DARTMOUTH COLLEGE

REMITTANCE ADVICE

HANOVER, N.H. 03755
603/646-2435

ositing

740207

	ollege erence	VENDOR REFERENCE	GROSS AMOUNT	DISCOUNT AMOUNT	NET AMOUNT
			6.00		6.00

rec'd + sent reports May 25/89

Technical Report Secretary
Technical Report Distribution
Department of Computer Science
University of Waterloo
Waterloo, Ontario, CANADA N2L 3O1

BILL TO:/SHIP TO: DATE: April 4, 1989

Julie McIntyre
Baker Library - Serials Section
Dartmouth College
Hanover, NH 03755
U.S.A.

Quantity	Report No.	Author/Title	Cost
1	CS-88-35	Errata: Theory of Computation/ Derick Wood	2.00
1	CS-88-36	Fast String Matching with k Mismatches/Baeza-Yates	2.00
1	CS-88-37	New Algorithm for Pattern Matching.../Baeza-Yates	2.00
		Total Due:	$ 6.00

If you would like to order any reports please forward your order, along with a cheque or international bank draft payable to the **Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, to the Research Report Secretary.**

Please indicate your current mailing address.

MAILING ADDRESS:

UMIACS Business Office
A.V. Williams Bldg.#115, Room 2129
University of Maryland
College Park, Maryland ~~U.S.~~A.
Attn: Ursula Gedra

RECEIVED
APR 26 7 27 AM '89
ACCOUNTS PAYABLE

U.M. Advanced Computer Studies
571
01-1-31450-4318

2M 62574
order form
4-26-89
0000 9905-1

6.00
4-26-89

O

sent
88-35,
36 + 37
May 18/89

5/1/89

University of Waterloo
Department of Computer Science
Waterloo, Ontario N2L 3G1
Research Reports 1988 (September to December)

CS-88-35 - ERRATA: THEORY OF COMPUTATION

AUTHOR: Derick Wood

ABSTRACT:

This report provides two lists of corrections for my textbook "Theory of Computation". The first list are those corrections that have been incorporated into the second printing. The second list are those corrections that will be incorporated into the third printing.

The second printing of the North American hardback edition was published in April, 1988, by John Wiley & Sons. It has ISBN Number 0-471-60351-1. I expect that the third printing will not be produced until late 1989; therefore, these lists will continue to be helpful for at least one more year.

PRICE:$2.00

CS-88-36 - FAST STRING MATCHING WITH k MISMATCHES

AUTHORS: Ricardo A. Baeza-Yates, Gaston H. Gonnet

ABSTRACT:

We describe and analyze three simple and fast algorithms for solving the problem of string matching with at most k mismatches. These are the naive algorithm, an algorithm based on the Boyer-Moore approach, and ad-hoc deterministic finite automata searching.

PRICE:$2.00

CS-88-37 - NEW ALGORITHM FOR PATTERN MATCHING WITH OR WITHOUT MISMATCHES

AUTHORS: Ricardo A. Baeza-Yates, Gaston H. Gonnet

ABSTRACT:

We introduce a family of simple and fast algorithms for solving the classical string matching problem, string matching with don't care symbols and complement symbols, and multiple patterns. We also solve the same problems allowing up to k mismatches. Among the features of these algorithms is that they are real time algorithms, that they don't need to buffer the input, and that they are suitable to be implemented in hardware.

PRICE:$2.00

CS-88-38 - A GLOBALLY CONVERGENT AUGMENTED LAGRANGIAN ALGORITHM FOR OPTIMIZATION WITH GENERAL CONSTRAINTS AND SIMPLE BOUNDS

AUTHORS:A.R. Conn, N.I.M. Gould, Ph.L. Toint

ABSTRACT:

The paper extends an algorithm for optimization with simple bounds (Conn, Gould and Toint, Siam Journal of Numerical Analysis 25, 433-460, 1988) to handle general constraints. The extension is achieved using an augmented Lagrangian approach. Global convergence is proved and it is established that a potentially troublesome penalty parameter is bounded away from zero.

PRICE:$2.00

CS-88-39 - THE COMPUTATIONAL STRUCTURE AND CHARACTERIZATION OF NONLINEAR DISCRETE CHEBYSHEV-PROBLEMS

AUTHORS:A.R. Conn, Y. Li

ABSTRACT:

We present the generalisation of some concepts in linear Chebychev theory to the nonlinear case. We feel these generalisations capture the inherent structure and characteristics of the best Chebychev approximation and that they can be usefully exploited in the computation of a solution to the discrete Chebychev problem.

Key Words. nonlinear Chebyshev approximation

Subject Classification. 41A50, 65D99, 65K05, 65K10

PRICE:$2.00

PURCHASE ORDER
UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742-5115

SHOW THIS ORDER NUMBER ON ALL SHIPMENTS, CORRESPONDENCE AND INVOICES.

S262574-P

REFERENCE BID NO. OR CONTRACT NO.	AMOUNT
	$6.00

REQ. NO.	FAS NO.	FISCAL YR.
571	01-1-31450/4318	88/89

FEI 000099051

TO
Department of Computer Science
University of Waterloo
Waterloo, Ontario, CANADA N2L3G1
Attn: Research Report Secretary

SHIP TO
University of Maryland S262574-P
A. V. Williams Bldg. #115, Rm. #2129
UMIACS Business Office
College Park, Maryland 20742
Attn: Ursula Gedra

[redacted]

Subject to purchase terms & conditions, furnish goods and/or services shown below.

