String Matching Algorithms

⇒ Naive String Matching Algorithm

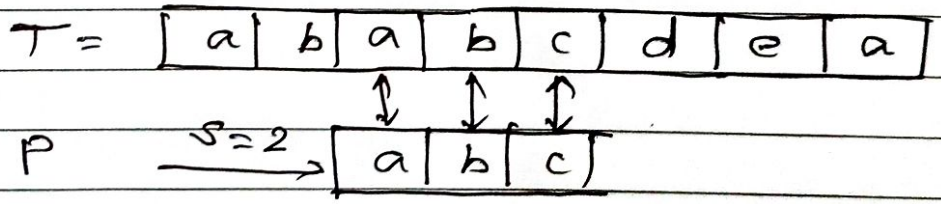
Problem: Assume that
text is an array $T[1 \ldots n]$
Pattern is an array $P[1 \ldots m]$
Length $(m) \leq$ Length (n)

- Elements of P and T are characters drawn from finite alphabet set Σ.
eg. $\Sigma = \{0, 1\}$ or $\Sigma = \{a, \ldots, z\}$

To find: To find valid shift 's' in given text 'T' so that pattern 'P' gets matched.
If No match, shift 's' is invalid.

- P occurs with shift s in T if $0 \leq s \leq n-m$ and $T[s+1 \ldots s+m] = p[1 \ldots m]$

E.g.



→ Algorithm finds all valid shifts using a loop that checks condition
$p[1 \ldots m] = T[s+1 \ldots s+m]$
for each of the $(n-m+1)$ possible values of s.

⇒ Algorithm Naive_String_Matcher(T, P)

{

1. n = T.length
2. m = P.length
3. for s = 0 to n-m:
4. if (P[1...m] == T[s+1 ... s+m])
5. Print "Pattern occurs with shift"
   Print s.

}

⇒ Analysis:

→ Worst Case scenario

Line 3 - for loop will check each possible shift.
It will run (n-m+1) time

Line 4 - It will check for all m characters in pattern

∴ Complexity = $O((n-m+1)m)$

Example:

T = | a | c | a | a | b | c |

P = | a | a | b |

n = 6

m = 3

∴ s = n - m

= 6 - 3 = 3

s = 0 | ✓ a | x a | b | Invalid

s = 1 → | x a | a | b | Invalid

s = 2 → | ✓ a | ✓ a | ✓ b | Valid shift

s = 3 → | ✓ a | x a | b | Invalid.