Knuth-Morris-Pratt matching

Goldsmiths Computing

Motivation

- deterministically $\Theta(m + n)$ string matching

Definition

Knuth-Morris–Pratt matching uses information about the pattern P to avoid redundant work when doing string matching.

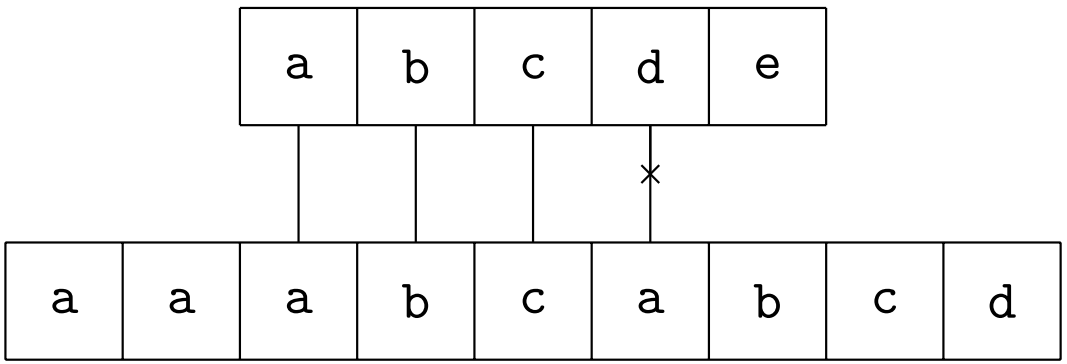
Example

Consider MATCH(abcde, text)

- all characters in P different
- mismatch in index position k
  - matches in all previous positions [0,k)
  - can safely advance next start position to k.

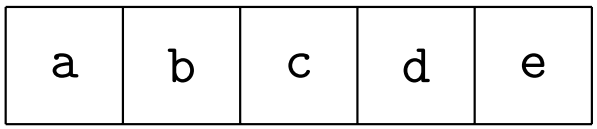
Diagram

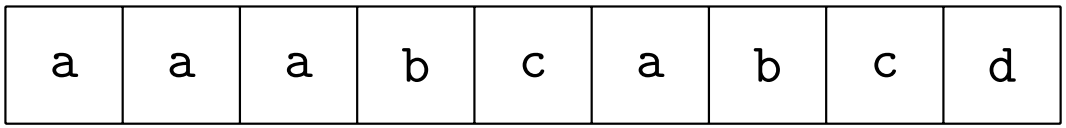
- all pattern characters different:



Diagram

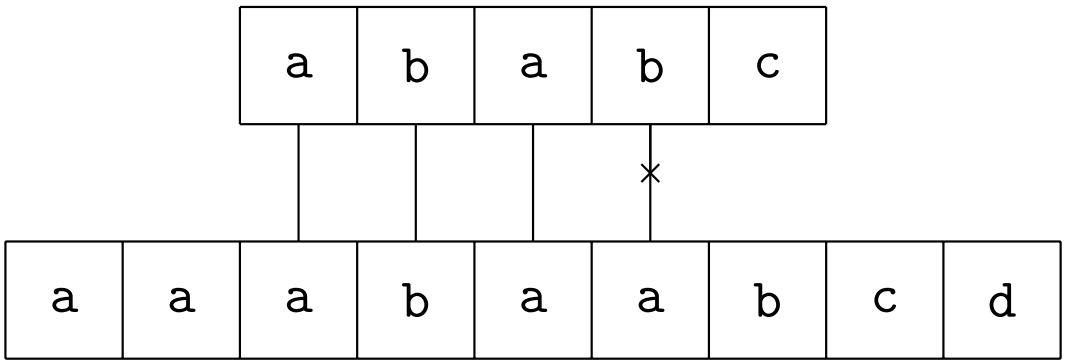
- all pattern characters different:





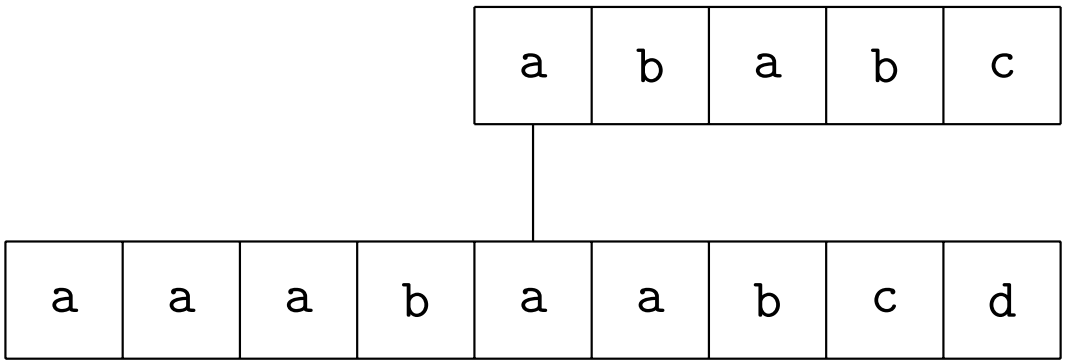
Diagram

- pattern contains similar suffixes:



Diagram

- pattern contains similar suffixes:



Prefix table

Also called “prefix function” or “failure function”

- encode for each index k the length of the longest prefix of the pattern P which is a suffix of the subsequence of the pattern P[0..k]

a	b	c	d	e
0	0	0	0	0

a	b	a	b	c
0	0	1	2	0

Knuth-Morris-Pratt algorithm

```
function KMPmatch(T,P)
    n ← length(T); m ← length(P)
    π ← computePrefix(P)
    q ← 0
    for 0 ≤ i < n do
        while q > 0 ∧ P[q] ≠ T[i] do
            q ← π[q-1]
        end while
        if P[q] = T[i] then
            q ← q + 1
        end if
        if q = m then
            return i - m + 1
        end if
    end for
    return false
end function
```

Knuth-Morris-Pratt algorithm: compute prefix

```
function computePrefix(P)
    m ← length(P)
    π ← new Array(m); π[0] ← 0
    k ← 0
    for 1 ≤ q < m do
        while k > 0 ∧ P[k] ≠ P[q] do
            k ← π[k-1]
        end while
        if P[k] = P[q] then
            k ← k + 1
        end if
        π[q] ← k
    end for
    return π
end function
```

Work

1. Reading
   - CLRS, section 32.4
   - Drozdek, section 13.1.2 “The Knuth-Morris-Pratt Algorithm”
     - NB: `next` table in Drozdek is very slightly different from result of computePrefix
2. Lab work
   - (week of 3rd December) implement Knuth-Morris-Pratt string match. Use `OpCounter` to count how many character comparisons happen in the best and worst cases, and verify the theoretical results in this lecture.