A LOWER WORST-CASE COMPLEXITY FOR SEARCHING A DICTIONARY

D. S. Hirschberg

July, 1978

TECHNICAL REPORT #7808

Research supported by NSF grant MCS-76-3933.
A LOWER WORST-CASE COMPLEXITY
FOR SEARCHING A DICTIONARY

D. S. Hirschberg

July, 1978

Abstract:
It is shown that \( k(p+3)/2 + p - 2 \) letter comparisons suffice to determine whether a word is a member of a lexicographically ordered dictionary containing \( 2^p - 1 \) words of length \( k \). This offers a potential savings (compared to worst case complexity of binary search) that asymptotically approaches 50 percent.

-----------------------------
Research supported by NSF grant MCS-76-3933. Author's current address: Department of Electrical Engineering, Rice University, Houston Texas 77001.
We consider the problem of searching a dictionary containing \( n \) words, each word consisting of at most \( k \) letters. It is assumed that the alphabet is ordered and that the dictionary is in lexicographic order.

Two letters can be compared (to determine which has lower rank in the alphabet) in unit time. Our goal is to exhibit an algorithm for searching the dictionary that has minimal worst case time complexity.

We denote the ordered set of words (\( k \)-vectors) by \( \mathcal{B} \) and refer to the \( i \)-th word (vector) by \( B_i \). We wish to determine if the vector \( A \) is in \( \mathcal{B} \). Assume that \( n=2^p-1 \), i.e. \( p=\lceil \log(n+1) \rceil \).

Let \( B_m \) be the "middle" vector (\( m=2^{p-1} \)), let \( B_q \) be the "quarter" vector (\( q=2^{p-2} \)), and let \( B_r \) be the "third-quarter" vector (\( r=2^{p-1}+2^{p-2} \)).

Using binary search, i.e. comparing \( A \) first with \( B_m \) and then, depending upon the result, comparing \( A \) with either \( B_q \) or with \( B_r \) etc. will lead to a worst case complexity of \( kp \) comparisons.

In a recent paper [2], Hirschberg has exhibited an algorithm, \textsc{Vector-Search}, that has a worst case complexity of less than \( k(p+2-\log k) \). For many values of \( n \) and \( k \), this algorithm offers a potential savings of from 5 to 25 percent.
Let there have been \( x \) "equal" comparisons between \( A \) and \( B \) (if no comparisons were made between \( A \) and \( B \) then \( x = 0 \)) and \( y \) "equal" comparisons between \( A \) and \( B \). Without loss of generality, assume that \( x \leq y \) (the case \( x > y \) will lead to symmetric arguments).

For the algorithm we shall next consider, which we call modified binary search, let \( T(x, y, p) \) be the minimum number of comparisons required for \( x \), \( y \), and \( p \) as defined above and for \( k \) (the length of the vectors) implicit. We shall demonstrate a savings that asymptotically approaches 50 percent of the worst case complexity as compared to the binary search method.

Note that for all \( 1 \leq i \leq n \) we need not compare \( a_{ij} : b_{ij} \) for \( j \leq x \) since the result must be "equal" from the results of previous comparisons and the fact that \( B \) is in lexicographic order.

Our algorithm first compares components of \( A \) with components of \( B \) (starting at component \( x+1 \)) until it is determined whether \( A < B \), \( A = B \), or \( A > B \).

\[
T(x, y, p) = T(x, t, p-2) + t+1-x \quad \text{(1)}
\]

**Case 1.** If \( A < B \) because \( a_{\bar{q}} < b_{\bar{q}} \) \((x \leq \bar{t} < k)\) then

\[
T(x, y, p) = T(x, t, p-2) + t+1-x
\]

**Case 2.** If \( A > B \) because \( a_{\bar{q}} > b_{\bar{q}} \) \( \text{then compare } A \text{ with another vector depending on the value of } t. \)

