SOLUTIONS

King's College London

UNIVERSITY OF 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.

BSc EXAMINATION

6CCS3TSP – TEXT SEARCHING AND PROCESSING

MAY 2013

TIME ALLOWED: TWO HOURS.

ANSWER FOUR OF THE SIX 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.

DO NOT REMOVE THIS EXAM PAPER FROM THE EXAMINATION ROOM

TURN OVER WHEN INSTRUCTED

1. **Karp-Rabin string searching**

a. Describe in pseudo-code the naive string searching algorithm.

[6 marks]

Answer

```
NAIVE_SEARCH(string x, y; integer m, n)
    pos ⟵ 0
    while pos ≤ n − m do
        i ⟵ 0
        while i < m and x[i] = y[pos + i] do
            i ⟵ i + 1
        if i = m then output('x occurs in y at position ', pos)
        pos ⟵ pos + 1
```

b. Describe in your own words how to accelerate the naive string searching algorithm using hashing.

[6 marks]

Answer

To accelerate the previous algorithm we can hash the pattern and the content of the window. Then, to compare the pattern with the content of the window we first check if their hash values match. If they do, a letter by letter comparison is required to check if an actual match is found.

c. Define a hash function that can be used to search for a pattern $x = a_0 a_1 \ldots a_{m-1}$ where a_i's are letters considered as integers.

[6 marks]

Answer

We use integer parameters d (like a base) and q (for the modulo). The hashing value of x by the function function h is defined by:

$$h(x) = (a_0 d^{m-1} + a_1 d^{m-2} + \cdots + a_{m-1}) \textbf{ mod } q.$$

d. Using the hash function defined in Question 1.c, show how the hash value of a string ub is computed knowing the hash value of au, where a and b are letters and u a string.

[7 marks]

Answer

Let d and q be the same parameters as in previous answer. Then,

$$h(ub) = ((h(au) - ad^{m-1})d + b) \textbf{ mod } q.$$

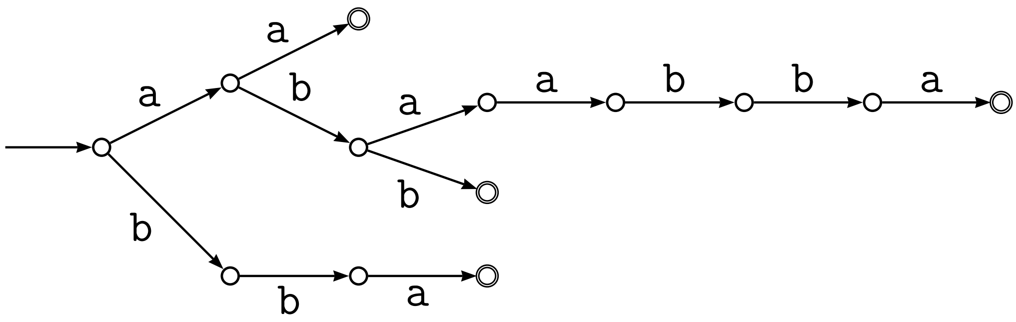
2. **Dictionary-Matching Automaton**

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

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

[6 marks]

