SOLUTIONS - DRAFT VERSION

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.

MSc EXAMINATION

CSMTSP – TEXT SEARCHING AND PROCESSING

MAY 2005

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

SOLUTIONS - DRAFT VERSION

1. Borders and overlaps

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

a. Give the table *Border* associated with the word `abaabcabbaabaa`.

[10 marks]

Answer

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$x[i]$	a	b	a	a	b	c	a	b	b	a	a	b	a	a	
$Border[i]$	-1	0	0	1	1	2	0	1	2	0	1	1	2	3	4

b. Let x and y be two strings on the alphabet $\{\mathtt{a}, \mathtt{b}\}$. Show how to find the positions of the occurrences of x in y using the table *Border* associated with the string $x\mathtt{c}y$ (`c` is a letter different from `a` and `b`).

[10 marks]

Answer

Note that $Border[i] \leq |x|$ because the letter `c` does not appear in y. If a positions i on y is such that $Border[i] = |x|$ then x is a suffix of $x\mathtt{c}y[0 \mathinner{..} i]$ and then of $y[0 \mathinner{..} i]$. So, x occurs in y at position $i - |x| + 1$. [5 marks]
Conversely, if x occurs in y at position j, then x is a border of $x\mathtt{c}y[0 \mathinner{..} i]$ for $i = j + |x| - 1$ because the border cannot be longer than x. Thus $Border[i] = |x|$. [5 marks]
Conclusion: condition $Border[i] = |x|$ can be used to signal occurrences of x in y.

c. 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]

Answer

Let $k = Border[|x| + |y| + 2]$, then $ov(x, y) = x[0 \mathinner{..} k-1]$. [5 marks]
With the table *Border* associated with the string x, apply MP algorithm until $j = |y| + 1$; then $ov(x, y) = x[0 \mathinner{..} i-1]$. [10 marks]

d. Design in pseudo-code an algorithm that computes the table $Border$ of the word x. State and prove its running time.

[15 marks]

Answer

```
COMPUTE_BORDERS(string $x$; integer $m$)
1  $Border[0] \leftarrow -1$
2  **for** $i \leftarrow 0$ **to** $m - 1$
3  **do** $j \leftarrow Border[i]$
4      **while** $j \geq 0$ **and** $x[i] \neq x[j]$
5      **do** $j \leftarrow Border[j]$
6      $Border[i+1] \leftarrow j + 1$
7  **return** $Border$
```

[8 marks]

The algorithm runs in time $O(m)$. [3 marks]

Indeed, 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. [4 marks]

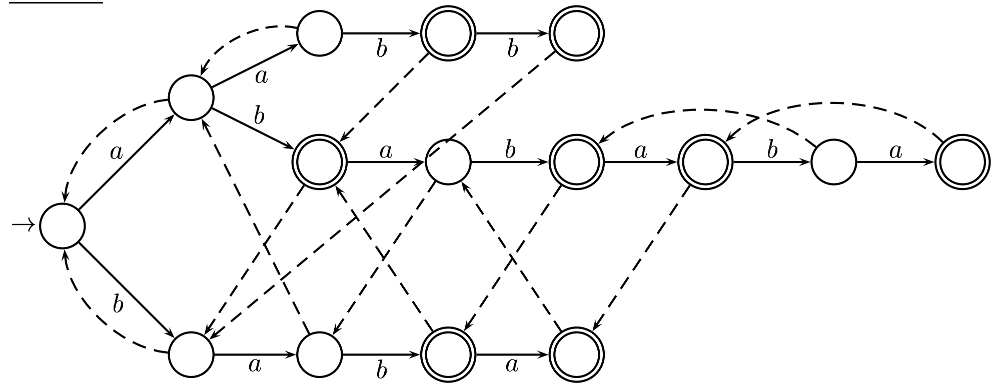
SOLUTIONS - DRAFT VERSION

2. Dictionary-matching automaton

a. Design the dictionary-matching automaton (DMA) over the alphabet $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}\}$ for the dictionary of strings: $\{\texttt{abababa}, \texttt{aabb}, \texttt{baba}, \texttt{ab}\}$.

[15 marks]

Answer



b. Define what is the failure link of a state of the DMA automaton.

Describe in pseudo-code the search procedure for finding in a text all the occurrences of patterns of the dictionary using its DMA automaton.

