—— SOLUTIONS ——

King's College London

This paper is part of an examination of the College counting towards the award of a degree. Examinations are governed by the College Regulations under the authority of the Academic Board.

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

MSc / MSci EXAMINATION

7CCSMTSP/CSMTSP – TEXT SEARCHING AND PROCESSING

MAY 2011

TIME ALLOWED: TWO HOURS.

ANSWER **TWO** OF THE **THREE** 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. String Matching

a. Assume we are given a single pattern. Name an algorithm that should be used to efficiently search for it in many texts? What are the possible preprocessing time and running time for a pattern of length m and k texts each of length n?

Assume we are given a single text. Name an algorithm should be used to search it for many patterns? What are the possible preprocessing time and running time for a text of length n and k patterns each of length m?

[10 marks]

Answer

In the first case the pattern can be preprocessed once for searching all texts. The preprocessing time can be done in $O(m)$ time and all searches in $O(kn)$ time. [5 marks]

In the second case the text is to be preprocessed (indexed). The preprocessing time can be done in $O(n)$ or $O(n \log a)$ time (a is the alphabet size) and all searches in $O(km)$ or $O(km \log a)$ time depending on the representation of the index. [5 marks]

[unseen]

b. Let x be the string abaababa. Give its Border table B ($B[i]$ is the maximal length of borders of $x[0\,..\,i-1]$), its Period table P ($P[i]$ is the smallest period of $x[0\,..\,i-1]$), its *MP_next* table, and its *KMP_next* table.

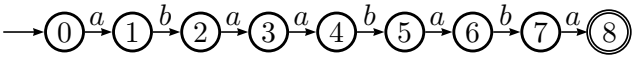
[10 marks]

Answer

	0	1	2	3	4	5	6	7	8
	a	b	a	a	b	a	b	a	
B	-1	0	0	1	1	2	3	2	3
P	–	1	2	2	3	3	3	5	5
MP_next	-1	0	0	1	1	2	3	2	3
KMP_next	-1	0	-1	1	0	-1	3	-1	3

[3 marks for each row 1,2,4; 1 mark for row 3] [unseen]

c. Give the failure table of the trie of abaababa:



Give the optimised failure function of the same trie.

[10 marks]

Answer

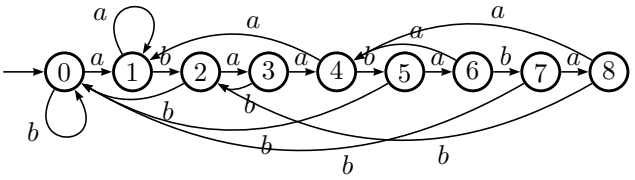
	0	1	2	3	4	5	6	7	8
	a	b	a	a	b	a	b	a	
F	–	0	0	1	1	2	3	2	3
Opt_F	–	0	–	1	0	–	3	–	3

[unseen]

d. Design the String Matching Automaton of the string x of sub-question 1.b, SMA(abaababa), on the alphabet $A = \{\mathsf{a}, \mathsf{b}\}$.

[10 marks]

Answer



[unseen]

e. For a string x of length m, its table suf is defined as follows:
for $0 \leq i < m$, $\mathrm{suf}[i]$ is the maximal length of any suffix of x that has an occurrence ending at position i on the string x. Describe in your own words the main elements for the computation of the table suf in linear time.

[10 marks]

Answer

The computation is done be scanning x from end to beginning (right to left).
Let k be the smallest position on x for which $x[k+1\,..\,j]$ is the longest suffix of x ending at a position j greater than the position i being processed.
If $\mathrm{suf}[i + (m-1-j)] < k - i$, then $\mathrm{suf}[i] = \mathrm{suf}[i + (m-1-j)]$.
If $\mathrm{suf}[i + (m-1-j)] > k - i$, then $\mathrm{suf}[i] = k - i$.
Otherwise scanning letters from position k down the positions find the value of $\mathrm{suf}[i]$. [in lectures]

SEE NEXT PAGE

2. Doubling

Let y be a fixed text of length n.

For a word u and a positive integer k, $First_k(u)$ is u if $|u| \leq k$ and is $u[0\,..\,k-1]$ otherwise. The integer $R_k[i]$ is the rank of $First_k(y[i\,..\,n-1])$ inside the sorted list of all $First_k(u)$ where u is a nonempty suffix of y (ranks are numbered from 0).

a. Give R_1, R_2, R_3, R_4, R_8 for the word abaababaab, assuming a $<$ b.