Answer

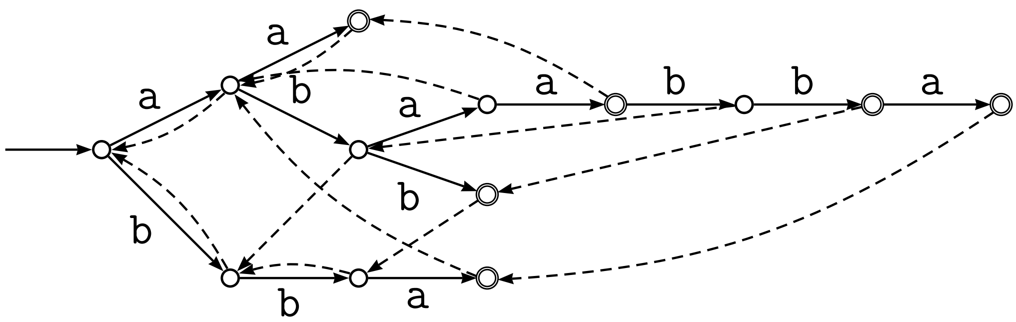


b. Define the notion of a failure link on a state (node) of the trie of X. Draw the implementation with failure links of the dictionary-matching automaton $\mathcal{D}(\{\mathtt{aa}, \mathtt{abaabba}, \mathtt{abb}, \mathtt{bba}\})$. Mark its terminal states.

[6 marks]

Answer

Let p be a state of the trie of X distinct from the root. Let $u \in A^+$, 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.



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

[6 marks]

<u>Answer</u>

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

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

[7 marks]

<u>Answer</u>

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).

3. **BM-Horspool**
Let x be a string of length m, $x = x[0\,..\,m-1]$.

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 shift length inferred from *DA* when the rule applies after comparing the text symbol $y[j]$ and the pattern symbol $x[i]$?

[6 marks]

Answer

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

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

[6 marks]

Answer

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

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

[6 marks]

Answer

COMPUTE_DA(string x; integer m)
  for all a **in** A **do**
    DA$[a] = m$
  for $i \longleftarrow 0$ **to** $m-2$ **do**
    DA$[x[i]] = m - i - 1$
  return *DA*

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

[7 marks]

Answer

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)
    $pos \longleftarrow pos + \max\{1, DA[y[pos + i]] - m + i + 1\}$

4. **Suffix sorting**

Let y be a string of length n. Let P_{01} be the positions on y of the form $3q$ or $3q+1$. Let P_2 be the positions on y of the form $3q+2$.

a. List in lexicographic order the nonempty suffixes of the string abababaa, assuming a < b.

[6 marks]

Answer

a,aa,abaa,ababaa, abababaa,baa,babaa,bababaa.

b. How do you sort a list of strings of length 3 over a fixed size alphabet? What is the running time to do it?

[6 marks]

Answer

With a stable bucket sort, we sort the elements of the list successively according to their 3rd letter, then to their 2nd letter and finally to their first letter.

The running time is proportional to the length of the list.

c. What are the four steps of the Skew algorithm to sort the suffixes of y.

[6 marks]

Answer

1. Sort the position in P_{01} according to their associated 3-grams. Let $t[i]$ be the rank of i in the sorted list.
2. Recursively sort the suffixes of $t[0]t[3]\dots t[1]t[4]\dots$. For a position i in P_{01}, let $s[i]$ be the rank of its associated suffix in the sorted list of them, L_{01}.
3. Sort the positions j in P_2. Let L_2 be the sorted list.
4. Merge lists L_{01} and L_2.

d. Let L_{01} be the list of positions of P_{01} sorted according to their associated suffixes; let $s[i]$ be the rank of i in L_{01}. Describe how to sort P_2 in time $O(|P_2|)$.

[7 marks]

Answer

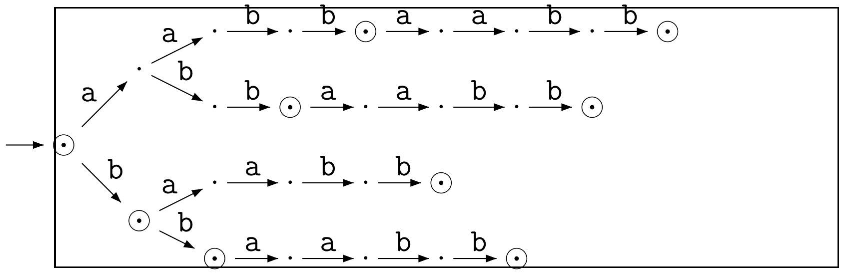
Sorting elements j of P_2 remains to sort their associated pairs $(y[j], s[j+1])$. This can be done in linear time using radix sort.

