—— SOLUTIONS ——

King's College London

UNIVERSITY OF LONDON

Enter your candidate number in the box provided above and on the answer book(s) provided. Do this now.

BSc EXAMINATION

CS3TSP – TEXT SEARCHING AND PROCESSING

MAY 2009

TIME ALLOWED: TWO HOURS.

ANSWER THREE OF THE FIVE QUESTIONS.

NO CREDIT WILL BE GIVEN FOR ATTEMPTING ANY FURTHER QUESTIONS.

ALL QUESTIONS CARRY EQUAL MARKS.

THE USE OF ELECTRONIC CALCULATORS IS **NOT** PERMITTED.

BOOKS, NOTES OR OTHER WRITTEN MATERIAL MAY **NOT** BE BROUGHT INTO THIS EXAMINATION.

NOT TO BE REMOVED FROM THE EXAMINATION HALL

TURN OVER WHEN INSTRUCTED

1. Borders and overlaps

Given a word $x = x[0\,..\,m-1]$, its *Border* table is defined by: $Border[0] = -1$, and $Border[j]$ is the maximal length of (proper) borders of $x[0\,..\,j-1]$, for $0 < j \leq m$.

a. Give the *Border* table associated with the word aaabaaababa.

[10 marks]

<u>Answer</u>

i	0	1	2	3	4	5	6	7	8	9	10	11
$x[i]$	a	a	a	b	a	a	a	b	a	b	a	
$Border[i]$	-1	0	1	2	0	1	2	3	4	5	0	1

b. Design, and describe using pseudo-code, in pseudo-code an algorithm that computes the *Border* table of a word x of length m in time $O(m)$.

[15 marks]

<u>Answer</u>

```
COMPUTE_BORDERS(string x; integer m)
1  Border[0] ← −1
2  for i ← 0 to m − 1
3  do j ← Border[i]
4      while j ≥ 0 and x[i] ≠ x[j]
5      do j ← Border[j]
6      Border[i + 1] ← j + 1
7  return Border
```

c. Give a tight upper bound on the number of symbol comparisons executed during a run of the algorithm of Question 1.b, and prove the bound.

[10 marks]

<u>Answer</u>

The number of symbol comparisons is bounded by $2m$. [5 marks]

The running time is proportional to the number of symbol comparisons done at Line 4. Each positive comparison leads to an increment of variable i which values are in increasing order. Each negative comparison leads to an increment of expression $i - j$ which values are in increasing order. So, there are at most $2m$ comparisons, which proves the result. [5 marks]

d. The overlap between x and y, $ov(x, y)$, is the longest word that is both a prefix of x and a suffix of y. How would you find $ov(x, y)$ using the table $Border$ associated with the string $x\mathtt{c}y$? How would you do it using the table $Border$ associated with the string x?

[15 marks]

<u>Answer</u>

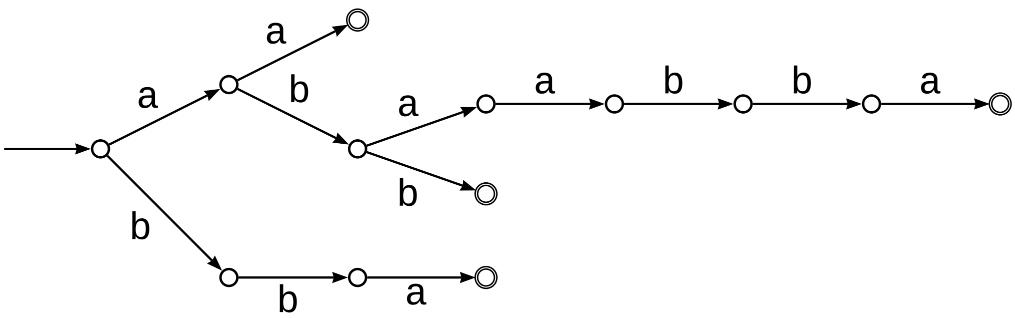
Let $k = Border[|x| + |y| + 1]$, then $ov(x, y) = x[0 \mathinner{..} k - 1]$. [7 marks]
With the table $Border$ associated with the string x, apply MP algorithm until $j = |y|$; then $ov(x, y) = x[0 \mathinner{..} i - 1]$. [8 marks]

