Haladó informatikai algoritmusok

Mintaillesztés

Knuth-Morris-Pratt algoritmus

Boyer-Moore algoritmus

Knuth-Morris-Pratt (KMP) algoritmus

Lineáris idejű mintaillesztő algoritmus.
Kiküszöböli a δ átmeneti függvény kiszámítását és az illesztési ideje $\Theta(n)$.

Egy $\pi[1..m]$ **segédfüggvény**t használunk, amelyet előzetesen számítunk ki a minta alapján $\Theta(m)$ idő alatt.

A π tömb segítségével a δ átmeneti függvény értékei "menet közben" hatékonyan számolhatók.
Ez azt jelenti, hogy bármely $q = 0, 1, \ldots, m$ állapotra és $a \in \Sigma$ jelre, $\pi(q)$ érték adja meg $\delta(q, a)$ számítási módját és ez független a-tól.

A π tömbnek csak m eleme van, míg δ-nak $\sigma(m|\Sigma|)$, így az előfeldolgozás során megspóroljuk a $|\Sigma|$ tényezőt, ha π-t határozzuk meg δ helyett.

Alapötlet

A léptetés mértéke előre kiszámítható a mintára, továbbá a léptetés után az összehasonlítást nem kell a minta elejéről kezdeni.
Legyen a minta S és $m = |S|$.
A minta prefix függvénye: Egy minta prefix függvénye tartalmazza azokat az ismereteket, amelyek megadják, hogyan illeszkedik a minta önmaga eltoltjaira:

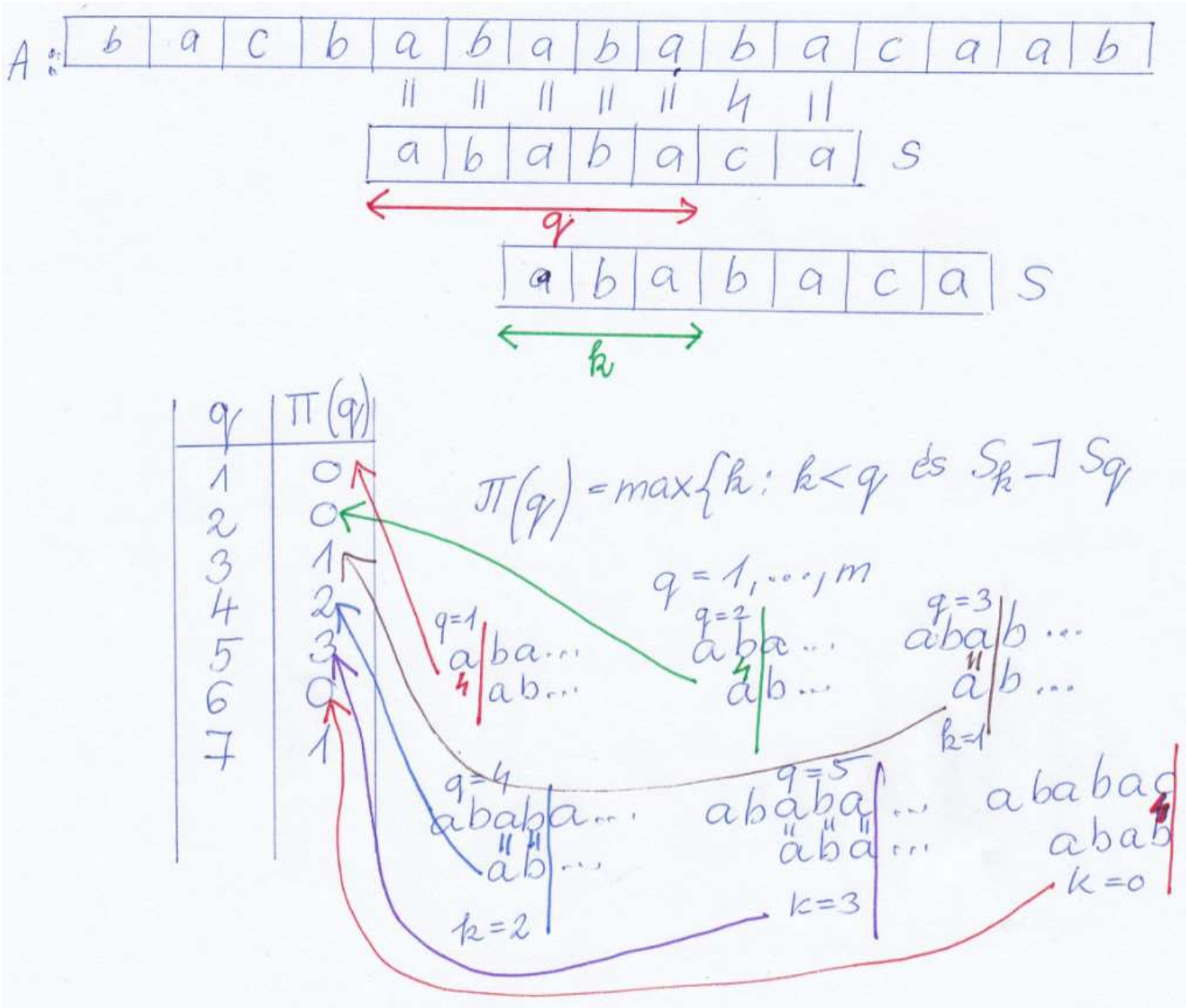
$$\pi : \{1, \ldots, m\} \rightarrow \{0, \ldots, m-1\}$$