DATE	DELIVER ON OR BEFORE	TERMS:	F.O.B.
4/21/89	6/15/89	Net	Destination

ITEM NO.	DESCRIPTION OF ARTICLES OR SERVICES	QUANTITY	UNIT OF QUANT.	TOTAL ESTIMATED COST	UNIT PRICE	TOTAL ACTUAL COST	SUBCODE
1.	CS-88-35-ERRATA: THEORY OF COMPUTATION Author: Derick Wood	1	ea.	~~2.00~~		2.00	
2.	CS-88-36-FAST STRING MATCHING WITH k MISMATCHES Authors: Ricardo A. Baeza-Yates, Gaston H. Gonnet	1	ea.	~~2.00~~		2.00	
3.	CS-88-37-NEW ALGORITHM FOR PATTERN MATCHING WITH OR WITHOUT MISMATCHES Authors: Ricardo A. Baeza-Yates, Gaston H. Gonnet	1	ea.	~~2.00~~		2.00	
					Total	$6.00	
					Canadian dollars		

Check for full amount attached.

ACCOUNTS PAYABLE RECEIVED
'89 APR 26 7 27 AM

ROUTING INSTRUCTIONS
VENDOR SHIP VIA US MAIL/UPS

QUESTIONS CONCERNING THIS ORDER SHOULD BE REFERRED TO THE BUYER: PERRY/mm

CANADIAN

PURCHASE TERMS AND CONDITIONS

NOTE: Terms & conditions continued on reverse side.

1. A separate invoice in TRIPLICATE for this purchase order or for each shipment thereon shall be rendered immediately following shipment. All copies of invoices must be forwarded directly to the Accounts Payable Department, South Administration Building, University of Maryland, College Park, Md. 20742.
2. The vendor's/contractor's Federal identification number or social security number must be included on the invoice.
3. This purchase order number must be shown on all related invoices, delivery memoranda, bills of lading, packages, and/or correspondence.

FAILURE TO COMPLY WITH THESE TERMS WILL RESULT IN THE INVOICE BEING RETURNED TO YOU.

UNIVERSITY OF MARYLAND
By Monique G. Perry 4/24/89
AUTHORIZED SIGNATURE

NOTE: THE UNIVERSITY OF MARYLAND IS EXEMPT FROM THE FOLLOWING TAXES:
1. State of Maryland Sales Tax by Certificate No. 30002563
2. District of Columbia Sales Tax by Exemption No. 9199 79411 01
3. Manufacturer's Federal Excise Tax Registration No. 52 730123K.

VENDOR'S COPY

To

From sent

Date April 19/89

memo

University of Waterloo

CS-86-36

Computer Resources Int'l

Corporate Technology Div.

Att: Hanne Albrechtsen

Bregnerodvej

144

DK 3460

Birkerod

Denmark

60808

Printing Requisition / Graphic Services

1. Please complete unshaded areas on form as applicable.
2. Distribute copies as follows: White and Yellow to Graphic Services. Retain Pink Copies for your records.
3. On completion of order the Yellow copy will be returned with the printed material.
4. Please direct enquiries, quoting requisition number and account number, to extension 3451.

TITLE OR DESCRIPTION: Fast String Matching with k Mismatches CS-88-36

DATE REQUISITIONED: March 27/89
DATE REQUIRED: ASAP
ACCOUNT NO.: 1 2 6 6 3 1 7 4 1

REQUISITIONER- PRINT: G. Gonnet
PHONE: 4460
SIGNING AUTHORITY

MAILING INFO – NAME: Sue DeAngelis DEPT.: C.S. BLDG. & ROOM NO.: DC 2314 [X] DELIVER / PICK-UP

NUMBER OF PAGES: 13
NUMBER OF COPIES: 50

TYPE OF PAPER STOCK: [ ] BOND [ ] NCR ___ PT. [ ] COVER [ ] BRISTOL [ ] SUPPLIED [ ] Alpac ~~Ivory~~ 140M

PAPER SIZE: [ ] 8½ x 11 [ ] 8½ x 14 [ ] 11 x 17 [ ] 10x14 Glosscoat 10 pt Rolland Tint

PAPER COLOUR: [ ] WHITE [X]
INK: [X] BLACK [ ]

PRINTING: [ ] 1 SIDE ___ PGS. [X] 2 SIDES ___ PGS.
NUMBERING: FROM ___ TO ___

BINDING/FINISHING: [X] COLLATING [ ] STAPLING [ ] HOLE PUNCHED [ ] PLASTIC RING

FOLDING/PADDING: 7x10 saddle stitched
CUTTING SIZE

Special Instructions

Beaver Cover

Both cover and inside in black ink please.

COPY CENTRE — OPER. NO. / BLDG. / MACH. NO.