2. Dictionary-Matching Automaton
Let Σ be the alphabet $\{a, b, c\}$ and X be a finite set of strings of Σ^*. The dictionary-matching automaton of X over Σ is denoted by $\mathcal{D}(X)$.

a. Draw the trie of the set $\{aa, abaabba, abb, bba\}$. Mark its terminal states.

[10 marks]

Answer



[unseen]

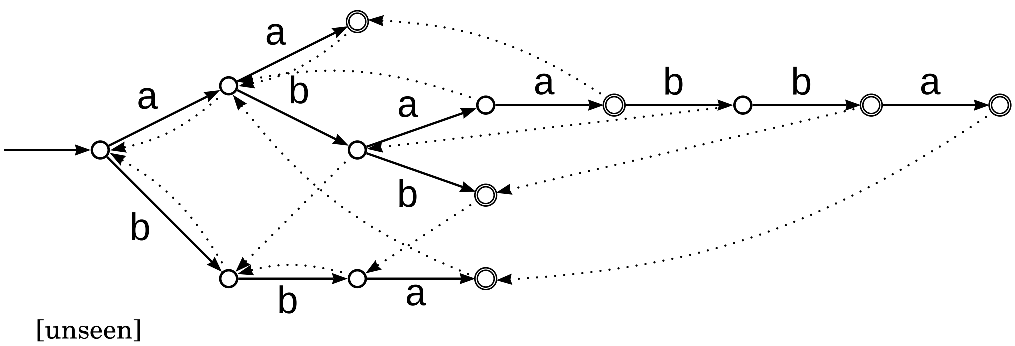
b. Define the notion of a failure link on a state (node) of the trie of X. Draw the implementation with failure links of the dictionary matching automaton $\mathcal{D}(\{aa, abaabba, abb, bba\})$. Mark its terminal states.

[15 marks]

Answer

Let p be a state of the trie of X distinct from the root. Let $u \in \Sigma^+$, be the label of the path from the root to state p. Then the failure state $f(p)$ of state p is the state of the trie whose path from the root is labelled by the longest possible proper suffix of u.

[bookwork]



[unseen]

c. Describe in pseudo-code the next-state function for the implementation with failure links of $\mathcal{D}(X)$.

[15 marks]

Answer

```
NextState(p, a)
    if there is an edge (p, a, q) in the trie
        return q
    else if f(p) is defined
        return NextState(f(p), a)
    else return the root of the trie
```

[bookwork]

d. What data structure would you use to implement a state of the dictionary-matching automaton?

[10 marks]

Answer

For each node in the automaton, one can use a structure comprising two pointers, one for the failure link and one for the list of next nodes defined by the transition function, a boolean field to mark terminal states, and possibly a field for storing some data associated with the state (for example, the letter labelling the incoming edge).

[unseen]

3. BM-Horspool

Let x be a string of length m, $x = x[0 \mathinner{.\,.} m-1]$.

a. The *DA* table of a string implements the bad-character rule for the BM algorithm. How do you define the *DA* table for the string x? What is the length of the shift inferred from *DA* when the rule applies after comparing the text symbol $y[j]$ and the pattern symbol $x[i]$?

[10 marks]

Answer

$DA[\sigma] = \min\{|z| > 0 \mid \sigma z \text{ suffix of } x\} \cup \{|x|\}$,
$shift = DA[y[j]] - m + i + 1$.
[bookwork]

b. On the alphabet $\{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$, give the *DA* table associated with the word $x = \mathsf{acbabaaba}$

[5 marks]

Answer

a	a	b	c	d
$DA[a]$	2	1	7	9

[unseen]

c. Describe in pseudo-code the computation of the *DA* table for the word x and the alphabet A.

