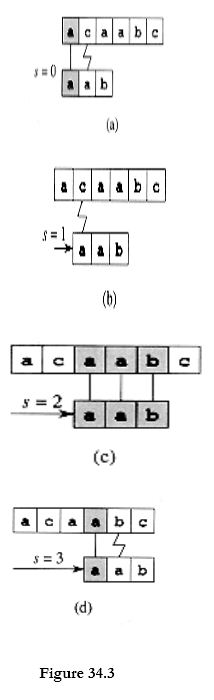
**String Matching Algorithms**

The naïve string matching Algorithms,

**Aim :** To implement the naïve string matching Algorithm

1. The naive algorithm finds all valid shifts using a loop that checks the condition P [1 . . m] = T[s + 1 . . s + m] for each of the n - m + 1 possible values of s.
2. NAIVE-STRING-MATCHER(T, P)
3. 1 n length [T]
4. 2 m length [P]
5. 3 for s 0 to n - m
6. 4 do if P [1 . . m] = T[s + 1 . . s + m]
7. 5 then print "Pattern occurs with shift" s
8. The naive string-matching procedure can be interpreted graphically as sliding a "template" containing the pattern over the text, noting for which shifts all of the characters on the template equal the corresponding characters in the text, as illustrated in Figure.
9. The for loop beginning on line 3 considers each possible shift explicitly.
10. The test on line 4 determines whether the current shift is valid or not; this test involves an implicit loop to check corresponding character positions until all positions match successfully or a mismatch is found.
11. Line 5 prints out each valid shift s.



1. The operation of the naive string matcher for the pattern P = aab and the text T = acaabc.
2. We can imagine the pattern P as a "template" that we slide next to the text. Parts (a)-(d) show the four successive alignments tried by the naive string matcher.
3. In each part, vertical lines connect corresponding regions found to match (shown shaded), and a jagged line connects the first mismatched character found, if any.
4. One occurrence of the pattern is found, at shift s = 2, shown in part (c).
5. Procedure NAIVE-STRING-MATCHER takes time ((n - m + 1) m) in the worst case. For example, consider the text string an (a string of n a's) and the pattern am.
6. For each of the n - m + 1 possible values of the shift s, the implicit loop on line 4 to compare corresponding characters must execute m times to validate the shift.
7. The worst-case running time is thus ((n - m + 1) m), which is (n2) if m = n/2 .
8. As we shall see, NAIVE-STRING-MATCHER is not an optimal procedure for this problem. Indeed, in this chapter we shall show an algorithm with a worst-case running time of O (n + m).
9. The naive string-matcher is inefficient because information gained about the text for one value of s is totally ignored in considering other values of s.
10. Such information can be very valuable, however. For example, if P = aaaband we find that s = 0 is valid, then none of the shifts 1, 2, or 3 are valid, since T [4] = b.
11. In the following sections, we examine several ways to make effective use of this sort of information. Example1:
12. 1) Input:
13. txt [] = “THIS IS A TEST TEXT”
14. pat [] = “TEST”
15. 2) Output:
16. Pattern found at index 10.
17. Example2:
18. 1) Input:
19. txt [] = “AABAACAADAABAAABAA”
20. pat [] = “AABA”
21. 2) Output:
22. Pattern found at index 0
23. Pattern found at index 9

Pattern found at index 13