Naive search

Let t be a text and x a word such that $|x| \leqslant |t|$, both using the alphabet Σ.

Formally, the problem of text search is stated as follows: determine if x is a factor of t and, if so, give the position of the first letter of x in y.

The simplest algorithm, often called **naive**, consists in comparing x to $t[k+1 \dots k+|x|]$ for all the $k \in \{0, 1, \dots, |t| - |x|\}$ in the worst case.

It helps to imagine the text is an array of characters and the word we are looking up is sliding along this array until we find a match. This is referred to as the **sliding window**, because we can think of a frame sliding along the text.

Naive search/Algorithm

The naive search algorithm we just described can be more formally described as

```
NAIVE(x, t)
i ← 1; j ← 1
while i ⩽ |x| and j ⩽ |t|
      do if t[j] = x[i]                  ▷ Compare a letter of x with a letter of t.
            then j ← j + 1; i ← i + 1    ▷ Schedule comparison with next letter in x.
            else j ← j − i + 2; i ← 1    ▷ Failed: we slide x on t and start again.
if i > |x|
   then ...                              ▷ Occurrence of x in t at position j − |x|.
   else ...                              ▷ No occurrences.
```

Naive search/Analysis

How many comparisons does the naive algorithm performs in the worst case?

In the worst case, x is not in t and the test at line 3 always fails on the last letter of x, leading to make x slide of one position at line 5 the latest as possible.

An example of positive match in the worst case is $t = \mathrm{a}^{m-1}\mathtt{b}$ and $x = \mathrm{a}^{n-1}\mathtt{b}$.

There are $n - m + 1$ positions in t to compare with the first letter of x, and since there are m letters in x, it makes a total of $(n - m + 1)m = nm - m^2 + m < |t| \times |x|$ positions.