Advanced Data Structures

Labwork 3: Data structures for operations on strings

November 2020

1. Construct the string-matching automaton for the pattern $P =$ aabab and illustrate its operation on the text string $T =$ aaababaabaababaab.

2. (Homework) Draw a state-transition diagram for a string-matching automaton for the pattern ababbabbababbababbabb over the alphabet $\{\mathtt{a}, \mathtt{b}\}$.

3. Construct the keyword tree and its failure links of the set of patterns

   $$\mathcal{P} = \{\mathtt{The}, \mathtt{hand}, \mathtt{and}, \mathtt{pork}, \mathtt{port}, \mathtt{pot}\}.$$

   Indicate a string-matching automaton which recognizes the occurrences of patterns in $\mathcal{P}$.

4. (Homework) Construct the keyword tree and its failure links of the set of patterns $\mathcal{P} = \{\mathtt{woman}, \mathtt{man}, \mathtt{meat}, \mathtt{animal}\}$. Indicate a string-matching automaton which recognizes the occurrences of patterns in $\mathcal{P}$.

5. The construction of the transition function of the string matching automaton for $O[1..m]$ described in Lecture 7 has time complexity $O(m^3 \cdot |\Sigma|)$. There are better methods to construct the transition function, with time complexity $O(m \cdot |\Sigma|)$.

   Write down the pseudocode of an algorithm that constructs the transition function in time $O(m \cdot |\Sigma|)$, and prove that the complexity of your algorithm is $O(m \cdot |\Sigma|)$.

6. Draw the suffix tree and it suffix links for the text banana$.

7. (Homework) Draw the suffix tree and its suffix links for the text mamaia$.

8. (Homework) Draw the generalized suffix tree and its suffix links for the set of texts $\{\mathtt{tatar}, \mathtt{tabac}\}$.

Programming labwork

Write in C++ or Java a program which solves the following problem:

1. It reads a text T from a text file specified by the user
2. It reads from the terminal the number z of strings (patterns) $P_1, P_2, \ldots, P_z$
3. It reports all positions from T where there is an occurrence of a patterns P_i $(1 \leq i \leq z)$

The interaction of the user with the program should be as follows:

```
Enter the source file for the text: file-name
Enter the number of patterns: $z$
Enter pattern 1: $P_1$
...
Enter pattern $z$: $P_z$
```

Afterwards, the program displays the occurrences of every pattern in text the T which was read from the text file *file-name*:

```
Pattern 1 occurs at positions $p_{1,1}$ ... $p_{1,n_1}$
...
Pattern $z$ occurs at positions $p_{z,1}$ ... $p_{z,n_z}$
```

The program should implement the Aho-Corasick algorithm which builds the keyword tree of the set of templates $\mathcal{P} = \{P_1, P_2, \ldots, P_z\}$ together with its failure links.

Illustrated example

Suppose that the file `source.txt` contains the text

```
Tim a mers la Timisoara sa-si cumpere o casa.
```

If we specify

```
Enter the source file for the text: source.txt
Enter the number of patterns: 4
Enter pattern 1: Tim
Enter pattern 2: Timis
Enter pattern 3: sa
Enter pattern 4: casa
```

then the program must display

```
Pattern 1 occurs at positions 1 15
Pattern 2 occurs at positions 15
Pattern 3 occurs at positions 25 43
Pattern 4 occurs at positions 41
```