CE-514 : Advanced Algorithms

Lab 5: String Matching- I

---

1. **Aim :** Write programs to implement Naive and Horsepool's string Matching algorithms.

2. **Objective** : Students should be able understand and implement basic string matching algorithms like Naive(Brute-force) and Horsepool. They should be able to differentiate between them and can do running time analysis.

3. Description: Given a text T of length N , T[0..N-1] and a pattern P of length M P[0..M-1], the task is to print all occurrences of P[] in T[]. You may assume that N > M. Pattern searching is an important problem in computer science. When we search for a string in a notepad/word file or browser or database, pattern searching algorithms are used to show the search results.

Examples:

Input: T[] = "THIS IS A TEST TEXT", P[] = "TEST"

Output: Pattern found at index 10

Input: T[] = "AABAACAADAABAABA", P[] = "AABA"

Output: Pattern found at index 0, Pattern found at index 9, Pattern found at index 12

4. Algorithm 1: Naive_Matcher(int T[], int P[])

```
1.    M = P.length,  N = T.length;
2.  /* A loop to slide P[] one by one */
3.    for (int i = 0; i <= N - M; i++)
4.    { /* For current index i, check for pattern match */
5.       for (j = 0; j < M; j++)
6.         {  if (T[i + j] != P[j])
7.              break ;
8.         }
```

```
9.          if (j  == M) // if P[0...M-1] = T[i, i+1, ...i+M-1]
10.             printf("Pattern found at index/shift  %d \n", i);
11.    }
```

Algorithm 2: Horsepool's String Matching:

The pattern is checked with the text from right to left and progresses left to right through the text. Horspool's algorithm shifts the pattern by looking up the shift value in the character of the text aligned with the last character of the pattern in the table made during the initialization of the algorithm. Let c be the character in the text that aligns with the last character of the pattern. If the pattern does not match there are 4 cases to consider.

Algorithm 2.1: Preprocessing Algorithm

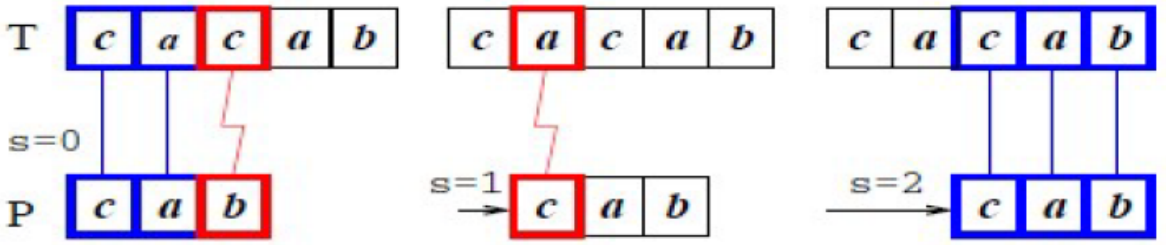
Input: text $T = T[0 \ldots n)$, pattern $P = P[0 \ldots m)$
Output: position of the first occurrence of P in T
Preprocess:
(1) for $c \in \Sigma$ do $shift[c] \leftarrow m$
(2) for $i \leftarrow 0$ to $m-2$ do $shift[P[i]] \leftarrow m-1-i$

Algorithm 2.2: String Matching

Input: text $T = T[0 \ldots n)$, pattern $P = P[0 \ldots m)$
Output: position of the first occurrence of P in T
Preprocess:
(1) for $c \in \Sigma$ do $shift[c] \leftarrow m$
(2) for $i \leftarrow 0$ to $m-2$ do $shift[P[i]] \leftarrow m-1-i$
Search:
(3) $j \leftarrow 0$
(4) while $j + m \leq n$ do
(5) if $P[m-1] = T[j+m-1]$ then
(6) $i \leftarrow m-2$
(7) while $i \geq 0$ and $P[i] = T[j+i]$ do $i \leftarrow i-1$
(8) if $i = -1$ then return j
(9) $j \leftarrow j + shift[T[j+m-1]]$
(10) return n

5. Example:

Example 1: Naive algorithm



Complexity : Θ (m(n-m+1)) is equivalent to Θ(mn) .

Example 2: Horsepool's Algorithm

Text: JIMY_HAILED_THE_LEADER_TO_STOP
Pattern: LEADER

```
JIMY_RAN_AND_HAILED_THE_LEADER_TO_STOP
     ||       |  |       |       ||   |
LEADER        |  |       |       ||   |
       LEADER  |       |       ||   |
         LEADER       |       ||   |
                LEADER       ||   |
                       LEADER|   |
                         LEADER   |
                             LEADER
```

The worst case cost is $\Theta(nm)$, but for random text is $\Theta(n)$.

6. Implementation Notes: You can use Array to implement the String Matching algorithms.

7. Exercise:

1. What is the Time Complexity of Naive String Matching algorithm? Justify.
2. Create a shift table for the pattern P= “umbrella”.
3. What is the Time Complexity of the Horsepool’s String Matching algorithm? Justify.
4. Trace String Matching using Naive algorithm over following inputs:

   T= “JIM SAW ME IN A BOOK SHOP” P= “SHOP”
5. Trace String Matching using the Horsepool’s algorithm over following inputs:

   T= “JIM SAW ME IN A BOOK SHOP” P= “SHOP”
6. Compare and contrast Naive and Horsepool’s algorithm.
7. What is the worst case scenario of the Horsepool’s algorithm? Justify your answer.

8. References:

- Introduction to Algorithms Third Edition by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
- COMPUTER ALGORITHMS by Ellis Horowitz University of Southern California Sartaj Sahni University ofFlorida Sanguthevar Rajasekaran University of Florida
- Web Resource: www.geeksforgeeks.org