DESIGN & PASTE-UP — OPER. NO. / TIME / LABOUR CODE

	OPER. NO.	TIME	LABOUR CODE
			D01
			D01
			D01

TYPESETTING — QUANTITY

	QUANTITY			
PAP000000				T01
PAP000000				T01
PAP000000				T01

PROOF

PRF				
PRF				
PRF				

NEGATIVES

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
FLM				C01
FLM				C01
FLM				C01
FLM				C01
FLM				C01

PMT

PMT				C01
PMT				C01
PMT				C01

PLATES

PLT				P01
PLT				P01
PLT				P01

STOCK

				001
				001
				001
				001

BINDERY

RNG				B01
RNG				B01
RNG				B01
MIS00000				B01

OUTSIDE SERVICES

$ ______ COST

TAXES – PROVINCIAL [ ] FEDERAL [ ]

GRAPHIC SERV. OCT. 85 482-2

CHECK NUMBER

04956347

PENNSTATE

ACCOUNTING OPERATIONS • 120 S. BURROWES ST.
UNIVERSITY PARK, PA. 16801

REMITTANCE ADVICE FOR **DEPT OF CO**

DATE **03-15-89**

ANY QUESTIONS CONCERNING A PAYMENT SHOULD INDICATE OUR CHECK AND VOUCHER NUMBERS AND BE ADDRESSED TO ACCOUNTING OPERATIONS.

PSU FED. I.D. #24-6000376

FORM S1-87 7/88

YOUR INVOICE NUMBER	VOUCHER NO.	OUR DEPARTMENT NO.	P.O. NO.	DISCOUNT	NET AMOUNT
	235048	228-28			4.00
					4.00**

sent reports
Mar. 27/89

CS-88-48 - SWITCH-LEVEL TESTABILITY OF THE DYNAMITE CMOS PLA

AUTHORS:B.F. Cockburn, J.A. Brzozowski

ABSTRACT:

Functional testing, as opposed to parametric testing, plays an important role in testing VLSI integrated circuits. However, it appears that designs are not always carefully analysed a priori to determine precisely which faults are *clean*, i.e. testable by logic means alone. The programmable logic array (PLA) is a popular circuit form used to implement a system of Boolean functions over a set of input variables. This report considers the testability of the dynamic CMOS PLA with respect to an extended set of switch-level faults models, namely: node faults, transistor stuck-opens and stuck-ons, interconnect breaks, ohmic shorts, and crosspoint faults. Single occurrences of each fault model are classified as either clean, unclean, or clean subject to conditions on the logical products and output functions computed by the PLA. Finally, a modified dynamic CMOS PLA design is described and its increased switch-level testability properties are stated.

PRICE:$2.00

If you would like to order any reports please forward your order, along with a cheque or international bank draft payable to the **Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, to the Research Report Secretary.**

Please indicate your current mailing address.

MAILING ADDRESS:

Laurie Hearn
The Pennsylvania State University
Department of Computer Science
333 Whitmore Laboratory
University Park, PA 16802

Enclosed is check for $4.00 for 1 copy of technical report #CS-88-36 and 1 copy of CS-88-37. Please mail to above address. Thank you.

CS-88-48 - SWITCH-LEVEL TESTABILITY OF THE DYNAMITE CMOS PLA

AUTHORS:B.F. Cockburn, J.A. Brzozowski

ABSTRACT:

Functional testing, as opposed to parametric testing, plays an important role in testing VLSI integrated circuits. However, it appears that designs are not always carefully analysed a priori to determine precisely which faults are *clean*, i.e. testable by logic means alone. The programmable logic array (PLA) is a popular circuit form used to implement a system of Boolean functions over a set of input variables. This report considers the testability of the dynamic CMOS PLA with respect to an extended set of switch-level faults models, namely: node faults, transistor stuck-opens and stuck-ons, interconnect breaks, ohmic shorts, and crosspoint faults. Single occurrences of each fault model are classified as either clean, unclean, or clean subject to conditions on the logical products and output functions computed by the PLA. Finally, a modified dynamic CMOS PLA design is described and its increased switch-level testability properties are stated.

PRICE:$2.00

If you would like to order any reports please forward your order, along with a cheque or international bank draft payable to the **Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, to the Research Report Secretary.**

Please indicate your current mailing address.

MAILING ADDRESS:

Laurie Hearn
The Pennsylvania State University
Department of Computer Science
333 Whitmore Laboratory
University Park, PA 16802

Enclosed is check for $4.00 for 1 copy of technical report #CS-88-36 and 1 copy of CS-88-37. Please mail to above address. Thank you.

Carleton University
Ottawa, Canada K1S 5B6

Feb. 14/89

Research Report Secretary
Dept. of Comp Sci.
Univ. of Waterloo.

I would appreciate your rushing me the following reports (cheque Enclosed)

CS-88-36 "Fast String Matching with k Mismatches" — Baeza-Yates & Gonnet

CS-88-37 "New algorithms for Pattern Matching with ~~out~~ or without mismatches" — Baeza-Yates & Gaston Gonnet

CS-88-44 A Study of Distributed Debugging — Cheung, Black, Manning

received cheque + sent reports IUTS Feb. 17/89

Yours Sincerely, Prof. B. PAGUREK

Department of Systems and Computer Engineering
C.J. Mackenzie Building □ (613) 564-3625

Printing Requisition/Graphic Services

20952

1. Please complete unshaded areas on form as applicable.
2. Distribute copies as follows: White and Yellow to Graphic Services. Retain Pink Copies for your records.
3. On completion of order the Yellow copy will be returned with the printed material.
4. Please direct enquiries, quoting requisition number and account number, to extension 3451.

