String matching

Goldsmiths Computing

Motivation

- generalisation of search operation (sequences, not just single elements)
- applications include text editors, classifiers, information retrieval systems
- extensions used in
  - spelling checkers
  - DNA sequence matching
  - protein structure representations

Definition

String matching returns the smallest index at which the *pattern*, P, is found exactly in the *text*, T, or false if the pattern is not present in the text at all.

C++ `std::string::find()`

Java `java.lang.String.indexOf()`

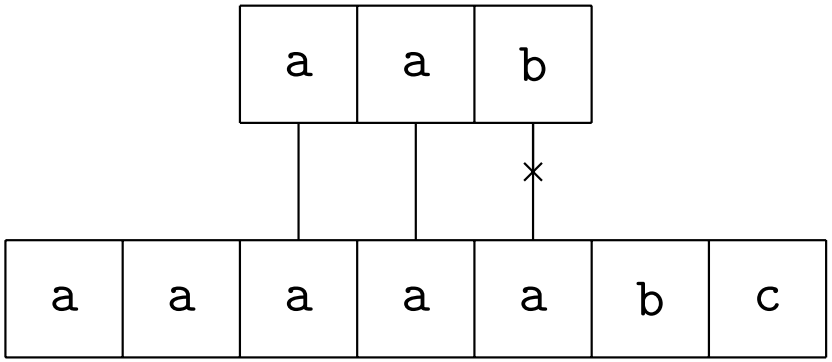
String matching algorithm

```
function MATCH(T,P)
    m ⟵ LENGTH(P)
    for 0 ≤ s ≤ LENGTH(T) - m do
        if T[s...s+m] = P[0...m] then
            return s
        end if
    end for
    return false
end function
```

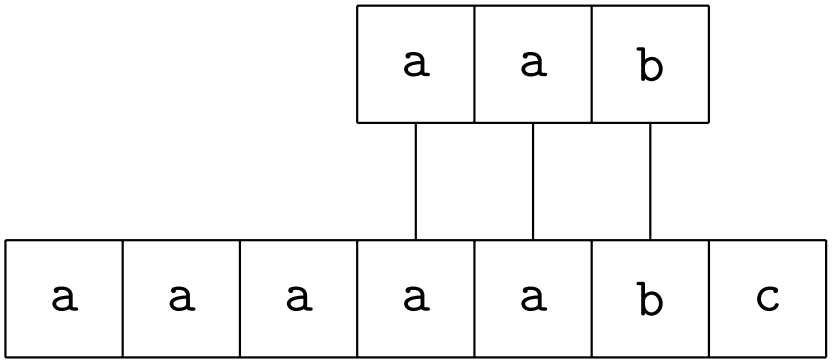
Naïve algorithm

```
function MATCH(T,P)
    m ← LENGTH(P)
    for 0 ≤ s ≤ LENGTH(T) - m do
        found ← true
        for 0 ≤ j < m do
            if T[s+j] ≠ P[j] then
                found ← false; break
            end if
        end for
        if found then
            return s
        end if
    end for
    return false
end function
```

Diagram



Diagram



Complexity analysis

space

- no particular requirements for additional storage

$$\Longrightarrow \Theta(1)$$

time

- outer loop happens $n - m + 1$ times (worst case)
- inner loop m times (worst case)

$$\Longrightarrow \Theta((n+1)m - m^2) \sim \Theta(nm)$$

For particular sizes of pattern:

small $m \sim c \Longrightarrow \Theta(n)$

large $m \sim n \Longrightarrow \Theta(n)$

intermediate $m \sim \frac{n}{2} \Longrightarrow \Theta(n^2)$

Work

1. Reading
   - CLRS, section 32.1
   - Drozdek, section 13.1.1 “Straightforward Algorithms”
2. Questions from CLRS

   Exercises 32.1-1, 32.1-2
3. Lab work
   - (week of 3rd December) implement naïve string match for strings of characters. Use `OpCounter` (remember that?) to count how many character comparisons happen in the worst case. Construct a table and verify the theoretical results in this lecture.