THE COMPLEXITY OF PATTERN MATCHING FOR A RANDOM STRING

by

Andrew C. Yao

STAN-CS-77-629
OCTOBER 1977

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

The Complexity of Pattern Matching for a Random String

Andrew Chi-Chih Yao

Computer Science Department
Stanford University
Stanford, California 94305

Abstract.

We study the average-case complexity of finding all occurrences of a given pattern α in an input text string. Over an alphabet of q symbols, let $c(\alpha,n)$ be the minimum average number of characters that need to be examined in a-random text string of length n . We prove that, for large m , almost all patterns α of length m satisfy

$$c(\alpha,n) = \Theta\left(\left\lceil \log_q\left(\frac{n-m}{\ln m} + 2\right)\right\rceil\right) \text{ if } m \le n \le 2m \text{ , and}$$

$$c(\alpha,n) = \Theta\left(\frac{\lceil \log_q m\rceil}{m}\, n\right) \text{ if } n > 2m \text{ .}$$

This in particular confirms a conjecture raised in a recent paper by Knuth, Morris, and Pratt [4].

Keywords: algorithm, average-case complexity, complexity, decision tree, pattern matching, random string, string, weighted q-ary tree.

This research was supported in part by National Science Foundation grant MCS 72-03752 A03.

Introduction.

A basic string pattern matching problem is to find all occurrences given string (called pattern) as a contiguous block in an input string (called text string). Thus, for the pattern 00100 , there are three occurrences of it to be located in the text string 1000100100000100011 . Several efficient algorithms have been devised to solve this problem [1,3,4]. For example, Knuth, Morris, and Pratt [4] constructed an algorithm that has a worse-case running time of $O(m+n)$, where m and n are the lengths of the pattern and the text string, respectively.

The optimality question of algorithms for the above problem was investigated in Knuth, Morris, and Pratt [4] and in Rivest [6]. In their model, an algorithm is a decision tree that examines the text string one character at a time, and the cost is measured in terms of the number of characters examined. (For a similar model in a related problem, see Aho, Hirshberg, and Ullman [2].) Rivest [6] proved that, for any pattern, an algorithm has to inspect $n-m+1$ characters for some text string. This means that, when $n >> m$, almost the entire text string has to be examined in the worst case. A different situation exists for the average-case complexity. Let $c(\alpha,n)$ be the minimum average number of characters that need to be examined in a random text string of length n , in order to locate all occurrences of α . Knuth described an algorithm [4, Section 8] to show that, for any given pattern α , $c(\alpha,n) \leq O(n\lceil \log_q m \rceil/m)$ for an alphabet of size q . Thus, for large m , only a small fraction of the characters in the text string need to be looked at, Such "sublinear" algorithms are particularly attractive in situations when a text string is input only once, but will be updated and searched for patterns many times. Knuth conjectured that the algorithm is optimal in the following sense: there exist patterns

α of arbitrarily large length m such that $\alpha,n) \geq \Omega(n\lceil \log_q m\rceil/m)$ for all sufficiently large n. This conjecture is interesting since, as shown in [4], there are patterns such as 0^m for which only $O(n/m)$ characters need to be tested on the average.

In this paper, we study the average-case complexity of pattern matching in the model of [4]. We'prove that, for large m , almost all patterns α of length m satisfy $c(\alpha,n) = \Theta\left(\left\lceil \log_q\left(\frac{n-m}{\ln m}+2\right)\right\rceil\right)$ if $m \leq n < 2n$, and $c(\alpha,n) = \Theta\left(\frac{\lceil \log_q m\rceil}{m} n\right)$ if $n > 2m$. Moreover, all lower bounds actually apply to the best-case performance of any algorithms, not just their average case. These results in particular confirm the above-mentioned conjecture when $n \geq 2m$. Note also a point of interest. In Knuth's algorithm, the text string is examined in a predetermined sequence of positions independent of the pattern (except its length m); whereas for $m \leq n \leq 2m$, we can show that any algorithm with a fixed sequence of probing positions have to examine $\Omega(\lceil \log_q (n-m+2)\rceil)$ characters, even in the best case, for some patterns. Thus, "non-adaptive" pattern matching algorithms cannot be optimal when n is close to m, e.g. when $n-m \approx (\ln m)(\ln\ln m)$.

Definitions and precise statements of the main results are given in Section 2. In Section 3, we familiarize ourselves with some useful concepts by analyzing the algorithm in [4] for $m \leq n \leq 2m$. In the course of analysis, we shall also develop insight into the design of a faster algorithm. An improved algorithm is then described and analyzed in Section 4 to establish the upper bounds. In Section 5, we define the complexity notion of a "certificate". Our lower bounds then follow from stronger results that we can prove about the length of a minimum certificate. Certain properties of a type of optimal digital search trees (cf. Knuth [5]) are needed in the paper; their derivations are given in the appendices.

2. Definitions and Main Results.

An alphabet is a finite, nonempty set of symbols. Throughout our discussions, we will assume a unique underlying alphabet Σ of size q. A string ζ of length ℓ is a concatenation of ℓ symbols from Σ, i.e., $\zeta = a_1 a_2 \ldots a_\ell$ where $\ell \geq 0$ and each $a_i \in \Sigma$. We use $\zeta[i]$ to denote a_i, the i-th symbol of ζ, and $\|\zeta\|$ to denote ℓ, the length of ζ. The collection of all strings of length ℓ is denoted by Σ^ℓ. Given two strings $\alpha \in \Sigma^m$ and $\beta \in \Sigma^n$ with $m < n$, α is said to be a subs-king of β if $\alpha = \beta[i]\,\beta[i+1] \ldots \beta[i+m-1]$ for some i, $1 \leq i \leq n-m+1$. Alternatively, we say a occurs in β, or β contains an occurrence of α, etc.; the index i is called the (leftmost) position of the occurrence. The substring $\beta[i]\,\beta[i+1] \ldots \beta[j]$ of $\beta \in \Sigma^n$, where $1 \leq i \leq j \leq n$, will be denoted by $\beta[i:j]$.

A pattern is a distinguished string of positive length. Given a pattern α of length m and an integer $n > m$, we shall be interested in locating all occurrences of α in any input string $\zeta \in \Sigma^n$ (ζ is called the text string). Let us refer to this as the pattern-matching problem with respect to α and n. From now on, the notations α, ζ and m, n will be used exclusively for the pattern, the text string, and their respective length in a pattern-matching problem. Since the problem is trivial when $q = |\Sigma| = 1$, we shall assume $q \geq 2$.

As our computation model, we consider algorithms that proceed by asking a series of questions $\zeta[i_1] = ?$, $\zeta[i_2] = ?$, . . . , where the choice of each position i_r may depend on answers to al!, previous probes at $\zeta[i_1], \zeta[i_2], \ldots, \zeta[i_{r-1}]$. When the algorithm halts, it must have enough information to determine $A(\alpha, \zeta)$, the set of all leftmost positions of α's occurrences in ζ . Formally, $A(\alpha,\zeta) = \{i \mid \zeta[i:i+m-1] = \alpha\}$. We shall assume that no question is repeated twice in a series $\zeta[i_1] = ?$, $\zeta[i_2] = ?$, . . . , so that an algorithm may be represented by a decision tree with q-ary branchings at each query. (For basic definitions regarding q-ary trees, see Knuth [5].) An example of such a decision tree is shown in Figure 1, with $\Sigma = \{a,b,c\}$, α = bb and $n = 3$. The queries are enclosed in circles, and an answer $A(\alpha,\zeta)$ is attached to each leaf of the ternary tree.

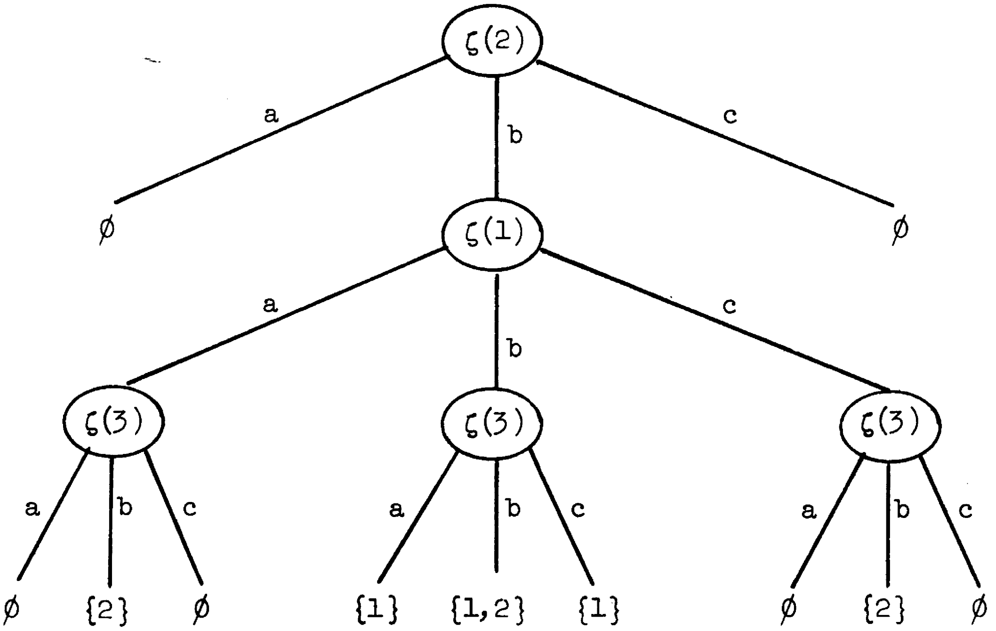

Figure 1. A pattern-matching algorithm for $\Sigma = \{a,b,c\}$, α = bb and $n = 3$.