TITLE OR DESCRIPTION
Fast String Matching with k Mismatches CS-88-36

DATE REQUISITIONED: Sept. 30/88
DATE REQUIRED: ASAP
ACCOUNT NO.: 1 2 6 6 3 1 7 4 1

REQUISITIONER- PRINT: G. Gonnet
PHONE: 4460
SIGNING AUTHORITY

MAILING INFO – NAME: Sue DeAngelis
DEPT.: C.S.
~~BLDG. & ROOM NO.~~ DC 2314
[X] DELIVER
[ ] PICK-UP

NUMBER OF PAGES: 13
NUMBER OF COPIES: 50

TYPE OF PAPER STOCK: [ ] BOND [ ] NCR ___ PT. [ ] COVER [ ] BRISTOL [ ] SUPPLIED [ ] Alpac Ivory ~~140M~~

PAPER SIZE: [ ] 8½ x 11 [ ] 8½ x 14 [ ] 11 x 17 [ ] 10x14 Glosscoat
~~10 pt Rolland Tint~~

PAPER COLOUR: [ ] WHITE [X] ___
INK: [X] BLACK [ ] ___

PRINTING: [ ] 1 SIDE ___ PGS. [X] 2 SIDES ___ PGS.
NUMBERING: FROM ___ TO ___

BINDING/FINISHING: [X] COLLATING [ ] STAPLING [ ] ___ HOLE PUNCHED [ ] PLASTIC RING

FOLDING/PADDING: 7x10 saddle stitched
CUTTING SIZE

Special Instructions

Beaver Cover

Both cover and inside in black ink please

COPY CENTRE — OPER. NO. | BLDG. | MACH. NO.

DESIGN & PASTE-UP — OPER. NO. | TIME | LABOUR CODE

OPER. NO.	TIME	LABOUR CODE
		D01
		D01
		D01

TYPESETTING — QUANTITY

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
PAP000000				T01
PAP000000				T01
PAP000000				T01

PROOF

PRF				
PRF				
PRF				

NEGATIVES

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
FLM				C01
FLM				C01
FLM				C01
FLM				C01
FLM				C01

PMT

PMT				C01
PMT				C01
PMT				C01

PLATES

PLT				P01
PLT				P01
PLT				P01

STOCK

				001
				001
				001
				001

BINDERY

RNG				B01
RNG				B01
RNG				B01
MIS00000				B01

OUTSIDE SERVICES

$ ___
COST

TAXES – PROVINCIAL [ ] FEDERAL [ ]

UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

Fast String Matching with k Mismatches

Ricardo A. Baeza-Yates
Gaston H. Gonnet

Data Structuring Group
Research Report
CS-88-36

September, 1988

Fast String Matching with k Mismatches

Ricardo A. Baeza-Yates
Gaston H. Gonnet

Data Structuring Group
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1 *

Abstract

We describe and analyze three simple and fast algorithms for solving the problem of string matching with at most k mismatches. These are the naive algorithm, an algorithm based on the Boyer-Moore approach, and ad-hoc deterministic finite automata searching.

1 Introduction

The problem of string matching with k mismatches consists of finding all ocurrences of a pattern of length m in a text of length n such that in at most k positions the text and the pattern have different symbols. The case of $k = 0$ is the well known string matching problem. We will assume that $0 \leq k < m$ and $m \leq n$ (otherwise the problem is trivial).

Landau and Vishkin [6] gave the first efficient algorithm to solve this particular problem. Their algorithm uses $O(k(n + m \log m))$ time and $O(k(n + m))$ space. While it is fast, the space required is unacceptable for practical purposes. Galil and Giancarlo [5] improved this algorithm to $O(kn + m \log m)$ time and $O(m)$ space. This algorithm is practical for small k. However, if $k = O(m)$ it is not so.

We present and analyse the naive or brute-force algorithm to solve this problem. While the worst case is $O(nm)$ time, the expected time is only $O(kn)$ without using any extra space. We also present a Boyer-Moore approach to the problem [3] that has the same complexity, but the probability of the worst case is much lower, using $O(m - k)$ extra space. Finally, we use

*The work of the first author was supported by a scholarship from the Institute for Computer Research and by the University of Chile and that of the second author by a Natural Sciences and Engineering Research Council of Canada Grant No. A-3353.

finite automata theory to solve the problem in time $O(m^{k+2} + n \log m)$ and $O(m^{k+2})$ space. This algorithm is better when k is comparable to m and m is not too big.

2 Naive algorithm

The naive algorithm is basically to try all possible positions and count the number of mismatches found. If more than k has been found we try the next position. When we reach the end of the pattern we report a match. Clearly, the worst case number of comparisons is $m(n-m+1)$.

Average Case Analysis

Let the text and the pattern be random strings of length n and m, respectively, over an alphabet of size $c > 1$. If the alphabet size is not known or not finite, we can replace c by n. The probability that two symbols, one from the pattern and one from the text, being equal is $p = 1/c$.

Let $\overline{C}_m^k$ be the average number of comparisons between the pattern and a text of length m to decide if there are at most k mismatches between both strings. Then

$$\overline{C}_m^k = \sum_{j=1}^{m} 1 \times P_{j-1,k} \ ,$$

where $P_{j,k}$ is the probability of finding at most k mismatches in the first j characters of the text (pattern), because we perform one comparison at position j with probability $P_{j-1,k}$.