**Case 2a.** If \( t \leq y \) then compare \( A \) with \( B \) (starting with
component t+1) 

__2aa. If A<B because a \text{h+1 < b \space m h+1} \space (x \leq t \leq h < k) then  
\[ T(x, y, p) = T(t, h, p-2) + h+2-x \tag{2} \]

__2ab. If A>B because a \text{h+1 > b \space m h+1} \space (x \leq t \leq h < k) then  
\[ T(x, y, p) = T(\min\{h, y\}, \max\{h, y\}, p-1) + h+2-x \tag{3} \]

__2ac. If A=B then  
\[ T(x, y, p) = k+1-x \tag{4} \]

Case 2b. If t \geq y then compare A with B (starting with \text{component y+1})

__2ba. If A<B because a \text{h+1 < b \space r h+1} \space (y \leq h, t < k) then  
\[ T(x, y, p) = T(\min\{h, t\}, \max\{h, t\}, p-1) + t+1-x + h+1-y \tag{5} \]

__2bb. If A>B because a \text{h+1 > b \space r h+1} \space (y \leq h, t < k) then  
\[ T(x, y, p) = T(y, h, p-2) + t+1-x + h+1-y \tag{6} \]

__2bc. If A=B then  
\[ T(x, y, p) = t+1-x + k-y \tag{7} \]

Case 3. If A=B then  
\[ T(x, y, p) = k-x \tag{8} \]

The resulting problem has either half or quarter the range (in B) of the original problem. If the range is reduced to zero then terminate with the solution that A is not in B. If the range consists of one vector B then compare a \text{...a} \space with the corresponding k-x components of B. Otherwise, recursively apply the modified binary search algorithm.
Theorem. For \( p \geq 1 \), \( T(x, y, p) \leq k(p+3)/2 + p - 2 - x - y \)

Proof. By induction on \( p \). Note that \( T(x, y, 0) = 0 \). The theorem, for \( p=1 \), is true since \( T(x, y, 1) = k-x \) (the algorithm treats this case separately). The theorem, for \( p=2 \), is also true as can been seen by substituting for \( T(\cdot, \cdot, 0) \) and \( T(\cdot, \cdot, 1) \) in the righthand side of the formulae given in all of the cases, one of which must occur.

For the inductive step, \( T(x, y, p) \) is bounded from above by the maximum of the cases. In all cases, substituting in the assumed complexity for \( T \) (with smaller value of the third argument) substantiates that the theorem holds. \( \square \)

The table below illustrates typical percentage savings (worst case analysis) of VECTOR_SEARCH and modified binary search as compared to binary search.

<table>
<thead>
<tr>
<th>(-n) (p)</th>
<th>Binary search</th>
<th>VECTOR_SEARCH (%)</th>
<th>modified search (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>125 (7) 4</td>
<td>28</td>
<td>26 (7.1)</td>
<td>25 (10.7)</td>
</tr>
<tr>
<td>8</td>
<td>56</td>
<td>46 (17.9)</td>
<td>45 (19.6)</td>
</tr>
<tr>
<td>16</td>
<td>112</td>
<td>78 (30.4)</td>
<td>81 (27.7)</td>
</tr>
<tr>
<td>1000 (10) 4</td>
<td>40</td>
<td>38 (5.0)</td>
<td>34 (15.0)</td>
</tr>
<tr>
<td>8</td>
<td>80</td>
<td>70 (12.5)</td>
<td>60 (25.0)</td>
</tr>
<tr>
<td>16</td>
<td>160</td>
<td>126 (21.2)</td>
<td>112 (30.0)</td>
</tr>
<tr>
<td>1000 (20) 4</td>
<td>80</td>
<td>78 (2.5)</td>
<td>64 (20.0)</td>
</tr>
<tr>
<td>8</td>
<td>160</td>
<td>150 (6.2)</td>
<td>110 (31.2)</td>
</tr>
<tr>
<td>16</td>
<td>320</td>
<td>286 (10.6)</td>
<td>202 (36.9)</td>
</tr>
</tbody>
</table>
References