For given α and n , let $\mathcal{T}(\alpha,n)$ be the set of all decision trees for the pattern-matching problem. For any $T \in \mathcal{T}(\alpha,n)$, let $h_T(\zeta)$ be the number of queries asked by T for the input text string $\zeta \in \Sigma^n$. In Figure 1, we have for example $h_T(\zeta) = 3$ if $\zeta = abc$. The average (or expected) number of queries asked of a random text string by T is

$$\bar{h}_T = \frac{1}{q^n} \sum_{\zeta \in \Sigma^n} h_T(\zeta) \quad . \tag{1}$$

Since the number of text strings that reach the same leaf as ζ does is $q^{n-h_T(\zeta)}$, an alternative form of (1) is

$$h_T = \sum_{\text{leaf } v} \frac{d_T(v)}{q^{d_T(v)}} \tag{2}$$

where $d_T(v)$ is the distance (path length) from the root to node v . The average-case complexity $c(\alpha,n)$ of the pattern-matching problem with respect to α and n , then, is the minimum expected number of queries asked by any algorithm. That is,

$$c(\alpha,n) = \min_{T \in \mathcal{J}(\alpha,n)} \bar{h}_T \quad . \tag{3}$$

In [4] it was shown that, for any pattern $\alpha \in \Sigma^m$,

$$n/m < c(\alpha,n) < \text{constant} \bullet \ \blacksquare\square\bullet\square♑_q \ (m+1)\rceil / m \ . \tag{4}$$

It was also conjectured in [4] that, for infinitely many m , there exists $\alpha \in \Sigma^m$ such that $c(\alpha,n) > a\,n\lceil \log_q(m+1)\rceil/m$ for some constant a when n is sufficiently large. The main results of the present paper are the following theorems. The first theorem strengthens the upper bound given by formula (4) in the range $m \leq n \leq 2m$. The second theorem proves the conjecture mentioned above in a somewhat stronger form. In fact, Theorem 2 as stated below follows from a result (Theorem 4) proved in Section 5, which implies that the lower bound in Theorem 2 actually holds even for the "best-case" complexity. (See Section 5.1 for precise formulations.)

Definition. For $n \underline{>} m > 0$, let

$$f_1(m,n) = \lceil \log_q((n-m)/\ln(m+1) + 2) \rceil \text{ , and}$$

$$f_2(m,n) = n \lceil \log_q(m+1) \rceil / 2m \text{ .}$$

Define

$$f(m,n) = \begin{cases} f_1(m, n) & \text{if } m \leq n \leq 2m \text{ ,} \\ f_2(m,n) & \text{if } n > 2m \text{ .} \end{cases}$$

Theorem 1. There exists a constant a_1 such that, for any $q \underline{>} 2$, $\alpha \in \Sigma^m$, and $n \geq m > 0$, we have $c(\alpha,n) \leq a_1 f(m,n)$.

Theorem 2. There exists a constant a_2 such that, for any $q \geq 2$ and $m > 0$, there exists a set of strings $L \subseteq \Sigma^m$ satisfying

(i) $|L| \geq \left(1 - \frac{1}{m^9}\right) q^m$, and

(ii) for each $\alpha \epsilon L$, $c(\alpha,n) \geq a_2 f(m,n)$ for all $n \underline{>} m$.

In the definition of $f_i(m,n)$ above, the constants +1 and +2 , as well as the ceiling function $\lceil\,\rceil$ are just to insure that $f(m,n)$ is well-defined and bounded away from zero. Indeed, as we have defined it, $f(m,n) > 1$ for all $n > m$. Notice also that, when $n \approx 2m$, we have $f_1(m,n) \approx f_2(m,n) \approx \lceil \log_q (m+1) \rceil$. Figure 2 shows the qualitative behavior of $f(m,n)$ as a function of n when m is fixed.

Remark All the constants implied in the "O", "Ω", and "θ" notations, as well as other constants used in the paper (e.g. al, a_2 above), are absolute constants (independent of q, n, m, etc.).

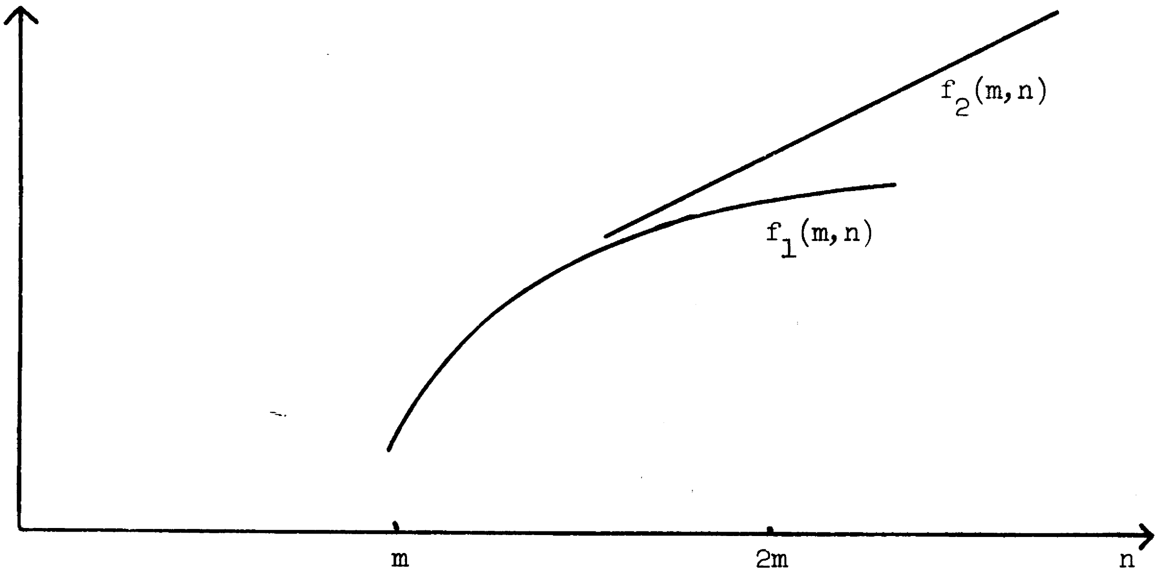

Figure 2. The behavior of $f(m,n)$ for a fixed m .

3. Analysis of a Simple Algorithm.

In [4, Section 8], a simple algorithm for pattern-matching was described and shown to have an average running time of $O(n\lceil\log_q(m+1)\rceil/m)$. This establishes the desired up-per bound of Theorem 1 for $n \geq 2m$. In fact, since $f_2(m,n) = O(f_1(m,n))$ for $(1+\varepsilon)m < n \leq 2m$ where ε is any positive constant, Theorem 1 is true as long as n-m is at least a positive fraction of m. Therefore, in our discussions of upper bounds in Sections 3 and 4, we shall only be concerned with the case when n-m is less than some fraction of m, say $n-m < m/2$.

In Section 3.1, we first show that the above-mentioned algorithm of [4] (which-we shall refer to as the Basic Algorithm from now on) has a tight bound of $O(\lceil\log(n-m+2)\rceil)$ for the present range $n-m \leq m/2$. Note that this performance is still weaker than the $O(f_1(m,n))$ bound we wish to establish. In Section 3.2 we then introduce an alternative, and perhaps less obvious way for looking at the behavior of the Basic Algorithm. This new analysis will shed light on how a better algorithm may be devised. In Section 4 we then present, an improved algorithm and show that it achieves the time bound $O(f_1(m,n))$.

3.1 The Basic Algorithm and Its Analysis.

We begin with a description of the Basic Algorithm from [4], slightly modified to fit our purpose.

The Basic Algorithm. Let $\alpha \in \Sigma^m$ be the pattern. For any input text string $\zeta \in \Sigma^n$, the algorithm examines ζ character by character, in the order $\zeta[m],\zeta[m-1],\ldots,\zeta[1],\zeta[m+1],\zeta[m+2],\ldots,\zeta[n]$. The algorithm halts as soon as enough information is known to determine $A(\alpha,\zeta)$, the set of all (leftmost) positions of α's occurrences in ζ.

We will show that for the case $n-m \leq m/2$, the Basic Algorithm only looks at $O(\lceil \log_q(n-m+2) \rceil)$ characters on the average. This analysis is a refinement of the approach used in [4] to prove the general $O(n\lceil \log_q(m+1) \rceil / m)$ bound for the same algorithm. The idea is that, for a random text string, it is unlikely that any occurrence of α will happen, and the Basic Algorithm can rule out that possibility after examining $O(\lceil \log_q(n-m+2) \rceil)$ characters on the average.

Definition. Let $d=n-m$. For any $\zeta \in \Sigma^n$, write $\zeta = \beta_1 \zeta' \beta_2$ where $\|\beta_1\| = \|\beta_2\| = d$. The substring ζ' of ζ will be called the prime substring of ζ, denoted always by ζ'. Let n' be the length of ζ'. Note that $n' = n-2d = m-d \geq m/2$ as $d \leq m/2$.