[15 marks]

Answer

The failure link is defined as follows: Suppose that in the DMA automaton, state q_i represents the string s_i and state q_j represents the string s_j. Then the failure link of the state q_i points to the state q_j, $f(q_i) = q_j$, if and only if s_j is the longest proper suffix of s_i that is also a prefix of some keyword $x \in X$. [5 marks]

The search procedure using the DMA automaton is based by the following procedure for moving from one state to another according to the current symbol being read from the text. Let $p \in Q$ be a state of the automaton, f be the failure link, and $\alpha \in \Sigma$ be a symbol of the alphabet.

```
$Next\_State(p, \alpha)$
    **if** $(\alpha, q)$ in the list $p \to list$ **then**
        **return** $q$;
    **else if** $p \to f$ non NULL **then**
        **return** $Next\_State(p \to f, \alpha)$
    **else return** initial state;
```

[10 marks]

c. Describe in detail how the failure link of a state is computed during the construction of the DMA automaton.

[20 marks]

Answer

Assume that we want to compute the failure link of node q. Also assume that parent of q is the node p and that the arc connecting the nodes p and q is labelled by the character τ. Then the failure link $f(q)$ of the node q is given by

$$f(q) = \textsc{NextState}(f(p), \tau)$$

if $f(p)$ is defined, otherwise $f(q) = q_0$, where q_0 is the initial state ($\textsc{NextState}(p, \delta)$ is the function described in 3.b). The initial state has no failure link.

Also, note that if $f(q)$ is a terminal state, q has to be set to terminal, too.

SOLUTIONS - DRAFT VERSION

3. From Suffix Array to 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

$$y[p[0]\,..\,n-1] < y[p[1]\,..\,n-1] < \ldots < y[p[n-1]\,..\,n-1]$$

and

$$LCP[i] = |lcp(y[p[i-1]\,..\,n-1], y[p[i]\,..\,n-1])|.$$

b. Describe in your own words how to compute the LCP (longest common prefix) table associated with the suffix array of y using its suffix tree.

[15 marks]

Answer

The depth of the deepest internal node between the consecutive suffixes is their LCP. Its computation is done during a traversal of the tree in which letters are processed in lexicographic order.

c. Describe in detail how to build the suffix tree of y using its suffix array.

[15 marks]

Answer

It uses the relation between LCP's and depth of internal nodes. Let p be the current node of depth k and let s be the suffix to be inserted in the tree.

There are three cases: if the LCP between s and its previous suffix in the suffix array is equal to k, add a new node and an arc from p to this node labeled by $s[k\,..\,n-1]$. [5 marks]
If the LCP is greater than k, add a new node q in the middle of the last arc from p, and label the two resulting arcs with $s[k\,..\,\text{LCP}-k-1]$ and $s[\text{LCP}-k\,..\,n-1]$ respectively. [5 marks]
If the LCP is smaller than k, find an lowest ancestor of p satisfying one of the above conditions. [5 marks]

d. What is the running time of your solution of Question 3.c? Does it depend on the size of the alphabet?

[10 marks]

Answer

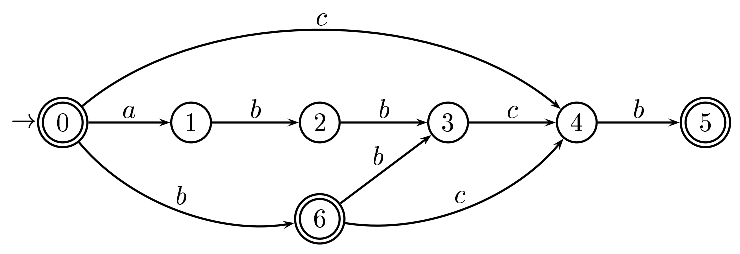
Running time is $O(|y|)$ [6 marks] independently of the alphabet size. [4 marks]

4. Suffix automaton

a. Design SA(`abbcb`), the suffix automaton of the string `abbcb`.

[10 marks]

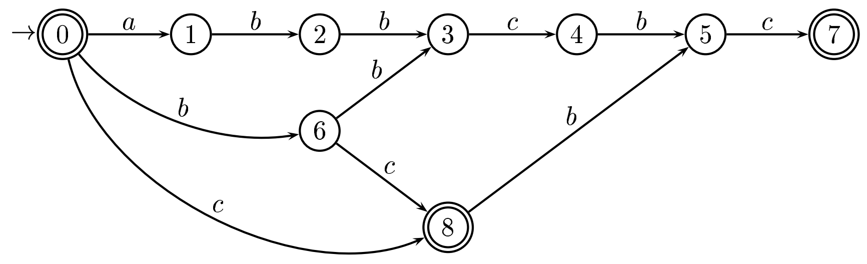
Answer



b. Indicate how the automaton of question 4.a is modified to get SA(`abbcbc`).

