Lecture 15

Brute Force Algorithms & their Analysis: Pattern Matching Algorithm & its Time Complexity

String Matching

- Pattern
  - A string of m characters to search for
- Text
  - A (long) string of n characters to search in
- Brute Force Algorithm
  1) Align pattern at the beginning of text
  2) Moving from LEFT to RIGHT, compare each character of pattern to the corresponding character in text UNTIL
     - All characters are found to match (Successful search); or
     - A mismatch is detected
  3) While Pattern is not found and the text is not yet exhausted, re-align pattern one position to the RIGHT and REPEAT Step-2

Definitions

- *Formal Definition of String-Matching Problem*

  - *Assume text is an array T[1..n] of length n and the pattern is an array P[1..m] of length* $m \leq n$

  - *This basically means that there is a string array T which contains a certain number of characters that is larger than the number of characters in string array P. P is said to be the pattern array because it contains a pattern of characters to be searched for in the larger array T.*

Definitions

- Alphabet

It is assumed that the elements in P and T are drawn from a finite alphabet Σ.

-Example

- Σ = *{a, b, …z}*
- Σ = *{0,1}*

Sigma simply defines what characters are allowed in both the character array to be searched and the character array that contains the subsequence to be searched for.

Definitions

- Strings

- Σ^* denotes the set of all finite length strings formed by using characters from the alphabet

- The zero-length empty string denoted by ε *and is a member of* Σ^*

- The length of a string x is denoted by $|x|$

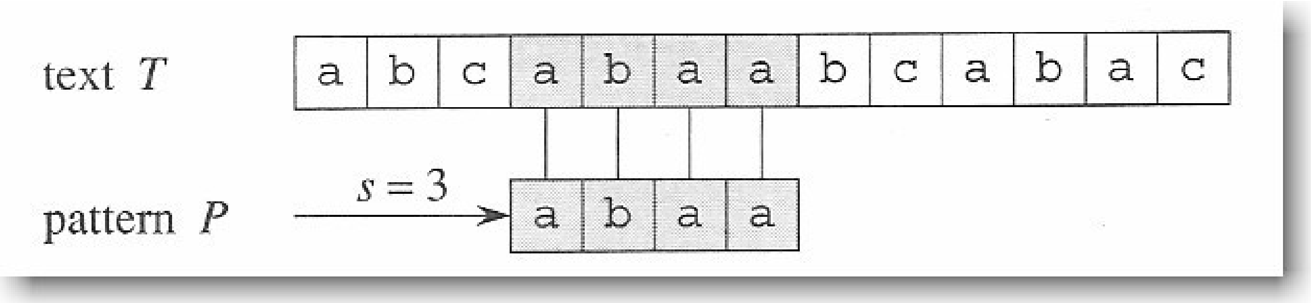
- The concatenation of two strings x and y, denoted xy, has length $|x| + |y|$ *and consists of the characters in x followed by the characters in y*