[15 marks]

Answer

```
COMPUTE_DA(string x; integer m)
    for all a in A do
        DA[a] = m
    for i ⟵ 0 to m − 2 do
        DA[x[i]] = m − i − 1
    return DA
```

[bookwork]

d. Describe in pseudo-code a string-matching algorithm, searching for x in y, using the DA table of x.

[20 marks]

<u>Answer</u>

BMH(string x, y; integer m, n);
    $pos \longleftarrow 0$
    while $pos \leq n - m$ **do**
        $i \longleftarrow m - 1$
        while $i \geq 0$ **and** $x[i] = y[pos + i]$ **do**
            $i \longleftarrow i - 1$
        if $i = -1$ **then**
            output('x occurs in y at position ', pos)
        $pos \longleftarrow pos + \max\{1, DA[y[pos + i]] - m + i + 1\}$

[unseen]

4. Suffix Array and Suffix Tree

a. Define the data structure called the Suffix Array of a string y of length n.

[10 marks]

Answer
It is composed of two tables p and LCP such that
$y[\mathrm{p}[0]\mathinner{..}n-1] < y[\mathrm{p}[1]\mathinner{..}n-1] < \ldots < y[\mathrm{p}[n-1]\mathinner{..}n-1]$
and $\mathrm{LCP}[i] = |\mathbf{lcp}(y[\mathrm{p}[i-1]\mathinner{..}n-1], y[\mathrm{p}[i]\mathinner{..}n-1])|$.
[bookwork]

b. Give the Suffix Array of the string $y =$ abaabbabb.

[15 marks]

Answer

i	0	1	2	3	4	5	6	7	8
$y[i]$	a	b	a	a	b	b	a	b	b
$\mathrm{p}[i]$	2	0	6	3	8	1	5	7	4
$\mathrm{LCP}[i]$	0	1	2	3	0	1	2	1	2

[unseen]

c. How do you compute the maximal length of prefixes common to the suffixes of y at positions i and j ($i < j$) using its Suffix Array? What is the running time of your solution?

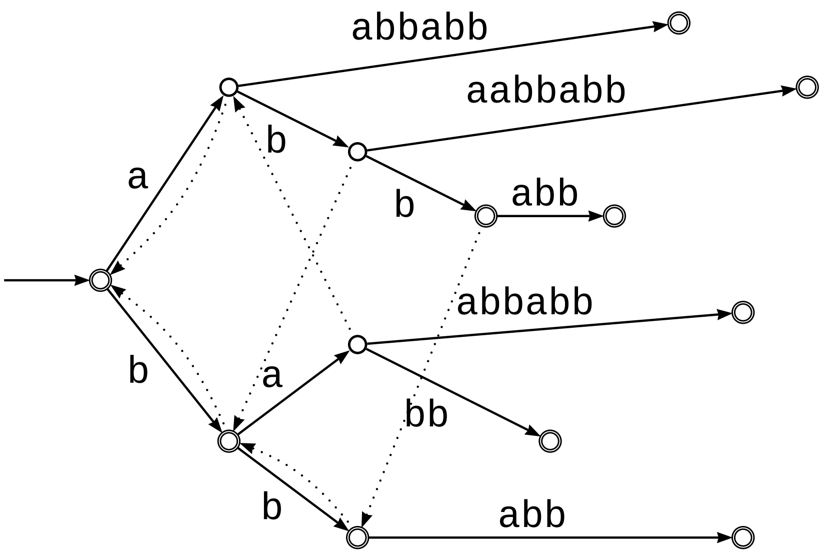
[10 marks]

Answer
The value is $\min\{\mathrm{LCP}[k] \mid i < k \leq j\}$. It can be computed by traversing the sub-array $\mathrm{LCP}[i+1\mathinner{..}j]$ in time $O(j-i)$. If the pair (i,j) is a pair of the binary search tree the running time is $O(\log(j-i))$.
[unseen]

d. Design the Suffix Tree of the string y = abaabbabb with the suffix links and with the final states.