[10 marks]

Answer



c. Let p be a state of $SA(y)$, for a string y. Let $SA_p(y)$ be the automaton obtained from $SA(y)$ by considering p as the only initial state. How do you characterize the words accepted by the automaton $SA_p(y)$?

[10 marks]

Answer

Words accepted by $SA_p(y)$ are suffixes of y that start with any of the words labeling paths from the initial state to p.

d. Let p be a state of $SA(y)$ and let $SA_p(y)$ be as in question 4.c. Consider that all states of $SA(y)$ (and then $SA_p(y)$) are terminal. Let $X(p)$ be the number of words accepted by $SA_p(y)$. Give a recurrence relation to compute $X(p)$ from the $X(q)$'s where q's are targets of transitions from p.

[10 marks]

Answer

$$X[p] = \begin{cases} 1 & \text{if } deg(p) = 0, \\ 1 + \sum_{(p,v,q)\in F}(|v| - 1 + X[q]) & \text{otherwise,} \end{cases}$$

where F is the set of arcs of the automaton.

e. What is the complexity of an algorithm that computes the number of strings accepted by $SA(y)$ using the recurrence of question 4.d? Explain why.

[10 marks]

Answer

It is $O(|y|)$. [3 marks]

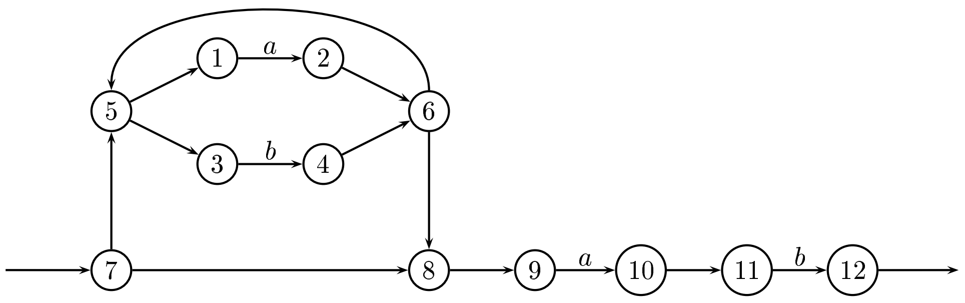
The computation is done during a traversal of the automaton starting in state p. Since no transition is executed, if the implementation is by lists of successors, the running time is $O(|y|)$. [7 marks]

5. Regular matching

a. Design the non-deterministic automaton $\mathcal{A}$ associated with the regular expression $(\texttt{a} + \texttt{b})^*\texttt{ab}$ and obtained by Thompson's construction.

[10 marks]

Answer



b. Describe data structures to efficiently implement the automata obtained by Thompson's construction.

[10 marks]

Answer

Use an array T indexed by states $T[p] = (a, q)$ or (ϵ, q, r), where a is a symbol, q and r are targets of arcs from p.

c. Simulate the regular pattern matching algorithm using the automaton $\mathcal{A}$ of question 5.a on the string `aabbab`.

[10 marks]

Answer

Initial state: $\{1, 3, 5, 7, 8, 9\}$

State after reading `a`: $\{1, 2, 3, 5, 6, 8, 9, 10, 11\}$

State after reading `a`: $\{1, 2, 3, 5, 6, 8, 9, 10, 11\}$

State after reading `b`: $\{1, 3, 4, 5, 6, 8, 9, 12\}$

State after reading `b`: $\{1, 3, 4, 5, 6, 8, 9, 12\}$

State after reading `a`: $\{1, 2, 3, 5, 6, 8, 9, 10, 11\}$

State after reading `b`: $\{1, 3, 4, 5, 6, 8, 9, 12\}$

Since the terminal state 12 is reached, the word is accepted by the automaton, which means that it belongs to the language described by the regular expression.

d. Design the automaton obtained by transforming using the subset construction the automaton $\mathcal{A}$ of question 5.a into a deterministic automaton.

[10 marks]

Answer

$$\begin{tikzcd}
 & & 2 \arrow[loop above, "a"] \arrow[rd, bend right=20, "b"] & & \\
{} \arrow[r] & 1 \arrow[ru, "a"] \arrow[r, "b"] & 3 \arrow[u, "a"] \arrow[loop below, "b"] & 4 \arrow[l, "b"] \arrow[lu, bend right=30, "a"'] \arrow[r] & {}
\end{tikzcd}$$

e. Discuss the complexity of matching regular patterns with Thompson's automata or their deterministic versions.

[10 marks]

Answer

Searching for a regular expression of size r in a text of length n.
Thompson's: time $O(rn)$, space $O(r)$
Deterministic automaton: time $O(n)$, space $O(2^r)$.