Home Work 3

Francisco J. Díaz Riollano
Student ID: 802-15-2172

September 26, 2018

```
1: procedure NAIVE(On input S[1,..m])
2:     n ← |T|
3:     m ← |S|
4:     for s=0 to n-m do
5:         if P[1,..m] = T[s+1...s+m] then
6:             print("pattern found")
```

Question 1

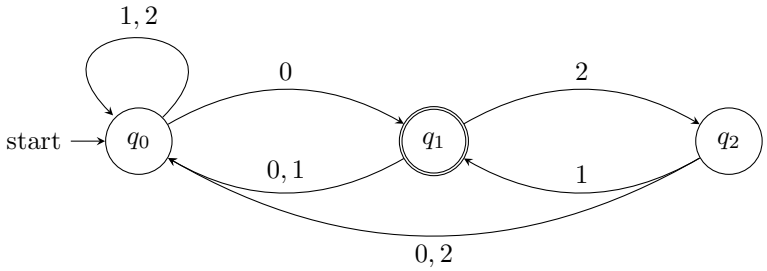
The string matching problem is defined as: "Given a text $T = T_1...T_n$ which is stored as array $T = T[1, ..., n]$, and a pattern $P = P_1...P_m = P[1...m]$ with $m < n$, where both are strings over the same alphabet Σ; decide whether S is a substring of T.

Algorithm 1 is the so-called naive-pattern finding algorithm. Use Algorithm 1 to construct a Finite State. Automata(deterministic or non-deterministic) for solving the matching problem.

Let M $= (Q, \Sigma, \delta, q_0, F)$ be 5-tuple NFSA where:
Q is a finite set of states
Σ is a finite set if input symbols
δ is a state transition function from $Q \times \Sigma \to Q$
$q_0 \in Q$ is the initial state
$F \subseteq Q$ Q is the set of final states
Let $\Sigma = \{s_1, s_2, ..., s_{|A|}\}$ for the alphabet of the input string and the text.

High Level Description of the automata:
M =: "On input S[1,...,m]"
1) Read the input strings S.
2) The automata checks if the string that is being read matches. 3) Otherwise it loops to the state of the closest seen pattern
4) If there is no previous seen pattern it loops to the start state
5) If the string read matches the P_jth pattern then continue to the $P_{j+1}th$ pattern state.
6) If all the pattern states have been visited, then this implies that a pattern has been found thus the machine "accepts".

An simple example: Let M be a machine such that it describes the pattern $P =$ "021" .
$\Sigma = \{0, 1, 2\}, Q = \{q_0, q_1, q_2\}$. For the given pattern the Automata look as such.

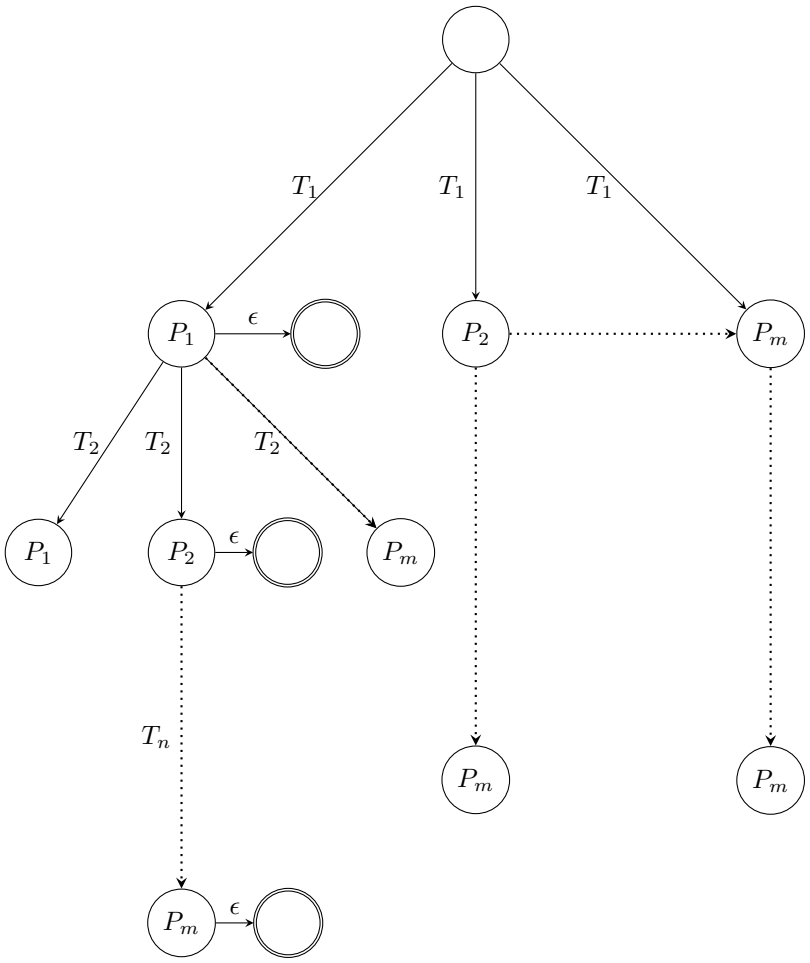


Question 2

Algorithm 1 returns a result in the time proportional to $O(|T||S|)$. Discuss the computation time of your automaton.

In the worst case the automata will have $O(|T||S|)$ complexity. We know this because the tree diagram of the Automata reads at most $|T|$. For each of these readings there are $|S|$ possible states to consider or that it compares.Thus the automata has the same time complexity as the algorithm.

In tree diagram below explores all of the possible text inputs from T_1 all the way to T_n. For the corresponding pattern we induce a empty transition to the acceptance state if the pattern.
We consider every possible comparison up to P_m. Note that the last P_m is not necessarily the only comparison we do of P_m, there may be multiple comparisons in between the tree diagram.


T_1
T_1
T_1
P_1
ϵ
P_2
P_m
T_2
T_2
T_2
P_1
P_2
ϵ
P_m
T_n
P_m
P_m
P_m
ϵ