Clearly $P_{j,k} = 1$ if $j \leq k$. We can define $P_{j,k}$ recursively as

$$P_{j,k} = pP_{j-1,k} + (1-p)P_{j-1,k-1}$$

The solution to this recurrence is

$$P_{k+j,k} = \begin{cases} 1 - (1-p)^{k+1} \sum_{i=0}^{j-1} \binom{k+i}{i} p^i & j > 0 \\ 1 & j \leq 0 \end{cases}$$

Hence for $m > k$ we have

$$\begin{aligned} \overline{C}_m^k &= k + 1 + \sum_{j=1}^{m-k-1} P_{k+j,k} \\ &= m - (1-p)^{k+1} \sum_{j=1}^{m-k-1} \sum_{i=0}^{j-1} \binom{k+i}{i} p^i \end{aligned}$$

But $\binom{k+i}{i} = (-1)^i\binom{-(k+1)}{i}$, then

$$\overline{C}_m^k = m - (1-p)^{k+1}\sum_{j=1}^{m-k-1}\sum_{i=0}^{j-1}\binom{-(k+1)}{i}(-p)^i$$
$$= m - (1-p)^{k+1}\sum_{i=0}^{m-k-2}(m-k-1-i)\binom{-(k+1)}{i}(-p)^i$$

Using the binomial theorem we obtain

$$\overline{C}_m^k = m - (1-p)^{k+1}\Big((m-k-1)(1-p)^{-(k+1)} - p(k+1)(1-p)^{-(k+2)}$$
$$- \sum_{i\geq m-k-1}(m-k-1-i)\binom{-(k+1)}{i}(-p)^i\Big)$$
$$= \frac{k+1}{1-p} + (1-p)^{k+1}\sum_{i\geq m-k-1}(m-k-1-i)\binom{k+i}{i}p^i$$
$$= \frac{k+1}{1-p} - O(p^{m-k}(1-p)^{k+1}m^k)$$

For a text of length n, we try $n-m+1$ times, that is

$$\bar{C}_n^k = \overline{C}_m^k(n-m+1) \leq \frac{c(k+1)}{c-1}n \leq 2(k+1)n$$

For $k=0$ (brute force string matching) we have

$$\bar{C}_n = \frac{1-p^m}{1-p}(n-m+1) = \frac{c}{c-1}\left(1-\frac{1}{c^m}\right)(n-m+1)$$

A result already obtained in [1].

Figure 1 shows the experimental results (100 trials) for an English text of size approximately 50K and an alphabet of size 32 (lowercase letters plus some other symbols). In the same graph the theoretical results are shown. The agreement is very good, and the differences are due to the fact that English text is not random [1].

For values of k closer to m it may be better to count matches rather than mismatches. That is, string matching with at least $m-k$ matches. In this case

$$\frac{\bar{C}_n^{m-k}}{n-m+1} = (m-k+1)c - O(p^{k+1}(1-p)^{m-k-1}m^{m-k})$$

Therefore the break-even point is $k \approx (m+2)(1-1/c)-1$, and this is greater or equal than $m/2$ for $c \geq 2$.

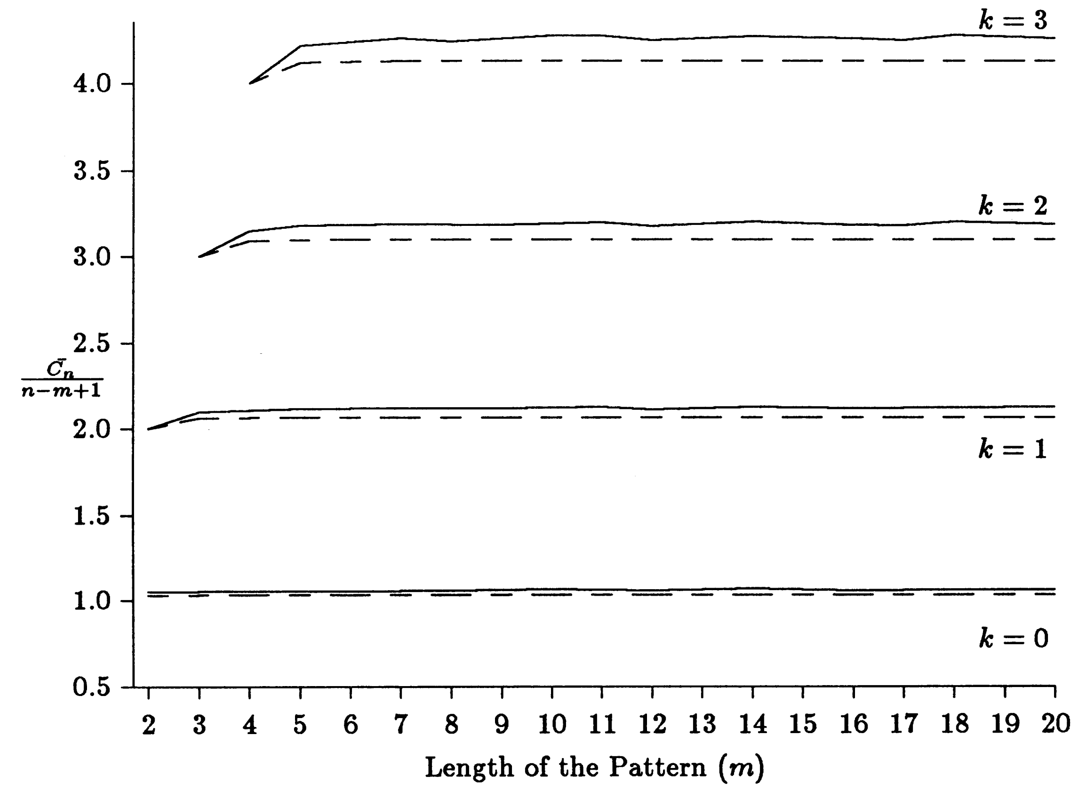

Figure 1: Theoretical results for random text (dashed line, $c = 32$) and experimental results in English text (Naive algorithm) for k from 0 to 3.

