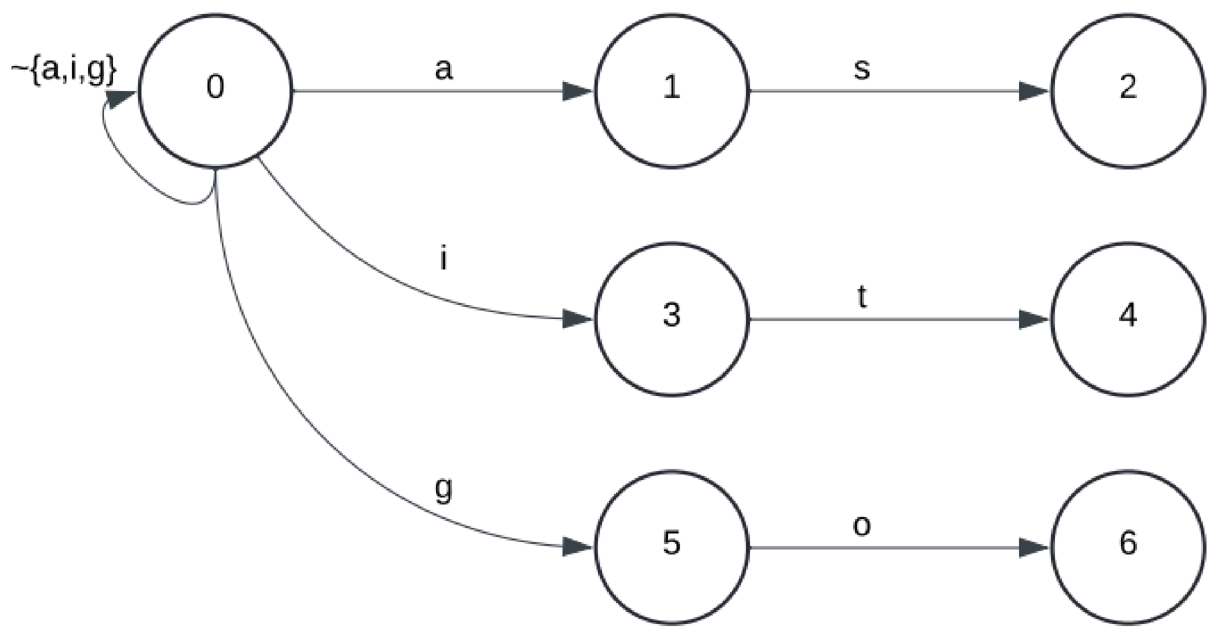
String Pattern Matching

Reference:
Aho, A. V., & Corasick, M. J. (1975). Efficient string matching: An aid to bibliographic search. *Communications of the ACM, 18*(6), 333–340. https://doi.org/10.1145/360825.360855

1)Chosen a simple set of 3 strings : **{“as”, “it”, “go”}**
Pattern Matching Machine consists of Goto, failure and output functions as shown below.

Fig1: Pattern Matching Machine

(a) Goto Function



(b) Failure Function

i	1	2	3	4	5	6
f(i)	0	0	0	0	0	0

(c) Output Function

iI	output(i)
2	as
4	it
6	go

Computation of goto function:
All our pattern matching machines have the property that g (0, ~r) # fail for all input symbols tr. We shall see that this property of the goto function on state 0 ensures that one input symbol will be processed by the machine in every machine cycle.

goto(0, 'a') = 1, goto(1, 's') = 2,
goto(0, 'i') = 3, goto(3, 't') = 4,
goto(0, 'g') = 5, goto(5, 'o') = 6

Computation of failure function:
The failure function is constructed from the goto function. Let us define the depth of a state s in the goto graph as the length of the shortest path from the start state to s. Thus in above Figure 1 (a), the start state is of depth 0, states 1,3 and 5 are of depth 1, states 2, 4, and 6 are of depth 2.
Computed the failure function for all states of depth 1, then for all states of depth 2 until the failure function has been computed for all states (except state 0 for which the failure function is not defined).

The algorithm to compute the failure function f at a state is conceptually quite simple. We make f(s) --0 for all states of depth 1. Now suppose f has been computed for all states of depth less than d. The failure function for the states of depth d is computed from the failure function for the states of depth less than d. The states of depth d can be determined from the nonfail values of the goto function of the states of depth d- 1. Specifically, to compute the failure function for the states of depth d, we consider each state r of depth d--1 and perform the following actions.
1. If g(r, a) =failfor all a, do nothing.
2. Otherwise, for each symbol a such that
g(r, a) -- s, do the following:
(b) Execute the statement state'-f (state) zero or
more times, until a value for state is obtained
such that g(state, a)#fail. (Note that since
g(O, a) # fail for all a, such a state will always
be found.)
(c) Setf(s) --g(state, a).