$$\pi(q) = max\{k : k < q \; es \; S_k \; szuffixe \; S_q\} \quad q = 1, \ldots, m$$

(a leghosszabb olyan S-beli prefix hossza, amely valódi szuffixe S_q-nak)

Ha $S[1..q] = A[i - q \ldots i - 1]$, de $S[q+1] \neq A[i]$, akkor legalább annyit kell léptetni a mintát, hogy $S_k \; szuffixe \; S_q$ teljesüljön.

Példa


A:
b a c b a b a b a b a c a a b
a b a b a c a
s
q
a b a b a c a
s
k
q π(q)
1 0
2 0
3 1
4 2
5 3
6 0
7 1
π(q) = max{k : k<q és S_k ⊐ S_q
q = 1, ..., m
q=1
a ba...
ab...
q=2
aba...
ab...
q=3
abab...
ab...
k=1
q=4
ababa...
aba...
ab...
k=2
q=5
ababaa...
ababa...
aba...
k=3
ababac
ababa
k=0

Eljárás

KMP-ILLESZTO(T,P)

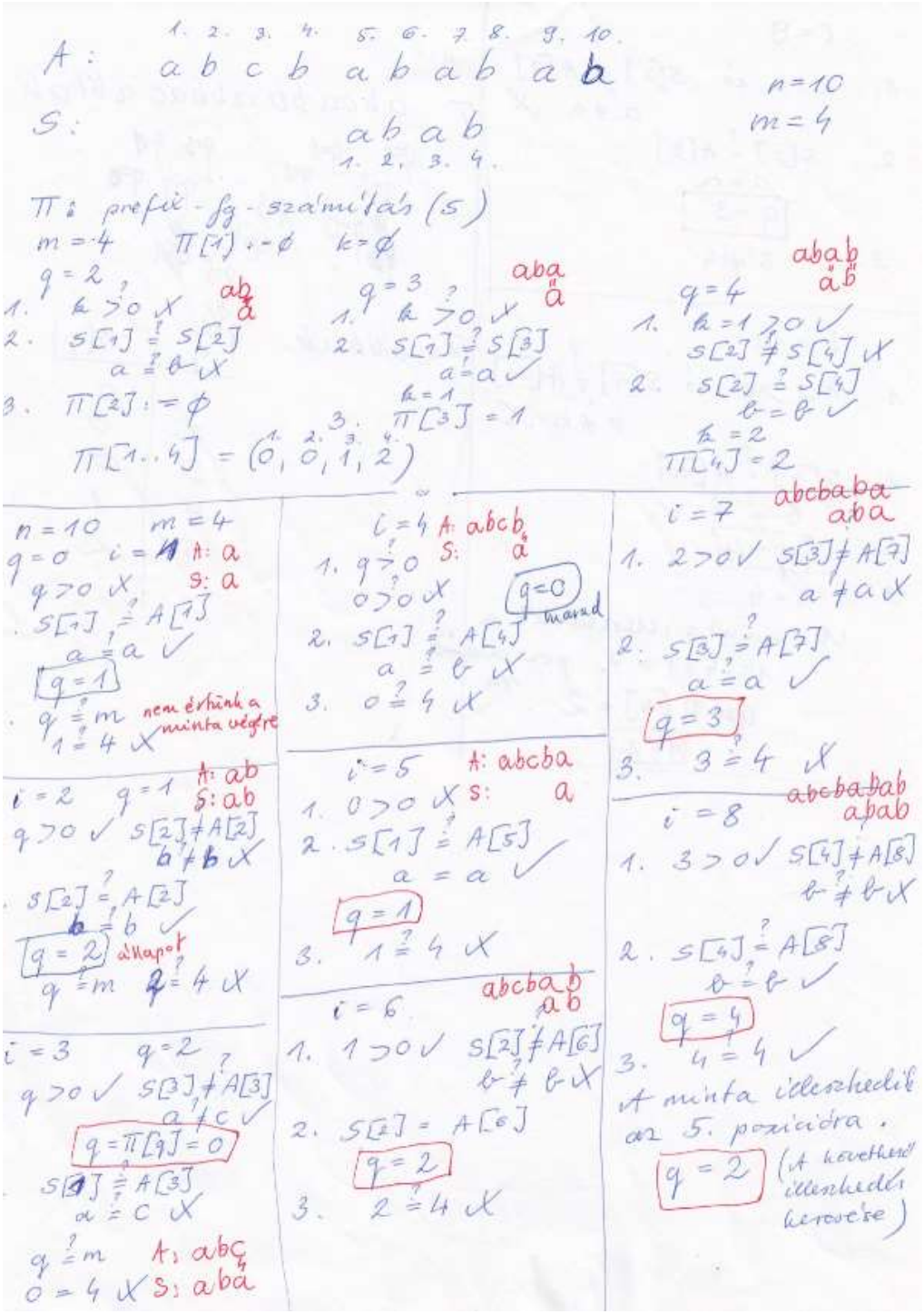
1. $n := hossz[A]$
2. $m := hossz[S]$
3. $\pi := PREFIX - FUGGVENY - SZAMITAS(S)$
4. $q := 0$
5. **for** $i := 1$ **to** n
6. **do while** $q > 0$ **and** $S[q+1] \neq A[i]$
7. **do** $q := \pi[q]$
8. **if** $S[q+1] = A[i]$
9. **then** $q := q + 1$
10. **if** $q = m$
11. **then** print "A minta illeszkedik az ("i-m+1")-edik pozícióra"
12. $q := \pi[q]$