3 A Boyer-Moore approach

The idea in this method is to search from right to left until we find a match or too many mismatches. At this point, using a precomputed table, we decide how much we can shift the pattern [3]. The definition of this table follows.

Suppose that we have found a partial match of length j (from the right) with at most k mismatches such that the next character is another mismatch. We define s_j as the maximum shift such that overlapping two copies of the pattern shifted in ℓ characters for ℓ from 1 to $s_j - 1$ implies that at least there are $2k+1$ mismatches between both strings. This is to make sure that there are at least $k+1$ mismatches in the overlap (at most k mismatches are overriden by the mismatches in the partial match).

Clearly s_j for j from 0 to $2k$ is 1. To compute the other values of s_j we slide two copies of the pattern until we find less than $2k+1$ mismatches. For example, if all the characters of the text are different, then $s_j = m - 2k$

for $j > 2k$. Therefore, this procedure will be useful for $k < m/2$. Figure 2 shows an example.

```
pattern:  pointing
           pointing
            pointing
             pointing
              pointing
               pointing
   s[j]:  666633111
     j :  876543210
```

Figure 2: Example for the table s_j $(k = 1)$.

Clearly $s_{j+1} \geq s_j$, because if we slide s_j positions and we have a partial match of j or more elements, then we have at least $2k+1$ mismatches. Also, because we have more characters in the partial match, potentially we have more mismatches. Using this property, there exists a very simple algorithm to set up the table in $m(m-2k)$ worst case time (if all the characters are different only $2mk$ comparisons are necessary). With a slightly more complex algorithm (based in [4]) a $O(km)$ preprocessing time can be achieved for any pattern.

The worst case is $2kn$ for many patterns (still $O(mn)$ in general but only for periodic patterns) using $m-2k$ space. On the average this algorithm will be slightly better than the naive algorithm, improving for long patterns. In the best case, when all the characters are different, we have that the average shift is

$$\bar{S} = 1 + (m - 2k - 1)P_{2k+1,k}$$

(the shift is $m - 2k$ if we compare more than $2k + 1$ characters and this happen with probability $P_{2k+1,k}$; otherwise the shift is 1). Then, a lower bound for random text is given by

$$\bar{C}_n^k \geq \frac{\bar{C}_m^k}{\bar{S}}(n - m + 1)$$

In Figure 3 are shown the experimental results for English text and the lower bound for random text. For long patterns, this algorithm is clearly is better. Then, we have a $(k+1)n+O(mk)$ total expected time and $O(m-2k)$ space algorithm.

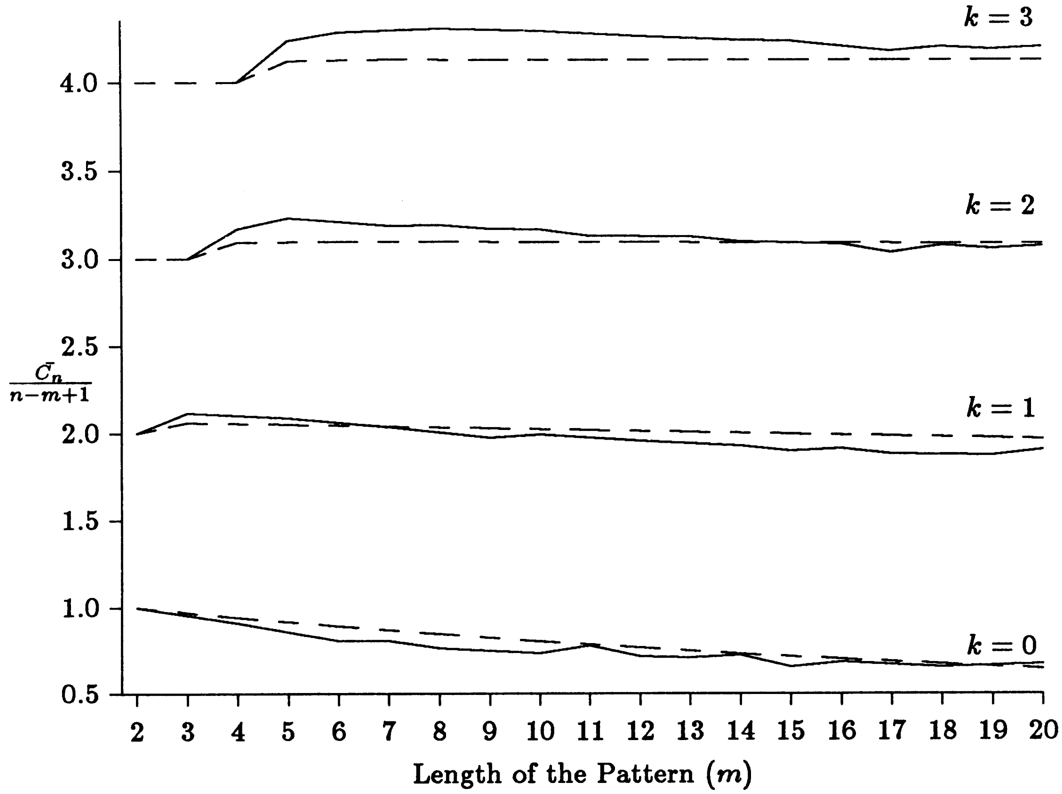

Figure 3: Theoretical results for random text (dashed line, $c = 32$) and experimental results in English text (Boyer-Moore approach) for k from 0 to 3.

