# Brute force String Search

A brute force string matching algorithm is quite obvious. Align the pattern against the first **m** characters of the text and start matching the corresponding pairs of characters from left to right until either all **m** pairs of characters match or a mismatching pair is encountered.

In the latter case, shift the pattern one position to the right and resume character comparisons, starting again with the first character of the pattern and its counterpart in the text.

**Example:**
Text: N O B O D Y_S A W _ M E
Pattern: S A W

```
N O B O D Y_S A W _ M E
S A W
  S A W
    S A W
      S A W
        S A W
          S A W
            S A W
              S A W
```

```
ALGORITHM BruteForceStringMatch(T[0...n-1], P[0...m-1])
// Implements brute force string match
// Input:   An array T[0...n-1] of n characters representing a text and an array P[0...m-1] of m
//          characters representing a pattern
// Output: The index of the first character in the text that starts a matching substring or
//          -1 if the search is unsuccessful
for i ← 0 to n-m do
    j ← 0
    while j < m and P[ j ] = T [ i + j ] do
        j ← j + 1
    if j = m
        return i
return -1
```

The worst case would be that the algorithm would have to make all the m comparisons before shifting the pattern and this can happen for n-m+1 tries. Thus, in the worst case, the algorithm efficiency is in **O(nm).**