It is easy to see that any occurrence of α in ζ must cover the prime substring ζ' (see Figure 2(a)). Thus, for $A(\alpha,\zeta)$ to be nonempty, ζ' must be a substring of α. In fact, if we write $\zeta' = a_1 a_2 \cdots a_{n'}$, then for $A(\alpha,\zeta)$ to be nonempty, any segment $\zeta'[i:j] = a_i a_{i+1} \cdots a_j$ of ζ' must be a substring of $\alpha[i,j+d]$ (see Figure 2(b)). Based on this observation, let us divide ζ' into consecutive segments of length r, such that $\zeta' = \delta \zeta_t \zeta_{t-1} \cdots \zeta_1$ where $\|\zeta_k\| = r$ for $1 \leq k \leq t$ and $\|\delta\| < r$. Then, in order for ζ to contain any occurrence of α, each ζ_k for $1 < k < t$ must occur in a certain substring α_k of α with $\|\alpha_k\| = \|\zeta_k\|+d = r+d$. The probability that this condition is met by all the ζ_k's of a random text string ζ is $\leq [(d+1)/q^r]^t$. Now, what the Basic Algorithm does is to examine the substrings $\zeta_1, \zeta_2, \ldots, \zeta_t$ in sequence, hence the probability P_k that it will ever look beyond ζ_k is $\leq [(d+1)/q^r]^k$ for $1 \leq k \leq t$. It follows that the average number of characters $\bar{h}$ examined by the Basic Algorithm is

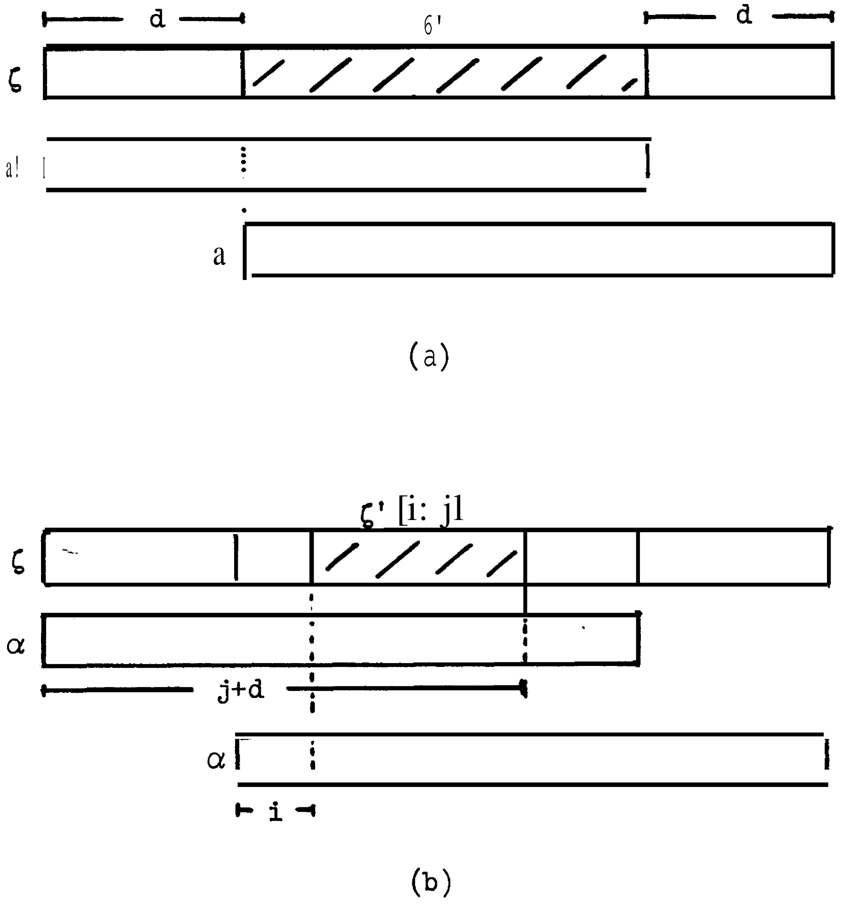

Figure 2. The prime substring ζ' of ζ relative to α.

$$\bar{h} \leq r(1+P_1+P_2+\dots+P_{t\ 1}) + m\cdot P_t \quad . \tag{5}$$

We now choose $r = 2\lceil \log_q (d+2) \rceil$, so that $P_k < [(d+1)/(d+2)^2]^k < [1/(d+2)]^k$.

Then,

$$
\begin{aligned}
\bar{h} &\le 2r + m\cdot(d+2)^{-\lfloor n'/r \rfloor} \\
&\le 2r + m\cdot O\left(2^{-m/(4\lceil \log_q(d+2)\rceil)}\right) \\
&= 2r + O(1) \\
&= O(\lceil \log_q(d+2)\rceil). \qquad (6)
\end{aligned}
$$

We now show that this bound is tight to within a constant factor, i.e., there exist patterns α for which $\Omega(\lceil \log_q(d+2)\rceil)$ characters on the average are examined by the Basic Algorithm. Again let $r = 2\lceil \log_q(d+2)\rceil$. We can **assume** that $d \ge 4q^2$ and $r \ge 4$. Consider a pattern $\alpha \in \Sigma^m$ which contains as a suffix the concatenation of all possible strings of length $\lfloor r/4 \rfloor$. That is, $\alpha = \eta\varphi$, where $\varphi = \varphi_u\varphi_{u-1}\cdots\varphi_1$, $u = q^{\lfloor r/4 \rfloor}$ and $\{\varphi_1,\varphi_2,\ldots,\varphi_u\}$ contains every possible string of length $\lfloor r/4 \rfloor$. Note that such α exists since, with $\lfloor r/4 \rfloor < \lceil \log_q(d+2)\rceil/2$, the total length of φ is

$$
\begin{aligned}
\|\varphi\| &= \lfloor r/4 \rfloor \cdot q^{\lfloor r/4 \rfloor} \\
&\le \frac{\lceil \log_q(d+2)\rceil}{2}(q(d+2))^{1/2} \\
&< \frac{\lceil \log_2(d+2)\rceil}{2}\;\frac{d^{1/4}}{\sqrt{2}}\;(d+2)^{1/2} \\
&< d \qquad (7)
\end{aligned}
$$

for $d > 4q^2 \ge 16$. For such an α , the Basic Algorithm cannot halt after examining the first block of $\lfloor r/4 \rfloor$ characters $\zeta[m],\zeta[m-1],\ldots,\zeta[m-\lfloor r/4 \rfloor+1]$. The reason is the following: if j is the index such that $\varphi_j = \zeta[m-\lfloor r/4 \rfloor+1 :m]$, then it is still possible for ζ to contain an occurrence of α exactly where $\zeta[m-\lfloor r/4 \rfloor+1:m]$ matches with φ_j (see Figure 3). Note that the fact $\|\varphi\| < d$ is used here. We have thus shown that for such a pattern α , the algorithm must look at more than $\lfloor r/4 \rfloor$ characters.

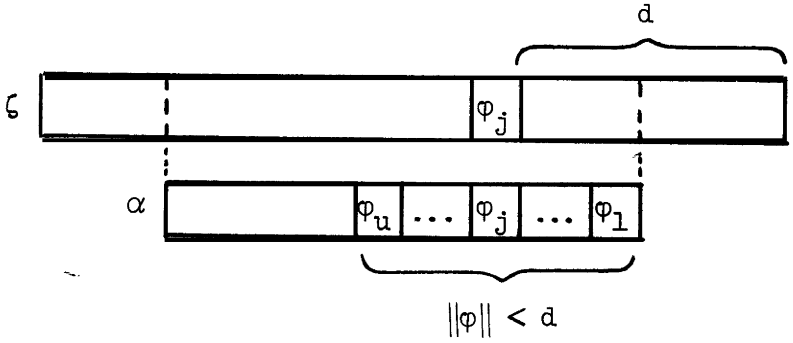

Figure 3. ζ may contain α between the dotted lines.

We have demonstrated that, for any pattern $\alpha \in \Sigma^m$, and a random text string $\zeta \in \Sigma^n$, the Basic Algorithm examines an average of o($\lceil \log_q(n-m+2)$ 1 characters assuming $n-m \leq m/2$. Furthermore, there exists a $\in \Sigma^m$ such that $\Omega(\lceil \log_q(n-m+2) \rceil$ characters are examined even in the best case for the Basic Algorithm. Thus, to achieve the better time bound of $O(f_1(m,n))$, the algorithm has to be improved even beyond its best-case performance.

3.2 A Closer Look at the Basic Algorithm.

In this subsection we give an alternative proof that the Basic Algorithm examines at most $O(\lceil \log_q (d+2) \rceil)$ characters on the average. This analysis may seem less straightforward than the previous one. However, it will provide new insight into the pattern-matching process, and help motivate the improved algorithm to be presented in the next section.

Let us refer to the decision tree corresponding to the Basic Algorithm. We will be interested in those nodes where a character of the prime substring ζ' is examined, i.e., those nodes at distance $t < n'$ from the root. Initially, before any query is asked, an occurrence of a may begin at any of the positions $1,2,\ldots,d+1$ in ζ. After the first character $\zeta[m] = a$ is examined, the feasible positions for α's occurrences in ζ is reduced from $D = \{1,2,\ldots,d+1\}$ to $D \cap R(m,a)$ where we use $R(i,a)$ for the set $\{j \mid \alpha[i-j+1] = a\}$. In general, for a node v at distance $t < n'$ from the root, if $\zeta[m] = a_0$, $\zeta[m-1] = a_1$, ... $\zeta[m-t+1] = a_{t-1}$ is the sequence of probes that led to v, then $D \cap \left(\bigcap_{k=0}^{t-1} R(m-k,\ a_k) \right)$ defines the set of positions in ζ where an occurrence of α is still feasible when computation reaches this point. We shall call $D \cap \left(\bigcap_{k=0}^{t-1} R(m-k,\ a_k) \right)$ the feasible set at v, and denote it by $F(v)$. (For $t = 0$, $F(\text{root}) = D$.) The size of $F(v)$ is called the weight of v, denoted by $w(v)$. We first show that the weight of an internal node v is equal to the total weights of v's sons, provided that the character examined by v is located inside the prime substring ζ'.

Definition. For an internal node v with query $\zeta[i] = ?$, let $\text{son}_a(v)$ where $a \in \Sigma$ denote the succeeding node corresponding to the outcome $\zeta[i] = a$.

