Aho-Corasick algorithm

INFO-F-438 — Algorithms in Computational Biology. Assignment 3.

Algorithm

The algorithm was written in Python 3.5.3. Please see attached the algorithm code $ahoCorasick.py$. Explanations are in the form of comment lines.

Abstract data structures

The most important variables used in the implementation of this algorithm have the following abstract data structures:

- *GoToGraph* is an adjacency list of dictionaries. An adjacency list is a collection of unordered lists used to represent a finite graph[1]. Here, the unordered lists are dictionaries. Each dictionary represents a node of the graph and stores four keys:
    - *char*: a string that is the character the node represents. Eg. *'p'*.
    - *next_nodes*: a list of integers that represent the adjacent nodes, i.e. the nodes just below the current node in the graph. Eg. *[2, 6]*.
    - *fail*: the failure transition, i.e. an integer that represents the node where the machine should resume from if it cannot pursue on this specific path. Eg. *0*.
    - *output*: a string that is the keyword that finishes at this specific node (if any) and should be emitted by the machine once there. Eg. *'pattern'*.
- *matches* (in *matchMachine* function) is a list of dictionaries. Each dictionary in *matches* represent one match, i.e. one keyword found in the text. Each match dictionary stores two keys:
    - *keyword*: a string that is the keyword (from input *keywords* list) that was found and emitted by the machine. Eg. *'pattern'*.
    - *location:* an integer that represents the index of keyword's first character in the text. Eg. *859*.

[1] Wikipedia : https://en.wikipedia.org/wiki/Adjacency_list

Output

Assignment 3 example

Input

Keywords: *pattern*, *tree*, *state*, *prove*, *the*, *it*.

Text: "*As discussed in the session on Combinatorial Pattern Matching, keywords trees provide an efficient solution to search for multiple k patterns in a text of length n. The algorithm requires first the construction of a keyword tree and then, using naïve threading, the patterns can be identified in O(nm), where n is the average length of the k patterns and m is the length of the text. Alfred Aho and Margaret Corasick proposed in 1975 a more efficient solution that allows one to identify the patterns in O(m) time. To achieve this improvement, the keyword tree is replaced by a finite state pattern matching automata. Once this machine is constructed the text can be processed and the starting positions for the different patterns can be returned as output. The full specification of the Aho-Corasick algorithm is provided in the original article included with this assignment.*"

Output

- *the* found at 16
- *tree* found at 72
- *pattern* found at 133
- *it* found at 174
- *the* found at 194
- *tree* found at 224
- *the* found at 233
- *the* found at 262
- *pattern* found at 266
- *the* found at 314
- *the* found at 336
- *pattern* found at 342
- *the* found at 360
- *the* found at 374
- *the* found at 488
- *pattern* found at 492
- *prove* found at 533
- *the* found at 544
- *tree* found at 556
- *it* found at 581
- *state* found at 585
- *pattern* found at 591
- *the* found at 651
- *the* found at 681
- *it* found at 697
- *the* found at 708
- *pattern* found at 722
- *the* found at 784
- *it* found at 808
- *the* found at 828
- *it* found at 859

Aho and Corasick 1975 example

Input

Keywords: *he, she, his, hers.*

Text[2]: "*He threw his shoes. She threw hers.*"

Output

- *he found at 0*
- *his* found at 9
- *she* found at 20
- *he* found at 21
- *he* found at 30
- *hers* found at 30

GoTo graph graphical representation

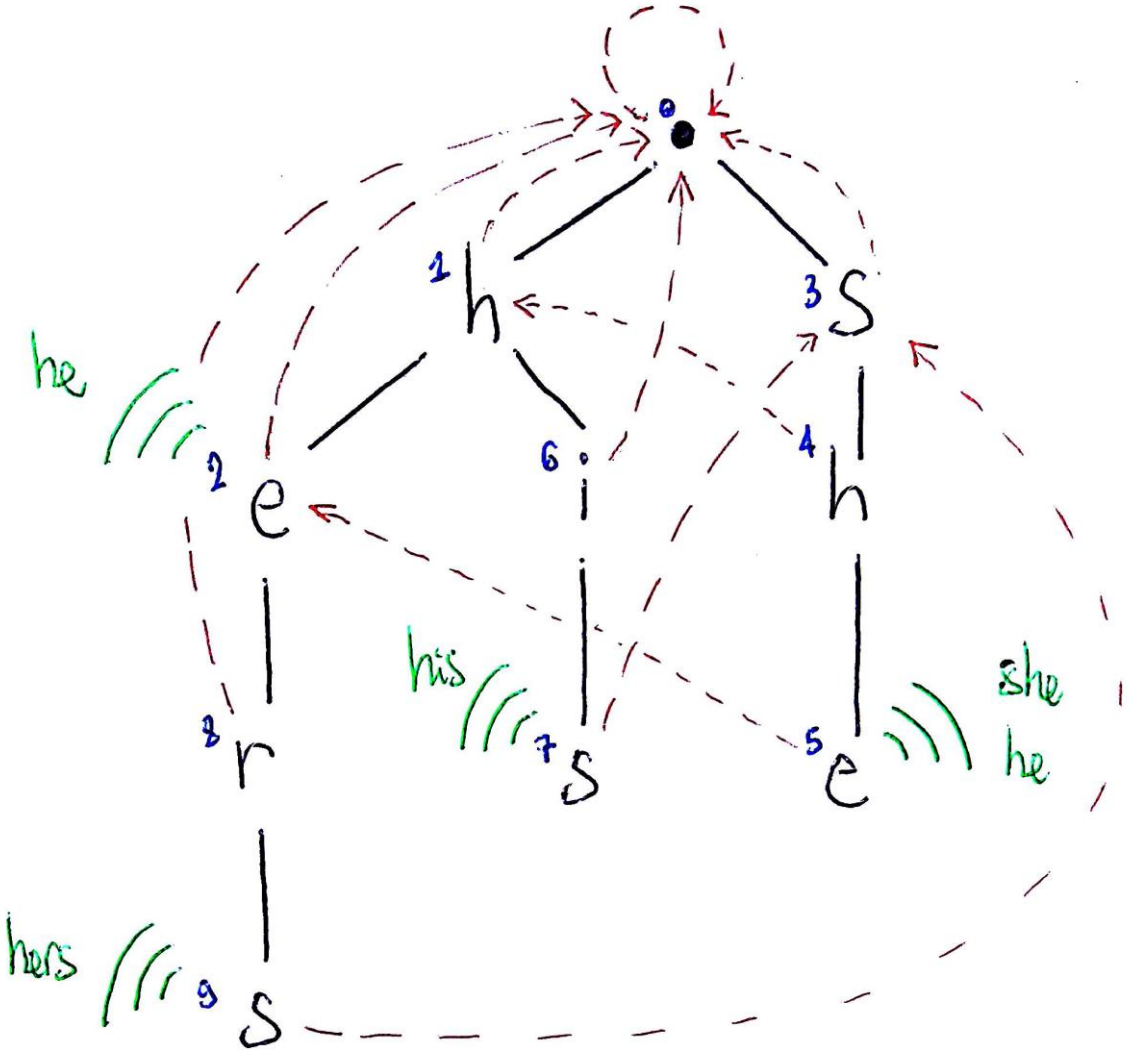

Figure 1 | Example of a GoTo graph. The red arrows represent the failure transitions; the outputs (when the machine "emits") are in green.

[2] This text is my own. No input text is provided in Aho and Corasick 1975 paper.