[15 marks]

Answer



[unseen]

5. Word transformation
Let $x = x[0\,.\,.\,m-1]$ be a word of length m. For an integer i, $0 \le i < m$, the i-rotation of x is the word $x[i\,.\,.\,m-1]x[0\,.\,.\,i-1]$. We assume in this question that the m rotations of x are pairwise distinct and that x is the smallest of them according to the lexicographic order.

The BW matrix of x, denoted by $BW(x)$, is the $m \times m$ matrix whose lines are the rotations of x in lexicographic order.

The BW transform of x, denoted by $L(x)$, is the last column of the BW matrix. (It is a word of length m.)

a. Give the BW matrix of the word $x = \texttt{aabbab}$. Give $L(\texttt{aabbab})$.

[10 marks]

Answer

$$BW(\texttt{aabbab}) = \begin{pmatrix} \texttt{a} & \texttt{a} & \texttt{b} & \texttt{b} & \texttt{a} & \texttt{b} \\ \texttt{a} & \texttt{b} & \texttt{a} & \texttt{a} & \texttt{b} & \texttt{b} \\ \texttt{a} & \texttt{b} & \texttt{b} & \texttt{a} & \texttt{b} & \texttt{a} \\ \texttt{b} & \texttt{a} & \texttt{a} & \texttt{b} & \texttt{b} & \texttt{a} \\ \texttt{b} & \texttt{a} & \texttt{b} & \texttt{a} & \texttt{a} & \texttt{b} \\ \texttt{b} & \texttt{b} & \texttt{a} & \texttt{b} & \texttt{a} & \texttt{a} \end{pmatrix}$$

$L(\texttt{aabbab}) = \texttt{bbaaba}$

b. How would you compute the BW matrix of x considering the Suffix Array of the string xx?
What would be the running time of the algorithm both if the alphabet is bounded and if it is unbounded?

[15 marks]

Answer

Rotations of x are segments of length m of the word $x' = x[0]x[1]\cdots x[m-1]x[0]x[1]\cdots x[m-1]$. Sorting the suffixes of this word gives the answer. [5 marks]
On an unbounded alphabet, suffixes can be sorted either by using the suffix tree or the suffix automaton of x', which is done in $O(m \times \log a)$ time, where a is the size of the alphabet of x. [5 marks]
On a bounded alphabet, suffixes can be sorted with the suffix array of x' which requires $O(m)$ time. [5 marks]

c. Let a be a letter and u, v be two different strings of the same length. Prove that $au < av$ if and only $ua < va$.

[10 marks]

Answer

Let w be the longest common prefix of u and v. Since $u \neq v$, w is a proper prefix of u and of v. Then, $u = wbu'$ and $v = wcv'$ for some letters b, c and some words u', v'. The condition $au < av$ is equivalent to $b < c$, because aw is the longest prefix of au and av, which is equivalent to $u < v$ and to $ua < va$, because none of u and v is a prefix of the other.

d. Let $F(x)$ be the first column of $BW(x)$. Let a be a letter occurring at two positions i and j on x: $a = x[i] = x[j]$. Show that the two occurrences a appear in the same relative order in $F(x)$ and in $L(x)$. [Hint: use Question 5.c.]

[15 marks]

Answer

The occurrences of a in $F(x)$ are associated with two rotations $au = x[i]u$ and $av = x[j]v$ of x. If $x[i]u < x[j]v$, by Question 5.c, we have $ux[i] < vx[j]$. If $x[i]u > x[j]v$, by Question 5.c, we have $ux[i] > vx[j]$. Therefore occurrences of a appear in the same relative order in $F(x)$ and in $L(x)$.