4 Finite Automaton approach

The problem can be also stated in terms of a regular expression. For example if we are searching ab with one mismatch, then that set is described by $\theta^*(\theta b + a\theta)$, where θ denotes a don't care symbol. In general there are $\binom{m}{k}$ terms inside the parenthesis, and hence the length of the regular expression is $(m+1)\binom{m}{k}+1$ without counting parentheses. Let r be the regular expression that denotes our searching problem and let $p_m...p_1$ be the pattern. A slightly more compact representation is

$$r = \theta^*(S_m^k)$$

and

$$S_m^k = \begin{cases} \epsilon & m = 0 \\ p_m S_{m-1}^k & k = 0 \\ \theta S_{m-1}^{k-1} & k = m \\ p_m(S_{m-1}^k) + \overline{p_m}(S_{m-1}^{k-1}) & 0 < k < m \end{cases}$$

where $\overline{x}$ denotes any symbol with the exception of x. The number of terms in this case will be proportional to $\binom{m+1}{k+1}$.

From this recursive definition, it is very easy to construct the DFA that recognizes r. For example if $r = \theta^*(a\theta + \overline{a}b)$ we have

State	*Symbol*	*Regular expression left*	*State*	*Output*
		r	0	
0	a	$r+\theta$	1	
0	$\theta - a$	$r+b$	2	
1	a	$r+\theta+\epsilon$	1	*match*
1	b	$r+b+\epsilon$	2	*match*
1	$\theta - a - b$	$r+b+\epsilon$	2	*match*
2	a	$r+\theta$	1	
2	b	$r+b+\epsilon$	2	*match*
2	$\theta - a - b$	$r+b$	2	

Note that we have replaced accepting states (ϵ) by output, and then we do not have final states (and hence additional states). Figure 4 shows the resultant automaton for the example.

The regular expression after reading a symbol (p_i) is computed by the following formulas

$$r(p_i) = r + S_m^k(p_i)$$

and

$$S_m^k(p_i) = \begin{cases} S_{m-1}^k & p_m = p_i \text{ (if } m = 1, \textit{ match}) \\ S_{m-1}^{k-1} & k = m \text{ or } p_m \neq p_i (m > 0) \\ \emptyset & \text{otherwise} \end{cases}$$

In general we will have $O(m^{k+1})$ states and any state can have at most $\min(c, m+1)$ different transitions, where c is the alphabet size. Hence the size of the automaton is $O(m^{k+1}\min(c,m))$. Table 1 gives the number of states when all the characters in the string are different for small values of k and m.

Clearly we can build the DFA in $O(m^{k+2})$ space and time (this reduces to $O(cm^{k+1})$ for an alphabet of size $c < m$). In general, the search takes

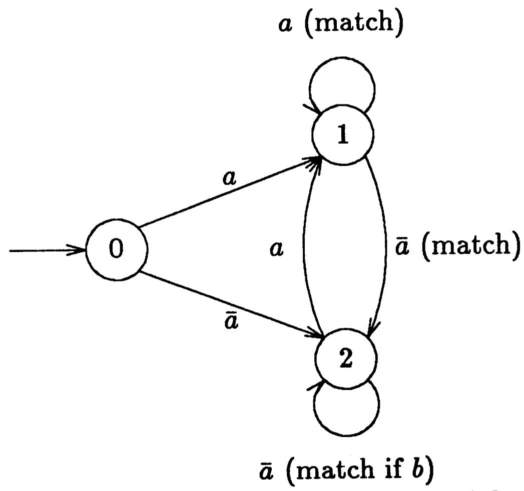

Figure 4: Automaton for the pattern ab and $k = 1$.

$m \setminus k$	1	2	3	4
2	3	2		
3	6	7	3	
4	10	16	15	4
5	14	30	43	31

Table 1: Number of states for some m and k.

$O(n \log(\min(c, m)))$ time. However, for finite alphabets, and using a complete table for each transition, we have $O(n)$ search time. It is worth to point out that the size of the automaton is in the worst case exponential in k and not in m.

The construction rules are so simple that we can build the automaton as needed during the search. That is, we will use $O(m^{k+2})$ worst case extra time and space only if all possible sequences of mismatches appear in the text. In many applications this is not the case.

Other possibility is to describe the automaton as pairs of the form (m, k), where m indicates the length of the string to match and k the maximum number of allowed errors. Clearly, at least k pairs (states) will be active at any point in the text and at most m pairs are generated by each symbol in the text. This suggests another $O(kn)$ expected time algorithm with $O(mn)$ worst case time and using $O(m)$ extra space. This approach also suggests to use a number in base m to represent the pairs, and then if the numbers are smaller than the word size it is possible to have a very efficient algorithm. This idea is pursued in [2] for standard string matching and this problem for more general patterns.

