Wyszukiwanie wzorca (c.d.)

Algorytm KMP

dotąt sprawdziliśmy

T ababab | x ⇝ x ≠ a

P ababab | aaa

próbujemy sensowne przesunięcia P:

P abab | abaaa
ab | ababaaa
| abababaaa

$\pi : \{1..m\} \to \underline{m}$

$\pi(i) = \max\{k \mid k < i \wedge P_k \sqsupset P_i\}$

P_k jest sufiksem P_i

Przykład

p = ababada

i	1	2	3	4	5	6	7
π(i)	0	0	1	2	3	0	0

$P_1 =$ a
$P_2 =$ ab
$P_3 =$ aba
$P_4 =$ abab
$P_5 =$ ababa
$P_6 =$ ababad
$P_7 =$ ababada

```
KMP P T =
  let π = undefined P
  for i ∈ [1..n]
    while q>0 ∧ T[i] ≠ P[q+1]
      q = π q
    when P[q+1] == T[i]
      q++
    when q==m
      yield i
      q = π q
```

$$\text{foldl}\left(\lambda c\, q .\ \text{fix}\left(\lambda f\, q.\ \begin{array}{l}\text{if } q>0 \wedge c \neq P[q+1] \\ \text{then } f(\pi\, q) \\ \text{else } q\end{array}\right)\right)\ 0$$

Proste oszacowanie: $O(n \cdot m)$ – słabo

Lepsze: $\Theta(n)$ ← (sama pętla)

```
for i ∈ [1..n]
  while q>0 ∧ T[i] ≠ P[q+1]
    q = π q        ← zmniejszamy q o ≥1
  when P[q+1] == T[i]
    q++            ← zwiększamy q o 1
  when q==m
    yield i
    q = π q        ← zmniejszamy q o ≥1
```

zmniejszeń ≤ # zwiększeń ≤ n

Liczenie π: $\Theta(m)$

```
π[1] = 0
k = 0
for q ∈ [2..m]
  while k>0 ∧ P[k+1] ≠ P[q]
    k = π[k]
  if P[q+1] = P[k]
    k++
  π[q] = k
```

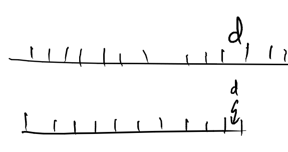
normalne zmienne →

```
π[1] = 0
q = 0
for i ∈ [2..m]
  while q>0 ∧ P[q+1] ≠ P[i]
    q = π[q]
  if P[q+1] = P[i]
    q++
  π[i] = q
```

Sumarycznie KMP: $\Theta(n+m)$

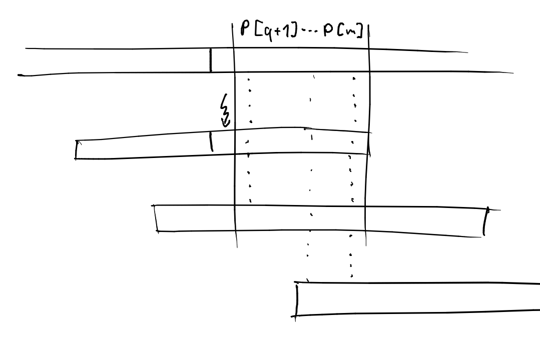
Algorytm Boyera-Moore'a

~ Sprawdzamy zgodność od końca przystawionego wzorca



~ jeśli w P nie ma d, to możemy od razu przesunąć o m

- w razie niezgodności 2 heurystyki
  - „dobry znak": k – ostatnie wystąpienie litery T[i] w P (ale nie cofamy wzorca)
  - „dobry sufiks"



Def

słowo Q jest podobne do słowa R $(Q \sim R)$
jeśli $Q \sqsupset R$ lub $R \sqsupset Q$

Ad „dobry sufiks": $k = \max\{k < m \mid P_k \sim P[i+1]..P[m]\}$

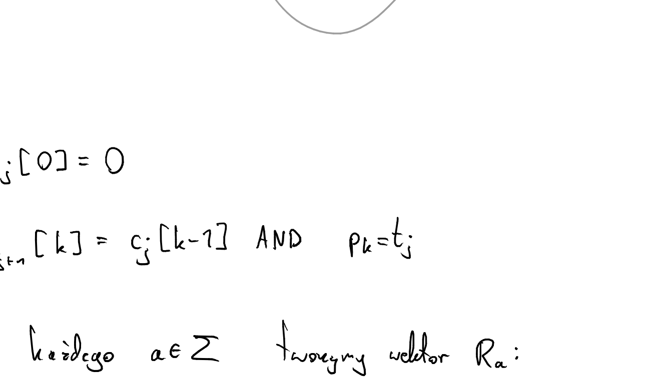
Algorytm Shift-And

- krótkie wzorce (|wzorzec| ~ |słowo maszynowe|)

c_j – wektor bool

$c_j[k] = 1 \iff P_k \sqsupset T_j$

$c_j \to c_{j+1}$



$c_j[0] = 0$

$c_{j+1}[k] = c_j[k-1]$ AND $p_k = t_j$

Dla każdego $a \in \Sigma$ tworzymy wektor R_a:

$R_a[0] = 1$
$R_a[i] \equiv T_i == a$

[dopełnia 1]

$c_{j+1} = c_j \gg 1$ AND $R_{t_{j+1}}$

wzorzec występuje na $j \iff c_j[m] = 1$

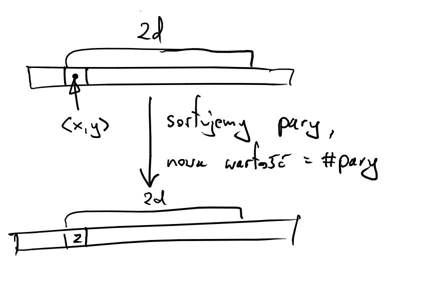
Algorytm KMR (Karl?-Miller?-Rosenberg)

Idea: nadajemy podsłowom numery

dla podsłów dł. $l = 1, 2, \ldots 2^k$

$l = 1 \;\to\; x \mapsto$ numer znaku w alfabecie (wedle jakiegoś porządku)

$l' = 2l \;\to$

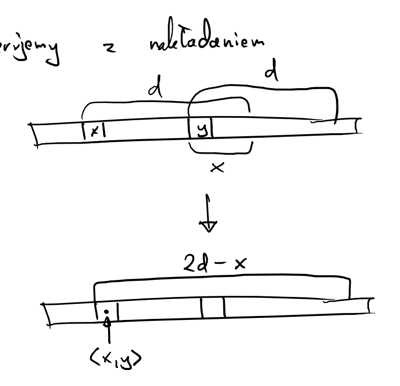


Wyszukiwanie wzorca:

Nadajemy numery podsłowom P#T

A co jeśli $|P| \neq 2^k$?

↳ numerujemy z nakładaniem



Czas: $\Theta(n \cdot \log m)$
$n \log n$?