[10 marks]

Answer

i	0	1	2	3	4	5	6	7	8	9
$y[i]$	a	b	a	a	b	a	b	a	a	b
$R_1[i]$	0	1	0	0	1	0	1	0	0	1
$R_2[i]$	1	3	0	1	3	1	3	0	1	2
$R_3[i]$	2	4	0	2	5	2	4	0	1	3
$R_4[i]$	3	6	1	4	7	3	6	0	2	5
$R_8[i]$	4	8	1	5	9	3	7	0	2	6

[2 marks for each row] [unseen]

b. State the doubling lemma and prove it.

[10 marks]

Answer

Lemma 1 *$Rank_{2k}[i]$ is the rank of the pair $(Rank_k[i], Rank_k[i+k])$ in the sorted list of these pairs.*

[5 marks] [bookwork]

Proof. Let i be a position on y and let $u = First_{2k}(y[i\,..\,n-1])$. Let j be a position on y and let $v = First_{2k}(y[i\,..\,n-1])$. We show that $u \leq v$, which is equivalent to $Rank_{2k}[i] \leq Rank_{2k}[j]$, iff $(Rank_k[i], Rank_k[i+k]) \leq (Rank_k[j], Rank_k[j+k])$.

First case: $First_k(u) < First_k(v)$. This is equivalent to $Rank_k[i] < Rank_k[j]$ so the result holds in this case.

Second case: $First_k(u) = First_k(v)$. This is equivalent to $Rank_k[i] = Rank_k[j]$. Then the comparison between u and v depends only on the second halves of these words; in other terms, $Rank_{2k}[i] \leq Rank_{2k}[j]$ is equivalent to $Rank_k[i+k] \leq Rank_k[j+k]$. [5 marks] [unseen]

c. Describe an efficient algorithm to compute R_{2k} from R_k. What is its running time?

Apply it to compute R_8 from R_4 on the string of Question 2.a.

[10 marks]

Answer

Two steps: first sort positions i according to the pairs $(R_k[i], R_k[i+k])$; then assign the same R_{2k} rank to positions associated with the same pair.

First step can be implemented by bucket sort (count sort) in linear time; second step is obvious and runs also in linear time. [5 marks] [bookwork]

List of positions with their pairs:
$0(3,7), 1(6,3), 2(1,6), 3(4,0), 4(7,2), 5(3,5), 6(6,-1), 7(0,-1), 8(2,-1), 9(5,-1)$
Sorted according to pairs:
$7(0,-1), 2(1,6), 8(2,-1), 5(3,5), 0(3,7), 3(4,0), 9(5,-1), 6(6,-1), 1(6,3), 4(7,2)$
Assignment of new ranks to positions:
$7:0, 2:1, 8:2, 5:3, 0, 4, 3:5, 9:6, 6:7, 1:8, 4:9.$
[5 marks] [unseen]

d. Define the two arrays SUF and LCP composing the Suffix Array of the string y. Using the result of Question 2.c, give the running time of the induced algorithm to compute the array SUF. Justify your answer.

[10 marks]

Answer

The array SUF contains the permutation of suffix positions in increasing order of the suffixes:

$$y[\mathrm{SUF}[0]\,..\,n-1] < y[\mathrm{SUF}[1]\,..\,n-1] < \ldots < y[\mathrm{SUF}[n-1]\,..\,n-1]$$

and the LCP array is defined by:

$$\mathrm{LCP}[i] = |\mathrm{lcp}(y[\mathrm{SUF}[i-1]\,..\,n-1], y[\mathrm{SUF}[i]\,..\,n-1])|$$

where $\mathrm{lcp}(u,v)$ is the longest common prefix of u and v. [5 marks]

The runtime of the induced algorithm is $O(n \times \log n)$ because there are $\lceil \log n \rceil$ steps and each step can be implemented to run in $O(n)$ from answer to Question 2.c. [5 marks]
[bookwork]

e. What is the running time to discover if a pattern x of length m occurs in y using the array SUF only?

What running time can be achieved if the array LCP is also used.

[10 marks]

Answer

Applying a binary search on the sorted list of suffixes of y given by SUF the running time is $O(m \log n)$. [5 marks]

Using LCP in addition leads to $O(m + \log n)$ running time. [5 marks]
[in lectures]

SEE NEXT PAGE

3. Suffix structures

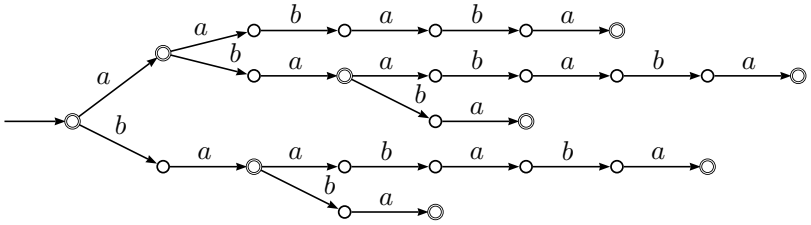
In this question we consider the string $z = \texttt{abaababa}$.

a. Design the Suffix trie of the word z.