Definitions

- *Shift*

- *If P occurs with shift s in T, then we call s a valid shift*

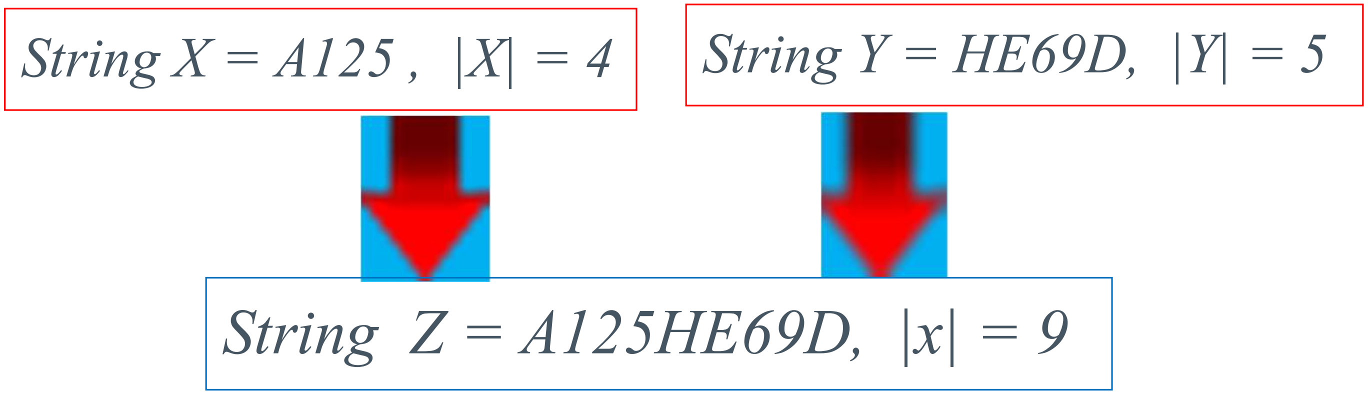
-*If P does not occur with shift s in T, we call s an invalid shift*



Definitions

- *String Concatenation Example*

$$\Sigma = \{A, B, C, D, E, H, 1, 2, 5, 6, 9\}$$



String $X = A125\ ,\ \ |X| = 4$

String $Y = HE69D,\ \ |Y| = 5$

String $Z = A125HE69D,\ \ |x| = 9$

The Concatenator

Definitions

- *Prefix*

  - *String w is a prefix of a string x if x = wy for some string y* ε Σ*

  - *w[x means that string w is a prefix of string x*

If a string w is a prefix of siring x this means that there exists some string y that when added onto the back of string w will make w = x

Definitions

- Prefix Examples

Σ = *{A,B}* Σ^* = *{A, B, AB, BA}*

Examples:

String x = AABBAABBABAB

String w =AABBAA

Is w[x ? Why?

To Prefix Or Not To Prefix

Definitions

- *Suffix*

- *String w is a suffix of a string x if x = yw for some y* ε Σ*

- *w]x means that string w is a suffix of string x*

If a string w is a suffix of string x this means that there exists some string y that when added onto the front of string w will make w = x

Definitions

- *Suffix Examples*

$\Sigma = \{A,B\}$ $\Sigma^* = \{A, B, AB, BA\}$

***Examples*:**

String x = AABBAABBABAB

String w = BABBA

Is w[x ? *Why?*

Et Tu Suffix?

Naïve String-Matching Algorithm

- Formal Definition of String-Matching Problem

- Assume text is an array T[1..n] of length n and the pattern is an array P[1..m] of length $m \leq n$

This basically means that there is a string array T which contains a certain number of characters that is larger than the number of characters in string array P. P is said to be the pattern array because it contains a pattern of characters to be searched for in the larger array T.

Basic Explanation

- The Naïve String-Matching Algorithm takes the pattern that is being searched for in the "base" string and slides it across the base string looking for a match. It keeps track of how many times the pattern has been shifted in variable s and when a match is found it prints the statement "Pattern Occurs with Shift s" .

- This algorithm is also sometimes known as the Brute Force algorithm.

Algorithm Pseudo Code

```
NAÏVE-STRING-MATCHER(T,P)
$N \leftarrow$ length $[T]$
$M \leftarrow$ length$[P]$
For $s \leftarrow 0$ to $n - m$
    do IF $P[1 \ldots m] = T[s+1 .. S+m]$ THEN
        PRINT “Pattern Occurs with shift” $s$
```

- This algorithm is also sometimes known as the

Brute Force algorithm.

Algorithm Time Analysis

```
NAÏVE-STRING-MATCHER(T,P)
N ← length [T]
M ← length[P]
For s ← 0 to n –m
    do IF P[1…m] = T[s+1 .. S+m] THEN
        PRINT "Pattern Occurs with shift" s
```

- The worst case is when the algorithm has a substring to find in the string it is searching that is repeated throughout the whole string. An example of this would be a substring of length am that is being searched for in a substring of length an.