Lemma 1. If v examines a character inside ζ', then

$$F(v) = \bigcup_{a\in\Sigma} (F(\text{son}_a(v)) \text{ and } F(\text{son}_a(v)) \cap F(\text{son}_b(v)) = \emptyset \text{ for } a \neq b .$$

Proof. Let $\zeta[i] = ?$ be the query raised at v . It is easy to see that the family of subsets $R(i,a) = (j \mid \alpha(i-j+1) = a]$, for $a \in \Sigma$, forms a partition of the set $\{1,2,\ldots,i\}$. It follows that, for any subset B of $\{1,2,\ldots,i\}$, $\{B \cap R(i,a) \mid a \in \Sigma\}$ forms a partition of B . Since $i \geq d+1$ by assumption, we have $F(v) \subseteq \{1,2,\ldots,d+1\} \subseteq \{1,2,\ldots,i\}$. Therefore the subsets $F(v) \cap R(i,a) = F(son_a(v))$, for $a \in \Sigma$, form a partition of $F(v)$. □

Lemma 3.2. If v examines a character inside ζ' , then $w(v) = \sum_{a \in \Sigma} w(son_a(v))$.

Proof. This follows immediately from Lemma 3.1. □

Note that Lemmas 3 .1 and 3.2 may not be true if v probes outside of ζ' , since we may have $F(son_a(v)) \cap F(son_b(v)) \neq \emptyset$.

Now, the probability that ζ will be examined outside of ζ' by the Basic Algorithm is quite small. In fact, it happens only if ζ' is a substring of α , which has probability less than $(d+1)/q^{n'}$.

Therefore, the cost of the Basic Algorithm is

$$\bar{h} = \sum_{\text{leaf } v} \frac{d(v)}{q^{d(v)}} = \sum_{\substack{\text{leaf } v \\ d(v) \leq n'}} \frac{d(v)}{q^{d(v)}} + \sum_{\substack{\text{leaf } v \\ d(v) > n'}} \frac{d(v)}{q^{d(v)}} \qquad (8)$$

where the second term s_2 is bounded by $m(d+1)/q^{n'} = O(1)$. To study the first term s_1 in (8), we shall use the weight function w . Remember that, when we follow a path in the decision tree from the root, as soon as $w(v) = 0$ the computation terminates. This fact, together with Lemma 3.2, will allow us to bound the quantity $\sum\limits_{\substack{\text{leaf } v \\ d(v) \leq n'}} d(v)/(q^{d(v)})$ by

$\log_q(w(\text{root}))+$ constant .

Definition. Let T be a finite q-ary tree. Assume each node v (internal or leaf) of T is assigned a non-negative integer w(v) such that

(i) $w(v) = \sum\limits_{i=1}^{q} w(\text{son}_i(v))$ for any internal node v ,

(ii) if w(v) = 0 then r is a leaf.

We call such a T a weighted q-ary tree. The initial weight of T is defined to be w(root) , and the terminal weight of T is $t(T) = \sum\limits_{\text{leaf } v} d(v)/(q^{d(v)})$, where d(v) is as usual the distance from root to node v .

Definition. Let $\tau_q(W) = \text{l.u.b.}\{t(T) \mid T$ is any weighted q-ary tree with initial weight $W\}$. (Let $\tau_q(0) = 0$.)

Theorem A. $\tau_q(W) = \lfloor \log_q W \rfloor + 1 + \frac{W}{4^{\lfloor \log_q W \rfloor}} \frac{1}{q-1}$, for $W \geq 1$.

[Proved in Appendix A.]

Corollary. $\tau_q(W) \leq \lfloor \log_q$ WJ $+3$, for $W \geq 1$.

(For a related result about optimal digital search trees with n leaves, see Knuth [5], Sec. 6.3, exercise 37.)

Clearly now, since w(root) = d+1 for the decision tree of the Basic Algorithm, we have

$$s_1 = \sum_{\substack{\text{leaf } v \\ d(v) \leq n'}} \frac{d(v)}{q^{d(v)}} \leq \tau_q(d+1) \leq \lfloor \log_q(d+1) \rfloor + 3 \quad . \qquad (9)$$

Therefore, $\bar{h} = s_1 + s_2 \leq \lfloor \log_q(d+1) \rfloor + O(1) = O(\lceil \log_q(d+2) \rceil)$, the same result as we showed in Section 3.1.

3.3 Discussions.

What have we gained by the more involved analysis in Section 3.2? Firstly, we notice that the $O(\lceil \log_q(d+2) \rceil)$ behavior is not restricted to the Basic Algorithm. Lemmas 3.1 and 3.2 are true not only for the decision tree corresponding to the Basic Algorithm, but also for an arbitrary decision tree, as long as the character examined at v lies inside ζ' . Therefore, the same analysis that led to (8) and (9) for $\bar{h}$ applies to any algorithm which first examines the substring ζ' of ζ , and halts as soon as $A(\alpha, \zeta) = \emptyset$ can be decided. Hence, the following family of algorithms all have an $O(\lceil \log_q(d+2) \rceil)$, upper bound.

Generalized Basic Algorithm.

```
G ← {d+1,d+2,...,m} .
While G ≠ ∅ do
     begin pick any i∈G and examine ζ[i];
          if it is determined that A(α,ζ) = ∅ then stop;
          G ← G-(i);
     end;
Examine ζ[i] for i ∈ {1,2,...,d} ∪ {m+1,m+2,...,n} in any order.
```

Secondly, the successful use of $F(v)$ as a measure of progress for the computation hints on the design of a better algorithm, explicitly exploiting the present $F(v)$ to decide where to probe next. An improved algorithm based on this idea will be given in the next section.

We conclude this section by discussing the following generalization of the Basic Algorithm. Let Λ_n be the set of all permutations on $(1,2,\ldots,n)$. Let α be any pattern of length m and $\lambda \in \Lambda_n$. We consider the following algorithm.

Algorithm - (λ,α) . For any input text string $\zeta \in \Sigma^n$, examine the characters in the order $\zeta[\lambda(1)], \zeta[\lambda(2)], \ldots, \zeta[\lambda(n)]$. Halt as soon as all occurrences of α in ζ can be determined.

The Basic Algorithm is essentially the use of **Algorithm** - (λ,α) , with a particular permutation λ for all α . We have seen that there exists a for which the Basic Algorithm examines on the average $\Omega(\lceil \log_q(n-m+2)\rceil)$ characters. Is it possible to improve over the Basic Algorithm simply by choosing a different λ ? The following theorem answers this question in the negative.

Theorem 3. Let $0 < m \leq n \leq 2m$. For any $\lambda \in \Lambda_n$, there exists an $\alpha \in \Sigma^m$ such that Algorithm- (λ,α) examines an expected $\Omega(\lceil \log_q(n-m+2)\rceil)$ characters for a **random** text string in Σ^n .

The proof of this result follows naturally from a counting technique to be developed in Section 5. We shall, therefore, delay the proof to Section 5.4. There we shall actually show a stronger result: for large d , most $\alpha \in \Sigma^m$ have the desired property required by Theorem 3.

4. An Improved Algorithm.

We will construct an algorithm whose performance is $O(f_1(m,n))$ for $d = n-m \leq m/2$. Without loss of generality, we assume $m > 16$, The crucial observation is the following. Suppose we are performing a Generalized Basic Algorithm. After a number of characters have been examined, assume we find outselves reaching a node v with $w(v) \approx (\log_q m)/2$. Suppose that at this time the set G still has $|G| \geq m/4$. We claim that it is possible to finish the computation, examining only $O(1)$ additional characters on the average, with a different strategy. Notice that in contrast, the analysis in Section 3.2 (Theorem A) only guarantees a $O(\log_q w(r))$-= $O(\log_q \log_q m)$ bound if we don't change strategy. Let us now prove the claim.

Let v be a node as described above, with $|F(v)| = w(v) \leq (\log_q m)/2$ and the present $|G| \geq m/4$. Consider all the positions $i \in G$ that we may choose to examine at this node v. By Lemma 3.1, any $i \in G$ would induce an (ordered) partition $\{F(v) \cap R(i,a) \mid a \in \Sigma\}$ of $F(v)$ into q parts. Denote this partition by $n(i)$. Note that there are only $q^{w(v)} \leq \sqrt{m}$ possible partitions of $F(v)$ all together. Let us divide G into $q^{w(v)}$ equivalence classes by the induced partitions; that is, i and j in G are equivalent if and only if $\pi(i) = n(j)$. Since $|G| \geq m/4$, few elements are in an equivalent class consisting of a single element. Indeed, if we arrange the equivalence classes as $E_1, E_2, \ldots, E_s, E_{s+1}, \ldots, E_{q^{w(v)}}$, so that $|E_k| \geq 2$ if and only if $1 \leq k \leq s$, then we have $\sum_{k=1}^{s} |E_k| \geq \frac{1}{4} m - \sqrt{m}$, which is positive assuming $m > 16$. Now, the key to a faster algorithm is contained in the following lemma.

Lemma **4.1**. Let i and j be two distinct elements in E_k , where $1 \leq k \leq s$. If $\zeta[i] \neq \zeta[j]$, **then** ζ does not contain any occurrence of a .

Proof. **Assume** $\zeta[i] \neq \zeta[j]$, **and** ζ does contain α as a substring. Let $a = \zeta[i]$, $b = \zeta[j]$, and suppose $\zeta[\ell]$ is a feasible starting position for pattern α . Since i and j are in G , both $\zeta[i]$ and $\zeta[j]$ lie **within** the prime **substring** ζ' . Therefore, $\alpha[i-\ell+1] = a$ and $\alpha[j-\ell+1] = b$. But this implies that in partition $\pi(i)$ we have $\ell \in F(v) \cap R(i,a)$, while in partition $\pi(j)$ we have $I \in F(v) \cap R(j,b)$. This contradicts the **assumption** that $\pi(i)$ and $\pi(j)$ are the same ordered partition of F(v) . □

As the string ζ is **initially** random, the probability that $\zeta[i] = \zeta[j]$ for $i \neq j$ is only 1/q . Thus, it is advantageous to examine $\zeta[i]$ **and** $\zeta[j]$ for $i,j \in E_k$, which have probability 1-1/q to be different, and would thereby terminate the computation with answer $A(\alpha, \zeta)$ ▪ $\emptyset$. This suggests the following procedure:

Procedure Cleanup (G,F);

comment: G is the set of remaining unprobed positions in $\{d+1,d+2,\ldots,m\}$, and F is the current feasible set.

