CSCI 1810 10/16

KMP Algorithm

Recall the failure function:

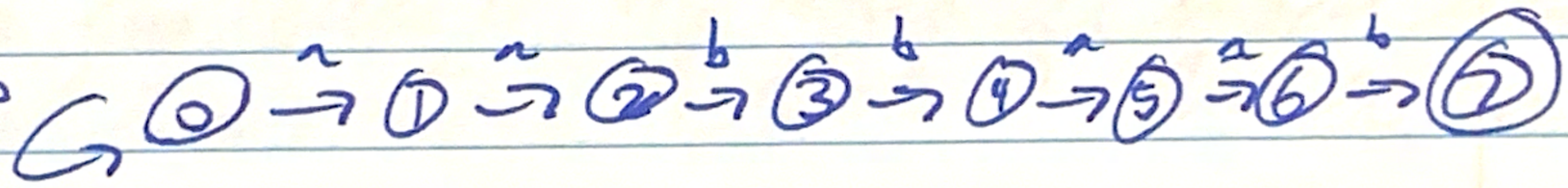
↳ Ex.

i	1	2	3	4	5	6	7
f(i)	0	1	0	0	1	2	3

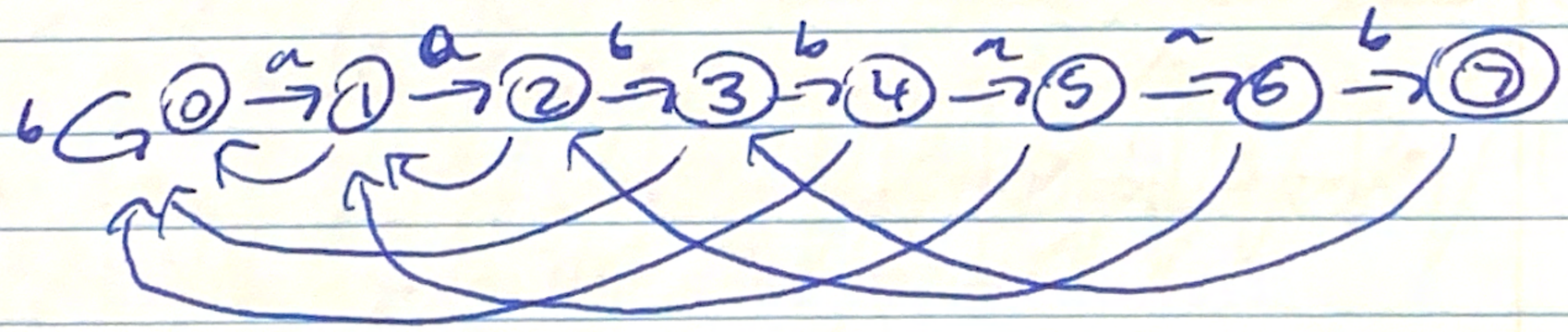
for p = aabbaab

↳ In words, the failure function, given an index i relative to a pattern p, returns the length of the longest proper prefix that is also a proper suffix of the string $p_1 p_2 \cdots p_i$.

To start the KMP algorithm, initialize a skeletal DFA for p as such:



The following is NOT the correct way of constructing the DFA, but it will help build intuition:



Here, the unlabeled edges are failure edges dictated by the failure function. On these failure edges, you change state without moving the input head.

Ex. For input text:

START a b a a b a a b b a a b END
0 1 0 1 2 3 1 2 3 4 5 6 7
FAIL GO FAIL GO

the above are the sequence of states visited.

CYC S1810 10/16

KMP Algorithm

To put what we did more formally, our machine M_p starts in state 0. Upon reading the first letter of the text t, M_p goes to state 1 because it reads in an "a". Since there is no transition from state 1 corresponding to the second letter of t, "b", M_p enters state f(1)=0 without moving its input head. From state 0, because there is a valid transition to state 0 via the letter "b", M_p moves its input head and transitions to state 0. And so on....

Therefore, so far we have a nondeterministic automata that accepts input texts containing a pattern p. How do we make this into a DFA? We'll do it by [illegible]

Let $M=(S, A, \delta, 0, \{l\})$ with states $S=\{0,1,2,\dots,l\}$.

FOR $j=1$ to l DO $\delta(j-1, p_j)=j$

FOR each $a \in A$, $a \neq p_1$, do $\delta(0,a)=0$

FOR $j-1$ to l DO

for each $a \in A$ and $a \neq p_{j+1}$ do

$\delta(j,a)=\delta(f(j),a)$.

} KEY POINT TO MAKE IT DETERMINISTIC

CYCS 1810 10/16

Failure Function Algorithm

Algorithm For the Failure Function

Input: pattern $p = p_1 p_2 \cdots p_\ell$
Output: $f(1), f(2), \ldots, f(\ell)$

```
BEGIN
L1      f(1)=0 ; i=0
L2      FOR j=2 to ℓ DO
L3          i ⇐ f(j-1)
L4          WHILE p_j ≠ p_{i+1} and i>0 DO
                i ⇐ f(i)
L5          IF p_j ≠ p_{i+1} and i=0 THEN
                f(j) ⇐ 0
L6          ELSE
                f(j) ⇐ i+1
```

Key: L1: Initialization. Remember that the failure function gives the length of the longest PROPER prefix that is also a proper suffix for the pattern up to index j. Thus, $f(1)=0$ and not 1. In this algorithm, i is simply a helper variable to keep track of "falling back".
L2: we are computing $f(j)$ for all $j=1,\ldots,\ell$.
L3: In this step, i gets assigned to be the length of the longest proper prefix that's also a suffix of $p_1 p_2 \cdots p_{j-1}$. In other words, we want to find $f(j)$ but we must start by considering $f(j-1)$.

CSC/180 10/16

Failure Function Algorithm

KEY CONTINUED:

L4: At this point, we know that $p_1 p_2 \cdots p_i$ is both a proper prefix and proper suffix of $p_1 p_2 \cdots p_{j-1}$. Therefore, we ask what if we add one more letter? Is $p_1 p_2 \cdots p_{i+1}$ also a proper prefix and suffix of $p_1 p_2 \cdots p_j$? As long as p_{i+1} and p_j don't match, we know this isn't the case, so we keep "falling back" by repeatedly doing $i \leftarrow f(i)$.

L5: If we have tried every possible i to see if $p_1 p_2 \cdots p_{i+1}$ is a proper prefix and suffix of $p_1 p_2 \cdots p_j$, and that has failed all those times, then we now no there is no proper prefix that also is a proper suffix of $p_1 p_2 \cdots p_j$, and so $f(j) \leftarrow 0$.

L6: On the other hand, in the process of "falling back", if we found such and i where $p_{i+1} = p_j$ and therefore $p_1 p_2 \cdots p_{i+1}$ is a proper prefix and proper suffix of $p_1 p_2 \cdots p_j$, we now know that $f(j) \leftarrow i+1$.