tau$, and hence $\left|I^{\text{sat}}_{\text{medium}}\right| \leq n/\tau$.

**Claim 3.7.** *We have $\left|I^{sat}_{big}\right| \leq 3\tau^2 n^{1-1/2\tau^2}$.*

*Proof.* Let $v$ be a big node. By Claim 3.6, $|\{\texttt{pred}_D(i_x) \mid i_x \in I \text{ is satisfied by } v\}| \leq \tau$. Furthermore, by construction, for each $x \in S$, the set of input nodes $i_y$ such that $\texttt{pred}_{D'}(i_y) = o_x$ has cardinality at most $3n^{a/2\tau^2}$. Hence, $v$ can satisfy at most $3\tau n^{a/2\tau^2}$ nodes.

Since $v$ is big, there exist more than $d_\beta$ input nodes $i_x$ such that $d_{D_0}(i_x, v) \leq \sigma$. As for each input node there are at most $\tau$ such nodes, there can be at most $\tau n/d_\beta$ big nodes. Recalling that $d_\beta = n^{\beta/\tau^2}$ and that $a = 2\beta - 1$, it follows that

$$|I^{\text{sat}}_{\text{big}}| \leq 3\tau n^{\beta/\tau^2 - 1/2\tau^2} \cdot \tau n^{1-\beta/\tau^2} = 3\tau^2 n^{1-1/2\tau^2}.$$

□

**Claim 3.8.** *We have $\left|I^{sat}_{small}\right| \leq \frac{1}{2}\sigma n^{1-1/2\tau^2}$.*

*Proof.* Let $v$ be a small node and let $i_x$ be an input node such that $d_{D_0}(i_x, v) \leq \sigma$. Since $v$ is small, there are at most $d_{\beta-1} = n^{(\beta-1)/\tau^2}$ such nodes $i_x$. Hence, $v$ can satisfy at most $n^{(\beta-1)/\tau^2}$ input nodes. If a small node $v$ satisfies an input node in $D$, there has to be an edge in $E(D) \setminus E(D_0)$ leaving $v$. By assumption, there are at most $\sigma|S| \leq \sigma\frac{1}{2}n^{1-a/2\tau^2}$ such edges. Using $a = 2\beta - 1$, it follows that

$$|I^{\text{sat}}_{\text{small}}| \leq n^{(\beta-1)/\tau^2} \cdot \frac{1}{2}\sigma n^{1-(\beta-1)/\tau^2 - 1/2\tau^2} = \frac{1}{2}\sigma n^{1-1/2\tau^2}.$$

□

By Claims 3.3, 3.5, 3.7, 3.8, we can conclude from (1) that

$$\begin{aligned} |I^{\text{sat}}| &\leq \frac{2}{3}\tau^2 n^{1-1/2\tau^2} + 3\tau^2 n^{1-1/2\tau^2} + \frac{n}{\tau} + \frac{1}{2}\sigma n^{1-1/2\tau^2} \\ &\leq \frac{n}{\tau} + 4\tau^2 n^{1-1/2\tau^2} \leq \frac{2n}{\tau}, \end{aligned}$$

since $4\tau^3 = 4\log n \leq 2^{\frac{1}{2}\log^{\frac{1}{3}} n} = n^{1/2\tau^2}$ for $n$ large enough. This finishes the proof. □

**Acknowledgments:** I wish to thank Bernard Chazelle for many useful discussions.

## References

[1] K. Mehlhorn, S. Näher, H. Alt, A lower bound on the complexity of the union-split-find problem, SIAM J. Comput. 17 (6) (1988) 1093–1102.

[2] P. Beame, F. E. Fich, Optimal bounds for the predecessor problem and related problems, J. Comput. System Sci. 65 (1) (2002) 38–72, special issue on STOC, 1999 (Atlanta, GA).

[3] M. Pătraşcu, M. Thorup, Time-space trade-offs for predecessor search, in: Proc. 38th ACM Symposium on Theory of Computing (STOC), 2006, pp. 232–240, see also arXiv:0603043.

[4] P. van Emde Boas, R. Kaas, E. Zijlstra, Design and implementation of an efficient priority queue, Math. Systems Theory 10 (2) (1976/77) 99–127, sixteenth Annual Symposium on Foundations of Computer Science (Berkeley, Calif., 1975), selected papers.

[5] D. E. Knuth, The Art of Computer Programming: Fundamental Algorithms, 3rd Edition, Vol. 1, Addison Wesley, Reading, Massachusetts, 1997.

[6] A. N. Kolmogorov, On the notion of algorithm, Uspekhi Mat. Nauk. 8 (1953) 175–176.

[7] A. Schönhage, Storage modification machines, SIAM J. Comput 9 (3) (1980) 490–508.

[8] R. E. Tarjan, A class of algorithms which require nonlinear time to maintain disjoint sets, Journal of Computer and System Sciences 18 (2) (1979) 110–127.