1. **Examine** characters $\zeta[i]$ for $i \in E_1$ one by one, then for $i \in E_2$ one by one , . . . , then for $i \in E_s$ one by **one.** Halt as soon as it is found that $\zeta[i] \neq \zeta[j]$ with $i,j \in E_k$ for some k .

2. Examine $\zeta[i]$ for $i \in \left(G - \bigcup_{k=1}^{s} E_k \right) \cup \{1,2,\ldots,d\} \cup \{m+1,m+2,\ldots,n\}$ in any **order.**

Analysis of Cleanup. Take t elements $i_1, i_2, \ldots, i_t$ of an equivalence class E_k, the probability that $\zeta[i_1] = \zeta[i_2] = .** = \zeta[i_t]$ is $1/q^{t-1}$. Thus the probability P that in step 1, the t+1 -st element of E_{k+1} will be examined is

$$P = \frac{1}{q^{\left(\sum_{j=1}^{k} |E_j| - k\right) + \varepsilon(t)}} \qquad \text{where } s(t) = \begin{cases} t-1 & \text{if } t > 1 \\ 0 & \text{if } t = 0 \,. \end{cases}$$

Since $\sum_{j=1}^{k} |E_j| - k \geq \sum_{j=1}^{k} |E_j| / 2$, and $s(t) \geq (t-1)/2$, we have

$$P \leq \frac{1}{q^{\left\lceil \left(\sum_{j=1}^{k} |E_j| + t - 1\right)/2 \right\rceil}} \,.$$

Therefore, the probability that I characters will be read in step 1 is no more than $1/q^{\lceil (\ell-2)/2 \rceil}$. The cost of step 2 is bounded by $n \leq 2m$, and it is executed with probability $\leq 1/q^{\lceil (u-1)/2 \rceil}$, where $u = \sum_{k=1}^{s} |E_k|$. Hence the total expected cost of Cleanup is bounded by

$$\sum_{\ell=1}^{U} \frac{1}{q^{\lceil (\ell-2)/2 \rceil}} + \frac{2m}{q^{\lceil (u-1)/2 \rceil}}$$

$$\leq O(1) + \frac{2m}{q^{\left\lceil \frac{1}{2}\left(\frac{m}{4} - \sqrt{m} - 1\right)\right\rceil}}$$

$$= o(1) \,. \qquad (10)$$

This proves our claim. We can now state our new pattern-matching algorithm.

Algorithm PM.

```
1.  $G \leftarrow \{d+1, d+2, \ldots, m\}$ ;
    $F \leftarrow \{1, 2, \ldots, d+1\}$ ;
2.  while $(|F| > (\log_q m)/2) \wedge (|G| \geq m/4)$ do
        begin pick any $i \in G$, examine $a = \zeta[i]$;
            if all occurrences of $\alpha$ can be determined then stop;
            $G \leftarrow G - \{i\}$;
            $F \leftarrow F \cap R(i,a)$;
        end;
3.  if $G < m/4$ then examine in any order the remaining characters of $\zeta$
    as needed, and halt.
4.  if $F \leq (\text{logq } m)/2$ then call Cleanup(G,F) to finish the computation.
```

To analyze the cost of Algorithm PM, let p_2 , p_3 , p_4 be the respective probability that steps 2, 3, 4 will be executed, and let h_2 , h_3 , h_4 be the average number of characters examined in steps 2, 3, 4, respectively, once they are executed. Then

$$h_{PM} = p_2h_2 + p_3h_3 + p_4h_4 . \qquad (11)$$

From the analysis of Cleanup, we know that $h_4 = O(1)$. The probability that step 3 is reached is bounded by the probability that the following happens (see Figure 4):

$$(\exists j)\left[(0 \leq j \leq d) \wedge \left(\bigwedge_{k=1}^{t} \alpha(j + i_k) = \zeta'(i_k)\right)\right] \qquad (12)$$

where $\{i_1, i_2, \ldots, i_t\}$ is the set of positions in substring ζ' that were examined in step 2, and $t > n' - \frac{m}{4} \geq \frac{m}{4}$. Therefore $p_3 \leq (d+1)/q^t \leq (d+1)/(q^{m/4})$, and $p_3h_3 = O(1)$. We shall now show that

$$h_2 = O(f_1(m,n)) . \qquad (13)$$

This till prove $\bar{h}_{PM} = O(f_1(m,n))$, and hence Theorem 1.

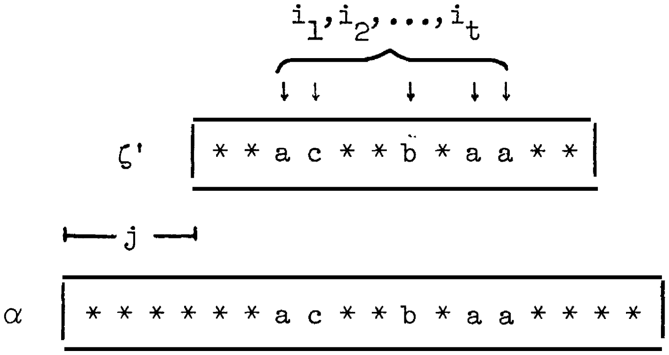

Figure 4. Matching ζ' with α.

Proof of (13). Let u be a positive integer. A weighted q-ary tree with initial weight W and cut value u is the same as in the definition of a weighted q-ary tree with initial weight W , except that condition (ii) is replaced by

(ii)' if $w(v) < u$, then v is a leaf.

Thus for $u = 1$, it reduces to the original definition.

Definition. Let $\tau_q(W,u) = \text{l.u.b.}\{t(T) | T$ is any weighted q-ary tree with initial weight W and cut value $u\}$.

Theorem B. $\tau_q(W,u) = \tau_q(\lfloor W/u \rfloor)$.

[Proved in Appendix B.]

Now, suppose we draw a decision tree for Algorithm PM beginning from the top, but only going as far down as step 2 of the algorithm is done.

If we designate these exit points from step 2 as "leaves", then clearly what we have is a weighted q-ary tree with initial weight $W = d+1$ and cut value $u = \lceil (\log_q m)/2 \rceil$, since condition (ii)' is satisfied. Therefore, the cost of step 2 satisfies

$$h_2 \leq \tau_q(d+1, \lceil (\log_q m)/2 \rceil)$$

If $d+1 < \lceil (\log_q m)/2 \rceil$, then $h_2 = 0$. Otherwise, from Theorems A, B,

$$h_2 \leq \log_q\left(\frac{d+1}{(\log_q m)/2}\right) + O(1)$$

$$= \log_q\left(\frac{d+1}{\ln m}\right) + \log_q \ln q + o(1)$$

$$= \log_q((d+1)/\ln m) + O(1) .$$

Thus, in both cases,

$$h_2 = O(f_1(m,n)) .$$

This completes the proof of Theorem 1.

5. Lower Bounds to the Complexity of Pattern-Matching.

We shall prove Theorem 2 by showing the existence of a set of "hard" patterns for which not only there is not any algorithm with good average behavior, but in fact there is not any algorithm with good best-case behavior. In Section 5.1, we define the concept of a "certificate", and carry out some preliminary reductions for the proof of Theorem 2. Section 5.2 proves a central lemma, and in Section 5.3 we complete the arguments for the lower bound. In Section 5.4 we prove Theorem 3 using a similar argument.

5.1 Preliminary Discussions.

For any ℓ, $1 \leq \ell \leq n$, let $S_n(\ell)$ be the set of strings in $(\Sigma \cup \{*\})^n$ with exactly $n-\ell$ *'s. For each $\varphi \in S_n(\ell)$, let $I(\varphi)$ be the set of those strings in Σ^n that agree with φ except in positions where φ has *'s. For example, let $\Sigma = \{0,1\}$ and $\varphi = {*}00{*}1 \in S_5(3)$, then $I(\varphi) = (00001, 00011, 10001, 10011\}$.

Let $\alpha \in \Sigma^m$ be a pattern. A string $\varphi \in S_n(\ell)$ is a certificate (of length ℓ) for α, if all elements in $I(\varphi)$ contain α in exactly the same set of positions. That is, $A(\alpha, \zeta_1) = A(\alpha, \zeta_2)$ for any $\zeta_1, \zeta_2 \in I(\varphi)$.

Definition. Let $g(\alpha,n)$ be the minimum length of a certificate for α, i.e.,

$$g(\alpha,n) = \min\{\ell \mid \exists\ \varphi \in S_n(\ell) \text{ such that cp is a certificate for } \alpha\}.$$

Let T be a decision tree that locates all occurrences of a in text strings from Σ^n. It is easy to see that any path in T from the root to a leaf must have length at least $g(\alpha,n)$. In fact, let

$\zeta[i_1] = a_1$, $\zeta[i_2] = a_2$, ... $\zeta[i_\ell] = a_\ell$ be the sequence of characters examined along the path, then $\varphi \in S_n(\ell)$ is a certificate for α where $\varphi[i_k] = a_k$ for $1 < k \leq \ell$, and $\varphi[j] = *$ otherwise. Thus, no algorithm can halt before examining $g(\alpha,n)$ characters even in the best case.

Lemma 5.1. $c(\alpha,n) \geq g(\alpha,n)$ for all α, n .

We shall prove the following strengthened version of Theorem 2.

Theorem 4. There exists a constant a_2 such that, for any $q \geq 2$ and $m > 0$, there exists a set of strings $L \subseteq \Sigma^m$ satisfying

(I) $\quad L \geq \left(1 - \frac{1}{m^9}\right) q^m$, and

(II) $\quad$ for each $\alpha \in L$, $g(\alpha,n) \geq a_2 f(m,n)$ for all $n \geq m$.

Before proceeding, we would like to make one more reduction.

Lemma 5.2. Let $n > 2m$, then $g(\alpha,n) \geq \left\lfloor \frac{n}{2m} \right\rfloor g(\alpha,2m)$.