5 Mismatches with different costs

All the algorithms presented can be extended when we have different costs for different classes of mismatches. For example, the cost of a mismatch between a vowel and a consonant is twice the cost of a vowel-vowel or consonant-consonant mismatch. In this section we discuss the necessary changes for this case. We will assume that there is a finite set of integer costs $\{cost_1, ..., cost_\ell\}$ and we want a match with at most cost C.

For the naive algorithm the solution is trivial. We count the total cost of the mismatches using a multiple if structure for each case. Using more space, we can replace the multiple if by a table indexed by the character in the pattern and the character in the text. This requires $O(|c|^2)$ extra space, where c is the alphabet size.

For the Boyer-Moore approach additionaly to the same changes of the naive algorithm, we have to modify the table s_j. Now, instead of slide two copies of the pattern until we find less that $2k+1$ mismatches, we have to slide until we have a mismatch cost less or equal than

$$C + \lfloor \frac{C}{\min(cost_i)} \rfloor \max(cost_i) ,$$

where $\lfloor \frac{C}{\min(cost_i)} \rfloor$ is the maximum number of mismatches in a partial match with at most cost C, and each one of them can override in the worst case a mismatch of maximal cost.

In the case of the DFA, we keep track of the total cost, associating the corresponding mismatch cost to each transition, and we output a match if the total cost is in the appropiate range in the appropiate transitions.

6 Final Remarks

Table 2 gives a summary of the different complexities.

Given the simplicity of the Boyer-Moore approach (actual code for this algorithm and the naive algorithm are presented in the appendix), this method is a good choice against Galil and Giancarlo algorithm [5].

For small alphabets the finite automaton is the best choice, mainly because the time will be independent of k.

Finally, all these algorithms can be extended to more general patterns (for example don't care symbols, a symbol that represents a class of symbols, etc.). In this case, only the concept of what constitutes a mismatch must be changed [2].

Algorithm	Search time		Preprocessing time	Extra space
	Worst case	Average		
LV [6]	kn	kn	$km \log m$	$k(n+m)$
GG [5]	kn	kn	$m \log m$	m
Naive	mn	kn	1	1
Boyer-Moore (diff. symbols)	mn kn	kn kn	$m(m-2k)$ mk	m 1
DFA (small alphabet)	$n \log m$ n	$n \log m$ n	m^{k+2} m^{k+1}	m^{k+2} m^{k+1}

Table 2: Summary of the time and space complexities (order notation).

References

[1] Baeza-Yates, R. "The Average Case Analysis of String Matching Algorithms", Research Report CS-87-66, Dept. of Computer Science, University of Waterloo, 1987.

[2] Baeza-Yates, R. and Gonnet, G.H. "New Algorithm for Pattern Matching with and without Mismatches", Dept. of Computer Science, University of Waterloo, 1988.

[3] Boyer, R. and Moore, S. "A Fast String Searching Algorithm", *Communications of the ACM*, 20 (1977), 762-772.

[4] Knuth, D., Morris, J. and Pratt, V. "Fast Pattern Matching in Strings", *SIAM Journal of Computing*, 6 (1977), 323-350.

[5] Galil, Z. and Giancarlo, R. "Improved String Matching with k Mismatches", *SIGACT NEWS* 17 (1986), 52-54.

[6] Landau, G. and Vishkin, U. "Efficient String Matching with k Mismatches", *Theor. Computer Science* 43 (1986), 239-249.

Appendix

The code in the C programming language for the Naive algorithm is:

```
Search( k, pattern, m, text, n )    /* n >= m, k < m */
int k, m, n;
char pattern[], text[];
{
```

```
    int i, j, count;

    for( i=0; i < n-m+1; i++ )
    {
        count = 0;
        for( j=0; j<m && count <= k; j++ )
            if( pattern[j] != text[i+j] ) count++;
        if( j == m )
            Report_match_at_position( i, count );
    }
}
```

A simple version for the Boyer-Moore approach is:

```
#define max(a,b) (a>b)? (a):(b)
#define MAXPAT 255

BMmist( k, pattern, m, text, n )   /* n >= m, k < m */
int k, m, n;
char pattern[], text[];
{
    int i, j, l, count, shift[MAXPAT+1];
    int matches;

    /* Preprocessing */
    for( i=m+1; i>m-2*k; i-- ) shift[i] = 1;
    for( l=1; i>0; i-- )
    {
        for( count=2*k+1; count > 2*k; l++ )
        {
            j = max(1, i-l);
            for( count=0; j<=m-l && count <= 2*k; j++ )
                if( pattern[j] != pattern[j+l] ) count++;
        }
        shift[i] = --l;
    }
    l = n-m+1;
    /* Code to avoid having special cases */
    text[0] = CHARACTER_NOT_IN_THE_PATTERN;
    pattern[0] = CHARACTER_NOT_IN_THE_TEXT;
    /* Search */
    matches = 0;
```

```
    for( i=0; i < l; i += shift[j+2] )
    {
        for( count=0, j=m; j>0 && count <= k; j-- )
            if( pattern[j] != text[i+j] ) count++;
        if( count <= k )
            Report_match_at_position( i+1, count );
    }
}
```