Algorithm Time Analysis

The algorithm is $O((n-m)+1)*m)$

↓

Inclusive subtraction

n = length of string being searched

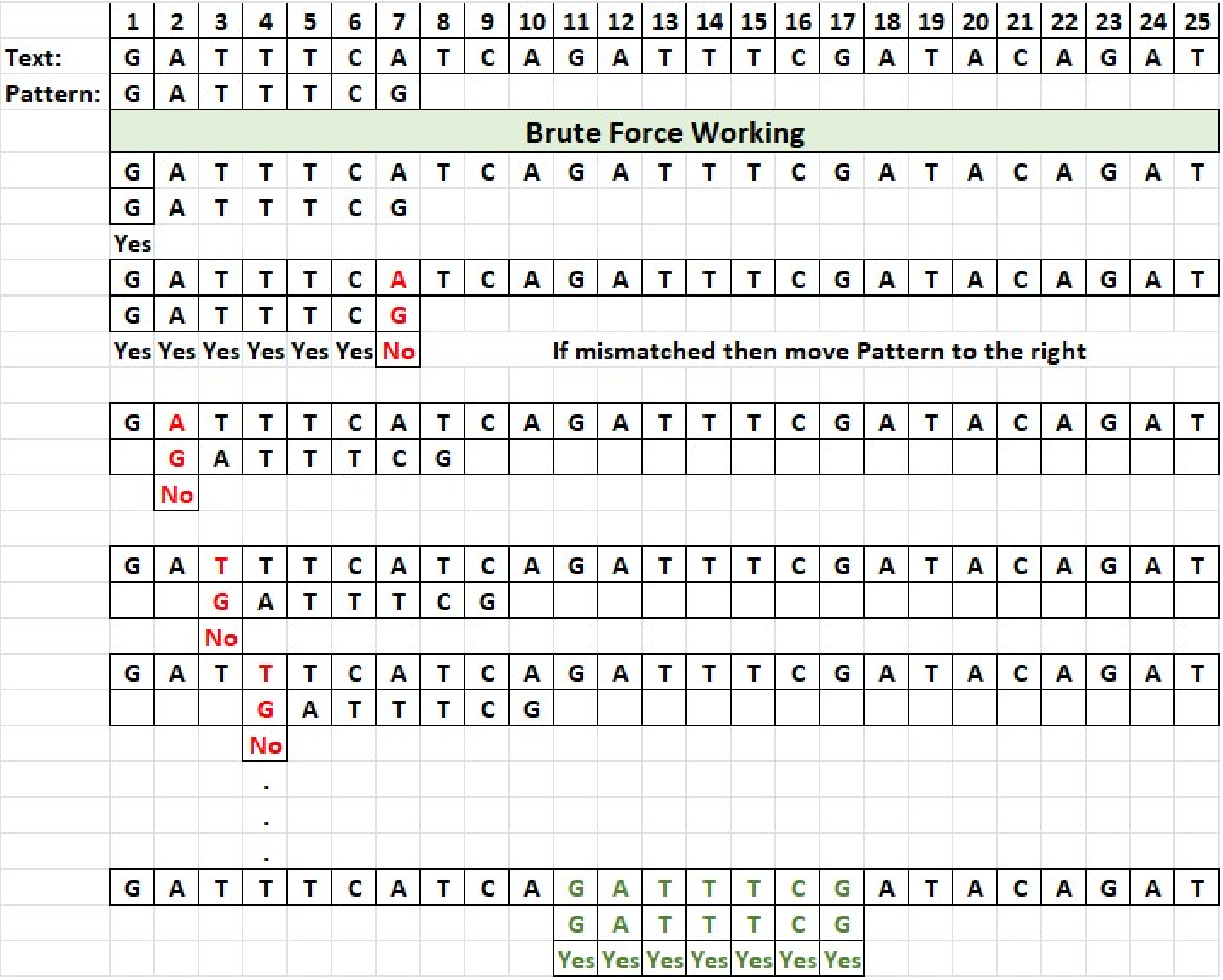
m = length of substring being compared

***Comments**:*

- The Naïve String Matcher is not an optimal solution

- It is inefficient because information gained about the text for one value of s is entirely ignored in considering other values of s.