Proof. For any string $\zeta \in \Sigma^n$, we write it as

$$\zeta = \zeta_1 \zeta_2 \cdots \zeta_{\lfloor n/2m \rfloor} \beta$$

where $|\zeta_j| = 2m$ for $1 \leq j \leq \lfloor n/2m \rfloor$. Similarly, we write $\varphi = \varphi_1 \varphi_2 \cdot m0\, \varphi_{\lfloor n/2m \rfloor} \eta$ for any $\varphi \in S_n(\ell)$. If φ is a certificate for α in Σ^n , then each φ_j must be a certificate for α_j in Σ^{2m} . (Note that the reverse may not be true.) Thus $g(\alpha,n) > \lfloor n/2m \rfloor g(\alpha,2m)$. □

This lemma allows us to reduce condition (II) of Theorem 4 to the following:

(II)' for each $\alpha \in L$, $g(\alpha,n) \geq a_2 f_1(m,n)$ for $m < n \leq 2m$.

This is so because $g(\alpha,2m) \geq a_2 f_1(m,2m)$ implies $g(\alpha,n) \geq \lfloor n/2m \rfloor g(\alpha,2m) \geq \lfloor n/2m \rfloor \cdot a_2 \lceil \log_q(m/(\ln(m+1)+2)) \rceil \geq a_2' f_2(m,n)$ for some $a_2' > 0$.

The next two subsections are devoted to a proof of Theorem 4.

5.2 The Counting Lemma.

A certificate φ for α is called a negative certificate if it disproves the containment of α as a substring, i.e., if $A(\alpha,\zeta) = \emptyset$ for all $\zeta \in I(\varphi)$. We first observe the fact that any certificate shorter than the pattern itself must, be a negative certificate.

Fact. Let $\alpha \in \Sigma^m$ be a pattern. If $\varphi \in S_n(\ell)$ is a certificate for a and $\ell < m$, then φ is a negative certificate for a .

Proof. Since φ does not check as many as m non-* characters, it is impossible for φ to certify the occurrence of α at any position in $\zeta \in I(\varphi)$. Therefore, it must be that $A(\alpha,\zeta) = \emptyset$, □

The next lemma is essential to the proof of Theorem 4. It says that not many patterns in Σ^m can share a common certificate which is short.

Definition. For any $\varphi \in S_n(\ell)$, where $1 \leq \ell \leq n$, let $\rho_m(\varphi)$ be the set of all patterns in Σ^m for which φ is a certificate. That is,

$$\rho_m(\varphi) = \{\alpha \mid \alpha \in \Sigma^m \text{ and } \varphi \text{ is a certificate for } \alpha\} .$$

The Counting Lemma, Let $1 < \ell < m < n$, and $\varphi \in S_n(\ell)$. Then

$$|\rho_m(\varphi)| \leq \left(1 - \frac{1}{q^\ell}\right)^{\left\lceil \frac{d}{\ell^2} \right\rceil} \cdot q^m \quad \text{where } d = n-m .$$

Proof. Let $1 \leq i_1 < i_2 < \cdot ** < i_\ell < n$ be the positions where φ has a non-* character. For $0 \leq j \leq d$, define

$$B_j = \{b \mid b \in \{1,2, \ldots, m\} \text{ and } j+b = i_t \text{ for some } 1 \leq t \leq \ell\} .$$

(See Figure 5.) Clearly $|B_j| \leq \ell$ for $1 \leq j \leq d$. Also, for any $\alpha \in \mathcal{P}_m(\varphi)$, since φ is a negative certificate by Fact, there must exist an $i \in B_j$ for each j such that $\alpha[i] \neq \varphi[j+i]$. Now we show that we can find $J \subseteq \{0,1,\ldots,d\}$, $|J| = \lceil d/\ell^2 \rceil$, such that $B_{j_1} \cap B_{j_2} = \emptyset$ for $j_1 \neq j_2$ in J.

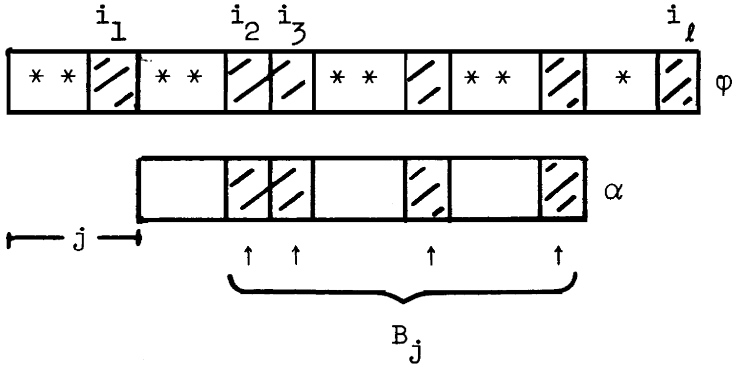

Figure 5. Definition of B_j for the Counting Lemma.

We find J by a "greedy" procedure. Let $j_1 = 0$. Inductively, j_s is obtained by finding the smallest j such that B_j is disjoint from $B = B_{j_1} \cup B_{j_2} \cup \cdots \cup B_{j_{s-1}}$. We claim that this procedure allows us to find at least $\lceil d/\ell^2 \rceil$ such sets. In fact, we shall show that $j_s \leq \ell^2(s-1)$. The key observation is that B contains at most $\ell(s-1)$ elements. We claim that at least one of the sets in $\mathcal{F} = \{B_0, B_1, \ldots, B_{a^2(s-1)}$ is disjoint from B. If not, for each r, $0 \leq r \leq \ell^2(s-1)$, let

(b,i_t) be a conflict where $b \in B_r \cap B$ and $r+b = i_t$ for some $1 \leq t \leq \ell$. The total number of such pairs is no more than $|B|\cdot \ell \leq \ell^2(s-1)$. But we have $\ell^2(s-1)+1$ sets in the family $\mathfrak{F}$, a contradiction.

To prove the lemma, consider a random string from Σ^m. For each $j \in J$, the probability that there exists some $i \in B_j$ with $\alpha[i] \neq \varphi[j+i]$ is $1 - 1/q^{|B_j|}$. Since all the sets B_j for $j \in J$ are disjoint, the probability that this holds for all j is

$$\prod_{j \in J}\left(1 - \frac{1}{q^{|B_j|}}\right) \leq \left(1-\frac{1}{q^p}\right)^{|J|}$$

$$\leq \left(1 - \frac{1}{q^\ell}\right)^{\left\lceil \frac{d}{\ell^2}\right\rceil} .$$

Since each $\alpha \in \rho_m(\varphi)$ must satisfy this condition, the lemma follows. □

5.3 Proof of Theorem 4.

In this subsection we complete the proof of Theorem 4. Roughly, the idea is to use the Counting Lemma to bound the number of patterns in Σ^m that have any "short" certificate.

Definition. Let x be a positive number such that (i) $x \geq 256$ and (ii) $y > (\lg y)^{12}$ for all $y \geq x$.

Lemma 5.3. Let $m + xq^4 \ln(m+1) \leq n \leq 2m$, $\ell = \left\lceil \frac{1}{2}\log_q\left(\frac{n-m}{\ln m}\right)\right\rceil$, and $\rho = \bigcup_{\varphi \in S_n(\ell)} \rho_m(\varphi)$. Then $|\rho| \leq \frac{1}{m^\ell} q^m$

Note. The assumption in the lemma ensures $m > 5$, thus In $m > 1$.

Proof. Clearly $\ell < m$. By the Counting Lemma, we have for each $\varphi \in S_n(\ell)$,

$$|\rho_m(\varphi)| \le \left(1 - \frac{1}{q^\ell}\right)^{\frac{d}{\ell^2}} \cdot q^m \quad .$$

Therefore,

$$|\rho| \le |S_n(\ell)| \cdot \left(1 - \frac{1}{q^\ell}\right)^{\frac{d}{\ell^2}} \cdot q^m$$

$$= \binom{n}{\ell} q^\ell \cdot \left(1 - \frac{1}{q^\ell}\right)^{\frac{d}{\ell^2}} \cdot q^m$$

$$\le (n \;\; q)^\ell \cdot e^{\frac{d}{\ell^2} \ln\left(1 - \frac{1}{q^\ell}\right)} \cdot q^m \qquad (14)$$

Since $n < 2m$, and $\ln(1 - q^{-\ell}) \le -q^{-\ell}$, (14) leads to

$$|\rho| \le (2m \;\; q)^\ell \exp\left(-\frac{d}{\ell^2 q^\ell}\right) \cdot q^m$$

$$= q^m \cdot \exp\left(-\left(\frac{d}{\ell^2 q^\ell} - \ell \cdot \ln(2m \;\; q)\right)\right) \quad . \qquad (15)$$

Fact. $\dfrac{d}{\ell^2 q^\ell} \ge 2\ell \cdot \ln(2m \;\; q)$. (16)

[Proved in Appendix C.]

Formula (15) then implies,

$$|\rho| \le q^m \bullet \quad \exp(44nm) = \frac{1}{m^\ell} q^m \; .$$

This proves the lemma. □

We now finish the proof of Theorem 4. As discussed in Section 5.1, we can assume that $n \le 2m$. We can assume that $m \ge xq^{20} \ln(m+1)$. Otherwise, $f(m,n) = \left\lceil \log_q\left(\frac{n-m}{\ln(m+1)} + 2 \right) \right\rceil = O(1)$, and we can choose $L = \Sigma^m$ to satisfy the conditions in Theorem 4.