For the states of depth 1:
failure function for 1,3 and 5 which are of the states of depth 1.
f(1)=f(3)=f(5)=0 since depth is 1

For the states of depth 2:
failure function for 2,4 and 6 which are of the states of depth 2.

f(2):
Set state =f(1)
f(1)=0 since g(0,'s') = fail
Hence, f(2)=0

f(4):
Set state=f(3)
As f(3)=0
Then g(0,'t')= fail
Hence, f(4) = 0.

f(6):
Set state = f(5)
As f(5)= 0
Then g(0, 'o') =fail
Hence, f(6) = 0.

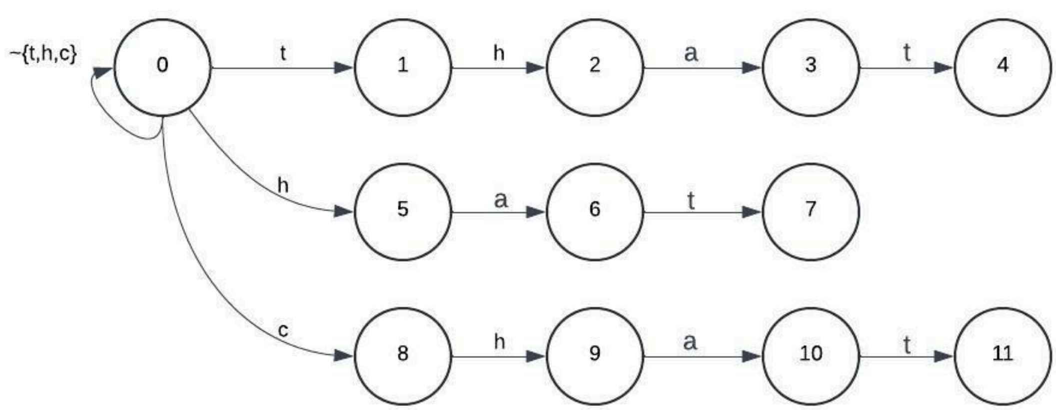
Hence,
The output “as” is associated with state 2
The output “it” is associated with state 4
The output “go” is associated with state 6

—---

The above set of strings resulted in the failure states of 0. Considering the below set of strings to illustrate non-zero failure states of output.

{“that”, “hat”, “chat”}

(a) Goto Function



(b) Failure Function

i	1	2	3	4	5	6	7	8	9	10	11
f(i)	0	5	6	7	0	0	1	0	5	6	7

(c) Output Function

iI	output(i)
4	{that}
7	{hat}
11	{chat, hat}

Computation of goto function:
goto(0, 't') = 1, goto(1, 'h') = 2, goto(2, 'a') = 3, goto(3, 't') = 4
goto(0, 'h') = 5, goto(5, 'a') = 6, goto(6, 't') = 7
goto(0, 'c') = 8, goto(8, 'h') = 9, goto(9, 'a') = 10, goto(10, 't') = 11

Computation of failure function:
Using the failure function algorithm:
For all the input symbols that can be reached directly through root ,we set f(s)=0
f(1) = 0, f(5) = 0, and f(8) = 0

f(2):
state = f(1) = 0.
goto(0, 'h') = 5, set f(2) = 5
Hence, f(2) = 5

f(3):
state = f(2) = 5.
goto(5, 'a') = 6, set f(3) = 6.
Hence, f(3)=6

f(4):
state = f(3) = 6.
As goto(6, 't') = 7, set f(4) = 7
Hence, f(4)=7

f(6):
state = f(5) = 0.
Since goto(0, 'a') = fail
Hence, f(6) = 0

f(7):
state = f(6) = 0.
Since goto(0, 't') = 1, set f(7) = 1.
Hence, f(7) = 1.

f(9):
state = f(8) = 0.
Since goto(0, 'h') = 5, set f(9) = 5.
Hence, f(9) = 5.

f(10):
state = f(9) = 5.
As goto(5, 'a') = 6, set f(10) = 6
Hence, f(10) = 6

f(11):
state = f(10) = 6.
As goto(6, 't') = 7, set f(11) = 7
Hence, f(11) = 7