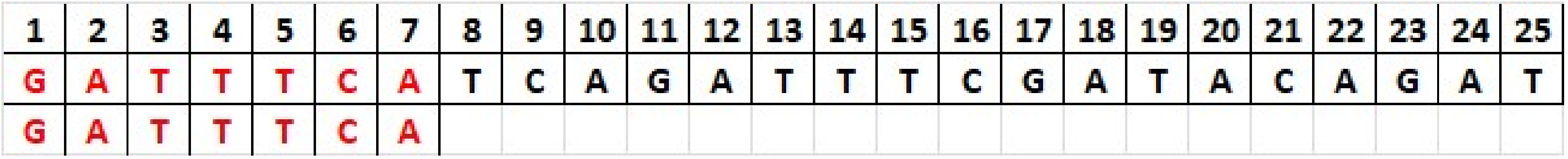
Brute Force Working



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Text:	G	A	T	T	T	C	A	T	C	A	G	A	T	T	T	C	G	A	T	A	C	A	G	A	T
Pattern:	G	A	T	T	T	C	G																		
	Brute Force Working																								
	G	A	T	T	T	C	A	T	C	A	G	A	T	T	T	C	G	A	T	A	C	A	G	A	T
	G	A	T	T	T	C	G																		
	Yes																								
	G	A	T	T	T	C	A	T	C	A	G	A	T	T	T	C	G	A	T	A	C	A	G	A	T
	G	A	T	T	T	C	G																		
	Yes	Yes	Yes	Yes	Yes	Yes	No				If mismatched then move Pattern to the right														
	G	A	T	T	T	C	A	T	C	A	G	A	T	T	T	C	G	A	T	A	C	A	G	A	T
		G	A	T	T	T	C	G																	
		No																							
	G	A	T	T	T	C	A	T	C	A	G	A	T	T	T	C	G	A	T	A	C	A	G	A	T
			G	A	T	T	T	C	G																
			No																						
	G	A	T	T	T	C	A	T	C	A	G	A	T	T	T	C	G	A	T	A	C	A	G	A	T
				G	A	T	T	T	C	G															
				No																					
				.																					
				.																					
				.																					
	G	A	T	T	T	C	A	T	C	A	G	A	T	T	T	C	G	A	T	A	C	A	G	A	T
											G	A	T	T	T	C	G								
											Yes	Yes	Yes	Yes	Yes	Yes	Yes								

Time Complexity: Best, Worst case

- Pattern Length: m=7
- Text Length: n
- Best Case: O(m)



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
G	A	T	T	T	C	A	T	C	A	G	A	T	T	T	C	G	A	T	A	C	A	G	A	T
G	A	T	T	T	C	A																		

  - The pattern is found right away. However, still must do m comparison to verify that pattern is found.

Time Complexity: Best, Worst case

- Pattern Length: m=7
- Text Length: n
- Worst Case: O(mn)

A A A A A A B

A A B After mismatch of B, move right – 15 comparison

  - After 15 comparison, match found

- The running time indeed belongs to O(m(n-m+1). However, Big-O notation is an upper bound, i.e. O(mn), as mn $\geq$ m(n-m+1)

Assignment Number 03

Submission Deadline: 09-Nov-2022

Assignment Number 03

- Write the steps of Bucket Sort algorithm [CLO-3]
  - Dry run it on the data 3, 7, 4, 9, 1, 2, 1, 5
  - Compute its time complexity
- Sort the given data 4, 2, 3, 3, 2, 3, 1, 7, 4, 11 by using the following algorithm with their steps and time & space complexity. [CLO-2]
  - Counting Sort Algorithm
  - Radix Sort
- Submission Deadline: 09-Nov-2022

Thank You!!!

Have a good day