For each n , $m+xq^{20} \ln(m+1) < n \le 2m$, let

$$\wp^{(n)} = \bigcup_{\varphi \in S_n(\ell_n)} \wp_m(\varphi) \quad , \text{ where } \quad I_n = \left\lceil \frac{1}{2} \log_q\left(\frac{n-m}{\ln m} \right) \right\rceil \ge 10 \ . \tag{17}$$

By Lemma 5.2, we have

$$|\wp^{(n)}| \le \frac{1}{m^{\ell_n}} q^m \le \frac{1}{m^{10}} \, {}^{m}q \ . \tag{18}$$

We define L as follows.

$$L = \Sigma^m - \bigcup_n \wp^{(n)} \quad , \text{ where the union is taken over}$$

$$m+xq^{20} \ln(m+1) \le n \le 2m \ . \tag{19}$$

Now we need only check that L has the properties specified in Theorem 4.

(I) $$|L| = q^m - \left| \bigcup_n \wp^{(n)} \right| \ge q^m - m \cdot \frac{1}{m^{10}} \, {}^{m}q = \left(1 - \frac{1}{m^9} \right) \cdot q^m \ .$$

(II)' We shall prove, for each $\alpha \in L$, $g(\alpha,n) \ge a_2 f(m,n)$ for all $m \le n \le 2m$, and an absolute constant a_2 .

There are two cases:

(a) If $m + xq^{20} \ln(m+1) \le n \le 2m$, then $\alpha \notin \wp^{(n)}$ by definition of L .

Thus,

$$g(\alpha,n) > \left\lceil \frac{1}{2} \log_q\left(\frac{n-m}{\ln m} \right) \right\rceil \ge a_2' \left\lceil \log_q\left(\frac{n-m}{\ln(m+1)} + 2 \right) \right\rceil = a_2' \cdot f(m,n)$$

for some absolute constant a_2' .

(b) If $m \leq n < m + xq^{20} \ln(m+1)$, then

$$f(m,n) = \left\lceil \log_q\left(\frac{n-m}{\ln(m+1)} + 2\right)\right\rceil = O(1) \text{ , and}$$

$$g(\alpha,n) \geq \alpha_2''\cdot f(m,n) \quad \text{for some absolute constant } a_2'' \text{ .}$$

Thus, in both cases, we have verified property (2)'.

Therefore, the set L defined by (19) satisfies the conditions (I) and (II)' set in Theorem 4. This completes the proof of Theorem 4.

Remark. In the condition I I; $\geq q^m\left(1 - \frac{1}{m^9}\right)$ of Theorem 4, the choice of the factor $1 - \frac{1}{m^9}$ is somewhat arbitrary. In fact, we can replace it by any factor $1 - \frac{1}{m^b}$ where b is any fixed positive number. Then, in the proof, we need to divide cases according to whether n is greater than or less than $m + xq^{2(b+1)} \ln(m+1)$. The resulting constant a_2 in the theorem will be different.

5.4 Proof of Theorem 3.

We can assume that $d = n-m > \max\{q^4, x\}$, where x is defined as in Section 5.3. Otherwise the bound $\Omega(\lceil \log_q(d+2)\rceil) = \Omega(1)$, and any pattern $\alpha \in \Sigma^n$ will meet the conditions in Theorem 3.

By assumption, $m \leq n \leq 2m$, and $\lambda \in \Lambda_n$. Let $\ell = \lceil (\log_q(n-m+2))/2 \rceil$. Recall that $S_n(\ell)$ is the set of strings of length n over $\Sigma \cup \{*\}$ with ℓ non- * characters. Let $H \subseteq S_n(\ell)$ be defined by

$$H = \{\varphi \mid \varphi \in S_n(\ell);\ \varphi[\lambda(i)] \in \Sigma \text{ for } 1 \leq i \leq \ell, \text{ and } \varphi[j] = * \text{ for all other } j) \text{ ,} \qquad (19)$$

Clearly, there are exactly q^ℓ elements in H , i.e., $|H| = q^\ell$.

Now, let $\wp'$ be the set of patterns $\alpha \in \Sigma^m$ such that **Algorithm-**(λ,α) halts for some text string in less than or equal to ℓ steps. For any $\alpha \in \wp'$, clearly there must be a $\varphi \in H$ such that $\alpha \in \wp_m(\varphi)$, Thus,

$$\wp' \subseteq \bigcup_{\varphi \in H} \wp_m(\varphi) \quad . \tag{20}$$

By the Counting Lemma, we have

$$|\wp_m(\varphi)| \leq (1-q^{-\ell})^{\frac{d}{\ell^2}} \cdot q^m \quad .$$

Therefore,

$$|\wp'| \leq |H| \cdot (1-q^{-\ell})^{\frac{d}{\ell^2}} \cdot q^m = q^{\ell} \cdot (1-q-')^{\frac{d}{\ell^2}} \cdot q^m \quad . \tag{21}$$

Since every a in $\Sigma^m - \wp'$ meets the conditions set in Theorem 3, we need only show that $q^{\ell} \cdot (1-q^{-\ell})^{\frac{d}{\ell^2}} < 1$. Now,

$$q^{\ell} \cdot (1-q^{-\ell})^{\frac{d}{\ell^2}} \leq q^{\ell} \cdot \exp\left(-\frac{d}{\ell^2 q^{\ell}}\right) \quad . \tag{22}$$

By using the definition of ℓ and the condition $d > q^4$, we obtain after some algebraic manipulations

$$q^{\ell} \cdot (1-q^{-\ell})^{\frac{d}{\ell^2}} \leq (d+2)^{3/4} \cdot \exp\left(-\frac{(d+2)^{1/4}}{2(\lg(d+2))^2}\right) \quad . \tag{23}$$

The right hand side of (23) can be shown to be less than 1 when $d \geq x$, This proves Theorem 3.

Remark. The right hand side of (23) is $O(\exp(-d^{1/5}))$ for large d. We have in fact shown that, for any fixed $\lambda \in \Lambda_n$, **Algorithm-**(λ,α) has to examine $\Omega(\lceil \log_q(d+2) \rceil)$ characters in the best case for all but a $O(\exp(-d^{1/5}))$ fraction of the patterns $\alpha \in \Sigma^m$.

An open question: Is the following statement true?

Let $0 < m \leq n \leq 2m$. For any $\alpha \in \Sigma^m$, there exists a $\lambda \in \Lambda_n$ such that **Algorithm-**(λ,α) examines $O(f_1(m,n))$ characters on the average for a random text string of length n .

References

[1] A. V. Aho and M. J. Corasick, "Fast Pattern Matching: An Aid to Bibliographic Search," Communications ACM 18 (1975), 333-340.

[2] A. V. Aho, D. S. Hirschberg, and J. D. Ullman, "Bounds on the Complexity of the Longest Common Subsequence Problem," 23 (1976), 1-12.

[3] R. S. Boyer and J. S. Moore, "A Fast String Searching Algorithm," Stanford Research Institute Technical Report 3 (March 1976).

[4] D. E. Knuth, J. H. M rris, and V. R. Pratt, "Fast Pattern Matching in Strings," SIAM J. on Computing 6 (1977), 323-350.

[5] D. E. Knuth, The Art of Computer Programming, Vol. 3, Sorting and Searching, Addison-Wesley (1973).

[6] R. L. Fives-t, "On the Worst-Case Behavior of String-Searching Algorithms," Information Processing Letters, to appear.

Appendix A: Proof of Theorem A.

In this appendix, we shall prove the following theorem used in Section 3.2 in the paper. For definitions and notations, see Section 3.2.

Theorem A. Let $q \geq 2$ be an integer. Then

$$\tau_q(W) = \lfloor \log_q W \rfloor + 1 + \frac{W}{q^{\lfloor \log_q W \rfloor}} \frac{1}{q-1} , \quad \text{for } W > 1 , \qquad (A1)$$

We first derive same properties of the function f defined below.

Definition. Let $q \geq 2$ be an integer, we define a function f by

$$f(0) = 0 \qquad (A2)$$

$$f(W) = \lfloor \log_q W \rfloor + 1 + \frac{W}{q^{\lfloor \log_q W \rfloor}} g-\overset{1}{1} , \quad \text{for } W \geq 1 .$$

Let $g(W) = f(W+1) - f(W)$, for all integers $W > 0$.

Property 1. $$g(W) = \frac{1}{q-1} \frac{1}{q^{\lceil \log_q (W+1) \rceil - 1}} \qquad \text{for } W \geq 0 .$$

Property 2. $g(W) \geq g(W')$ if $0 \leq W \leq W'$.

Property 2 follows from property 1, which can be verified directly,

Property 3. The function f satisfies the following recurrence relation:

$$f(0) = 0 \quad , \qquad (A3)$$

$$f(W) = 1 + \frac{1}{q} \sum_{i=1}^{q} f\left(\left\lfloor \frac{W+i-1}{q} \right\rfloor \right) \quad \text{for } W \geq 1 . \qquad (A4)$$

Proof of Property 3. Equation (A3) is true by definition. To prove (A4), let

$$W = tq + s \,, \qquad \text{with } 0 \leq s < q \;. \tag{A5}$$

Then

$$\left\lfloor \frac{W+i-1}{q} \right\rfloor = \begin{cases} t & \text{if } 1 \leq i \leq q-s \;, \\ t+1 & \text{if } q-s+1 \leq i \leq q \;. \end{cases}$$

Thus, we need only prove

$$f(tq+s) = 1 + \frac{1}{q}\left((q-s)\cdot f(t) + s\cdot f(t+1)\right) \,, \text{ i.e.,}$$

$$f(tq+s) = 1 + f(t) + \frac{s}{q}\, g(t) \;. \tag{A6}$$

From the explicit forms of f and g are given in (A2) and Property 1, it is not difficult to verify (A6). This implies Property 3. □

Remark. We shall interpret (A4) as follows. Let us write $W = W_1 + W_2 + \bullet$ ⊠⊠▤♦$_q$ such that $|W_i - W_j| \leq 1$ for all i, j . Then

$$f(W) = 1 + \frac{1}{q} \sum_{i=1}^{q} W_i \;.$$

We are now ready to prove Theorem A. Let T be any weighted q-ary tree, and T_i be the sub-tree rooted at $\text{son}_i(\text{root})$, $1 \leq i \leq q$. Then the terminal weight of T satisfies

$$t(T) = 1 + \frac{1}{q} \sum_{i=1}^{q} t(T_i) \;.$$

This leads to the following equations:

$$\begin{cases} \tau_q(0) = 0 \\ \tau_q(W) = 1 + \frac{1}{q} \max\left\{ \sum_{i=1}^{q} \tau_q(W_i) \,\middle|\, \text{integer } W_i \geq 0 \,,\, \sum_{i=1}^{q} W_i = W \right\} \end{cases} \tag{A7}$$

for $W > 1$.