Give an example of a word of length n on the alphabet $\{\texttt{a}, \texttt{b}\}$ having a Suffix trie of size $\Omega(n^2)$.

[10 marks]

Answer



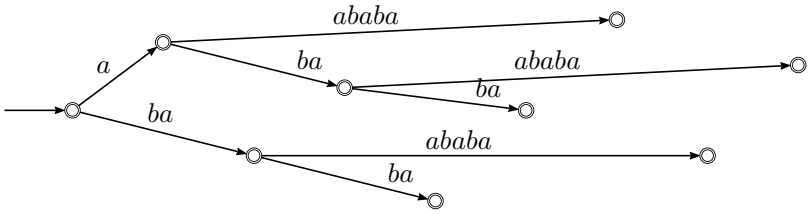
[5 marks] [unseen]

The trie of the word $a^{n/4}b^{n/4}a^{n/4}b^{n/4}$, for two distinct letters a and b, has at least $n/4$ branches each of them having $n/4$ nodes. Which gives at least $(n/4)^2 = \Omega(n^2)$ nodes. [5 marks] [in lectures]

b. Design the Suffix tree of z. What are the properties characterising the Suffix tree of a non-empty string y? Describe how to get the Suffix tree of y from its Suffix trie.

[10 marks]

Answer



[5 marks] [unseen]

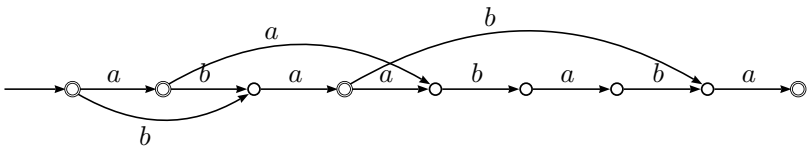
The suffix tree of y is a digital tree in which edges are labelled by non-empty strings. The label of a path from the root to a terminal node is a suffix of y and all the suffixes of y appear as such. Edges outgoing a given node have labels starting with different letters. No node has only on outgoing edge. [bookwork]

Each node having only one outgoing edge in the suffix trie should be deleted to get the suffix tree, and edges should be labelled accordingly. For edges (p, u, q) and (q, a, r) where u is a word, a a letter and q has no other outgoing edge, it is deleted with the two edges, and the new edge (p, ua, r) is created. [5 marks] [unseen]

c. What is the Suffix automaton of a non-empty string y? Design the Suffix automaton of the string z. Which algorithm produces the Suffix automaton of a string y from its Suffix trie?

[10 marks]

Answer



[5 marks] [unseen]

The Suffix automaton of y is obtained by minimising its suffix trie. [5 marks] [bookwork]

d. Describe how to discover if a pattern x of length m occurs in a string y using the Suffix automaton of y. Discuss the running time of the method with respect to the implementation of the automaton.

[10 marks]

Answer

To discovering if x occurs in y we just have to follow the path labelled by x in the Suffix automaton. If the path does not exist, x does not occur in y. Otherwise it occurs. [5 marks]

If the automaton is represented by a transition (goto) table, each branching from a state takes constant time, which leads to $O(m)$ time. If the automaton is represented by successor lists it takes $O(m \log a)$, where a is the number of letter in the alphabet, because the automaton is deterministic and then there are no more than a edges outgoing a given state; the edges can be arranged in a balanced tree to get $O(\log a)$ for branching. [5 marks]

[mostly unseen]

e. Let us assume that string x of length m occurs in string y of length n. Describe how to find all the positions of x in y using the Suffix automaton of y.

Let q be the state at the end of the path labelled by x from the initial state in the Suffix automaton of y. How would you preprocess the Suffix automaton to produce the greatest position of x in y from q in constant time?

[10 marks]

Answer

Let q be the state at the end of the path labelled by x from the initial state in the Suffix automaton of y. The get all the positions of x in y, we have to follow all the paths leading to a terminal state from q.

A path of length ℓ gives an occurrence of x at position $n - m - \ell$ in y. [5 marks]

To get the largest position we need to have the length of the shortest path from q to a terminal state. The Suffix automaton can be preprocessed in linear time accordingly. [5 marks]

[unseen]