5. **Suffix trie and suffix tree**

a. Design the trie of suffixes of the word $y = \texttt{aabbaabb}$.

[6 marks]

Answer



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

[6 marks]

Answer

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 $(n/4)^2 = \Omega(n^2)$ nodes.

c. Design an algorithm to compact the trie of suffixes of a word into its suffix tree.

[6 marks]

Answer

The following procedure compacts a trie T, even if suffix links are defined on states.

Compact(trie T)
  $r \leftarrow$ root of T
  for each arc (r, a, p) do
    Compact(subtrie of T rooted at p)
    if(p has exactly one child)
      $q \leftarrow$ that child
      $u \leftarrow$ label of (p, q)
      replace p by q as child of r
      set $a \cdot u$ as label of (r, q)

d. Describe possible data structures required to implement the suffix tree of a word y.

[7 marks]

Answer

Each node or state p of the tree can be implemented as a structure containing two pointers: the first pointer to implement the suffix link; the second pointer to give access to the list of arcs outgoing state p. The list of arcs can contain 4-tuples in the form (a, i, ℓ, q) where a is a letter, i and ℓ are integers, and q is a pointer to a state. They are such that (p, u, q) is an arc of the automaton with $a = y[i]$ and $u = y[i \mathinner{.\,.} i + \ell - 1]$.

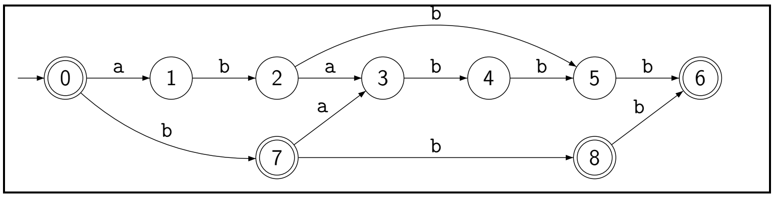
6. Suffix automaton

Let y be a string of length n. Let $SA(y)$ be the smallest deterministic automaton accepting the suffixes of y.

a. Design $SA(\texttt{ababbb})$.

[6 marks]

Answer



b. What are the suffix (or failure) links on states of the automaton of Question 6.a?

[6 marks]

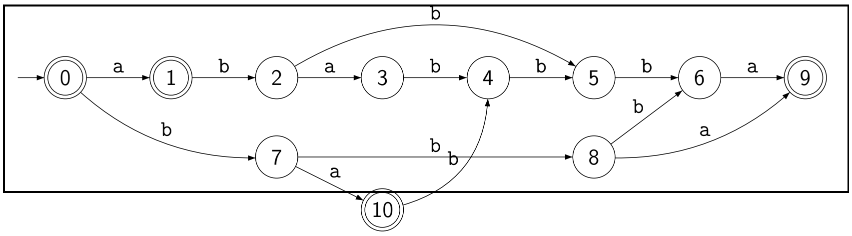
Answer

$f(1) = 0$, $f(2) = 7$, $f(3) = 1$, $f(4) = 2$, $f(5) = 7$, $f(6) = 8$, $f(7) = 0$, $f(8) = 7$,

c. Indicate how to modify the automaton of question 6.a to get $SA(\texttt{ababbba})$.

[6 marks]

Answer



d. How do you compute the number of (different) factors of y using $SA(y)$? What is the running time of your computation?

[7 marks]

Answer

Let $X(p)$ be the number of words accepted by $SA(y)$ from state p. We have: $X[p] = \begin{cases} 1 & \text{if } deg(p) = 0, \\ 1 + \sum_{(p,a,q)\in F} X[q] & \text{otherwise,} \end{cases}$
where F is the set of arcs of the automaton. Then, the number of factors is $X[initial]$.

The computation takes $O(|y|)$ time with a mere traversal of the automaton.