Clearly (A7) determines $\tau_q(W)$ uniquely, Therefore, in order to prove (Al), we need only prove that f(W) satisfies (A7). Because of Property 3, it suffices to prove

$$\max\left\{ \sum_{i=1}^{q} f(W_i) \;\middle|\; W_i \geq 0 \,,\; \sum_i W_i = W \right\} = \sum_{i=1}^{q} f\left(\left\lfloor \frac{W+i-1}{q} \right\rfloor \right) \,. \qquad (A8)$$

That is, the sum $\sum_i f(W_i)$ achieves a maximum value when all the Wi differ from each other by at most 1 . This can be demonstrated as follows. If, for some i and j , $W_i \geq W_j+2$, we make the changes $W_i \leftarrow W_i-1$ and $W._j \leftarrow W_j+1$. The value of $\sum_i f(W_i)$ is increased by an amount $f(W_j+1) + f(W_i-1) - f(W_i) - f(Wj) = g(W_j) - g(W_i-1)$, which is non-negative because of Property 1. It can be shown that the value of $\sum_{i,j} |W_i - W_j|$ is decreased by at least 2 by such a transformation. Therefore, by a finite number of such transformations, all the Wi will be within 1 to each other. The value of $\sum f(W_i)$ is at least as great as the initial value before the transformations, This proves (A8), and hence Theorem A.

Appendix B: Proof of Theorem B.

[See Section 4 for notations.]

Theorem B. Let $q > 2$ and $u > 1$. . Then

$$\tau_q(W, u) = \tau_q(\lfloor W/u \rfloor), \quad \text{for all } W \geq 1 . \qquad \text{(B1)} ,$$

By definition, a q-ary tree with initial weight $W < u$ and cutoff u can only consist of a single leaf. Therefore,

$$\tau_q(W,u) = 0 \quad \text{for } 0 < W < u . \qquad \text{(B2)}$$

The following facts can be established by deriving a recurrence relation on $\tau_q(W, u)$ similar to (A7), and performing some simple reductions.

$$\tau_q(W,u) = \frac{q}{q-1} \quad \text{for } u < W < 2u . \qquad \text{(B3)}$$

$$\tau_q(W,u) = 1 + \frac{1}{q} \max \left\{ \sum_{i=1}^{q} \tau_q(W_i,u) \;\middle|\; 0 \leq W_i < W \text{ for } 1 \leq i \leq q, \text{ and } \sum_{i=1}^{q} W_i = W \right\} \quad \text{for } W > u . \qquad \text{(B4)}$$

We shall now use (B4) in an inductive proof of formula (B1).

Consider q , u as fixed, and the induction is on variable W . By (B2) and (B3), the formula (B1) is true for $0 \leq W < 2u$. Now, assume $W > 2u$, and we have proved (B1) for all smaller values of W . We shall prove that it is also true for W .

By (B4), we have

$$\tau_q(W,u) = 1 + \frac{1}{q} \max \left\{ \sum_{i=1}^{q} \tau_q(W_i,u) \;\middle|\; 0 \leq W_i < W \text{ for } 1 \leq i \leq q, \text{ and } \sum_{i=1}^{q} Wi = W \right\} .$$

By inductive hypothesis, $\tau_q(W_i, u) = \tau_q(\lfloor W_i/u \rfloor)$ for $1 \leq i \leq q$.

Therefore,

$$\tau_q(W,u) = 1 + \frac{1}{q} \max \left\{ \sum_{i=1}^{q} \tau_{q'}(\lfloor W_i/u \rfloor) \,\middle|\, 0 \leq W_i < W, \ \sum_i W_i = \right\} . \tag{B5}$$

We complete the proof in two steps:

(i) $$\tau_q(W,u) \leq \tau_q(\lfloor W/u \rfloor) . \tag{B6}$$

Proof. It is not difficult to verify that τ_q is a non-decreasing function of its argument. Noting that $\sum_i \lfloor W_i/u \rfloor \leq \lfloor W/u \rfloor$. Then

$$1 + \frac{1}{q} \sum_{i=1}^{q} \tau_q(\lfloor W_i/u \rfloor) \leq \tau_q\left(\sum_i \lfloor W_i/u \rfloor \right) \leq \tau_q(\lfloor W/u \rfloor) ,$$

where we have used (A4) in the first step. This proves (B6) because of (B5). □

(ii) $$\tau_q(W,u) \geq \tau_q(\lfloor W/u \rfloor) . \tag{B7}$$

Proof. Let $W = tu + v$, where $0 \leq v < u$. Define

$$W_i = \begin{cases} \left\lfloor u\frac{t+i-1}{q} \right\rfloor & \text{for } 1 \leq i \leq q-1 , \\ \left\lfloor \frac{t+q-1}{q} \right\rfloor u + v & \text{for } i = q . \end{cases} \tag{B8}$$

Then

$$\lfloor W_i/u \rfloor = \left\lfloor \frac{t+i-1}{q} \right\rfloor \qquad \text{for } 1 \leq i \leq q . \tag{B9}$$

From (B5), we have

$$\tau_q(W,u) > 1 + \frac{1}{q} \sum_{i=1}^{q} \tau_q(\lfloor W_i/u \rfloor)$$

$$= 1 + \frac{1}{q} \sum_{i=1}^{q} \tau_q\left(\left\lfloor \frac{t+i-1}{q} \right\rfloor \right) \quad . \qquad \text{(B10)}$$

In Appendix A, we have shown that the right-hand side of B(10) is equal to $\tau_q(t)$. (See (A4); remember that $\tau_q(W) = f(W)$.) Therefore, (B10) leads to

$$\tau_q(W,u) \geq \tau_q(t) = \tau_q(\lfloor W/u \rfloor) \ . \ \square$$

We have now proved that $\tau(W,u) = \tau(\lfloor W/u \rfloor)$. This completes the inductive step in the proof of Theorem B.

Appendix C. Proof of Formula (16).

We shall prove formula (16) used in Section 5.3. For easy reference, we repeat all the notations and assumptions.

Notations. $d = n-m$, $I = \left\lceil \frac{1}{2} \log_q \left(\frac{n-m}{\ln m} \right) \right\rceil$. The number x is a positive number such that (i) $x > 256$, and (ii) for all $y \geq x$, $Y \geq (\lg y)^{12}$.

Assumptions. $q \geq 2$, and $m \geq d \geq xq^4 \ln(m+1) > 0$.

We wish to prove:

$$\frac{d}{\ell^2 q^\ell} \geq 2\ell \cdot \ln(2mq) \quad . \tag{C1}$$

Proof. We shall prove

$$\frac{d}{q^\ell \cdot \ln(2mq)} \geq 2\ell^3 \quad . \tag{C2}$$

Now $\ell \leq 1 + \frac{1}{2} \log_q \left(\frac{d}{\ln m} \right)$, hence

$$q^\ell \leq q \left(\frac{d}{\ln m} \right)^{1/2} \quad . \tag{C3}$$

Also $m \geq d \geq q^4$. Thus, $m \geq 2q$ because $q \geq 2$.

$$\ln(2mq) < \ln(m^2) = 2 \ln m \quad . \tag{C4}$$

From (C3) and (C4), we have

$$\frac{d}{q^\ell \cdot \ln(2mq)} \geq \frac{d}{q(d/\ln m)^{1/2} \cdot 2 \ln m} = \frac{1}{2q} \left(\frac{d}{\ln m} \right)^{1/2} . \tag{C5}$$

Therefore, (C2) will be proved, if we can show

$$\frac{1}{2q} \left(\frac{d}{\ln m} \right)^{1/2} \geq 2\ell^3$$, i.e.,

$$\frac{d}{\ln m} \geq 16q^2\ell^6 \quad . \tag{C6}$$

Notice that, by assumption, $\frac{d}{\ln m} \geq q^4$, hence $\log_q\left(\frac{d}{\ln m}\right) \geq 4$. Therefore

$$\ell \leq 1 + \frac{1}{2}\log_q \ A_{\ln \bar{m}} \quad L \ \log_q \ \frac{d}{\ln m} \quad . \tag{C7}$$

Because of (C7), we can prove (C6) if the following is true.

$$\frac{d}{\ln m} \geq 16q^2\left(\log_q\left(\frac{d}{\ln m}\right)\right)^6 . \tag{C8}$$

We shall now prove (C8) to complete the proof of (Cl).

By assumption, $\frac{d}{\ln m} \geq xq^4$.

(i) Since $x > 256$, we have

$$\left(\frac{d}{\ln m}\right)^{1/2} \geq (xq^4 \ {}^{1/2}_{,} \geq 16q^2 \quad . \tag{C9}$$

(ii) Since $\frac{d}{\ln m} \geq x$, the following inequality is true. We have $\frac{d}{\ln m} \geq \left(\lg\left(\frac{d}{\ln m}\right)\right)^{12}$, which implies that

$$\left(\frac{d}{\ln m}\right)^{1/2} \geq \left(\log_q\left(\frac{d}{\ln m}\right)\right)^6 \quad . \tag{C10}$$

It follows from (C9) and (C10) that

$$\frac{d}{\ln m} \geq 16q^2\left(\log_q\left(\frac{d}{\ln m}\right)\right) \quad .$$

This proves (C8), and hence (Cl).