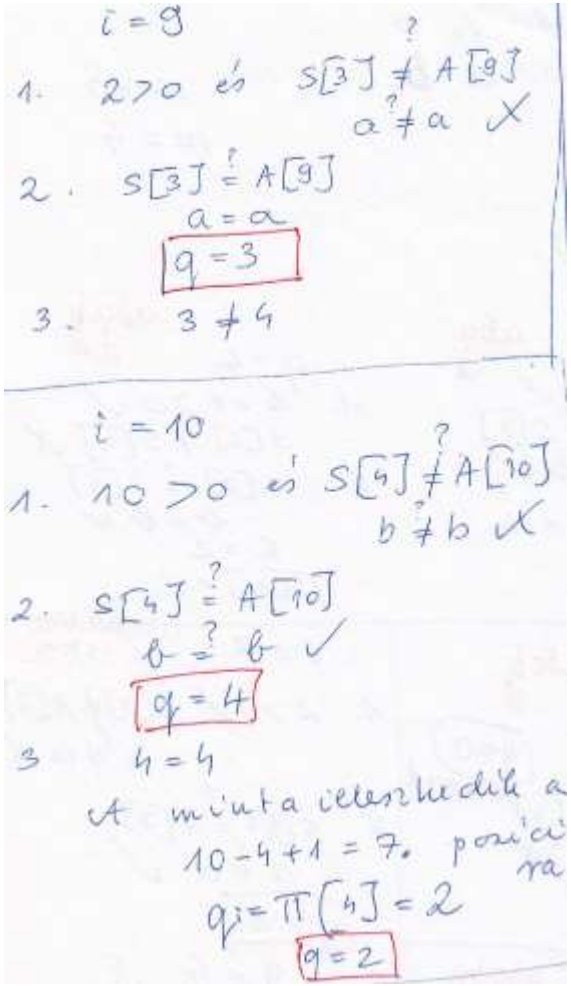
Eljárás

PREFIX-FUGGVENY-SZAMITAS(S)

```
1.  m := hossz[S]
2.  π[1] := 0
3.  k := 0
4.  for q := 2 to m
5.      do while k > 0 and S[k + 1] ≠ S[q]
6.          do k := π[k]
7.      if S[k + 1] = S[q]
8.          then k := k + 1
9.      π[q] := k
10. return π
```

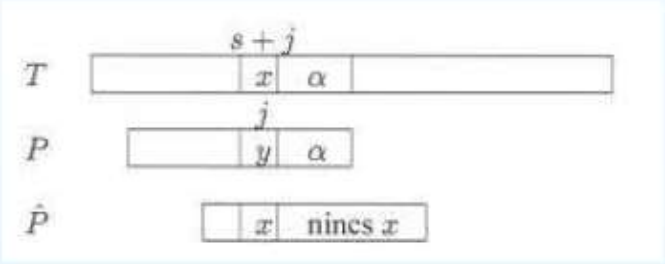
Algoritmus végigkövetése



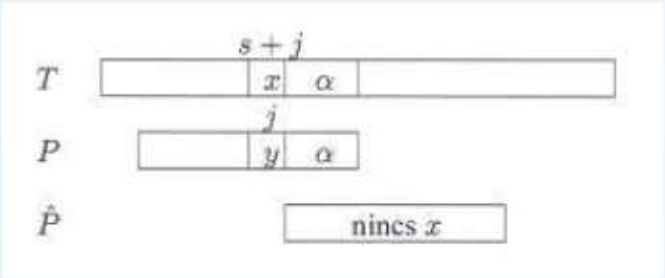


Boyer-Moore algoritmus

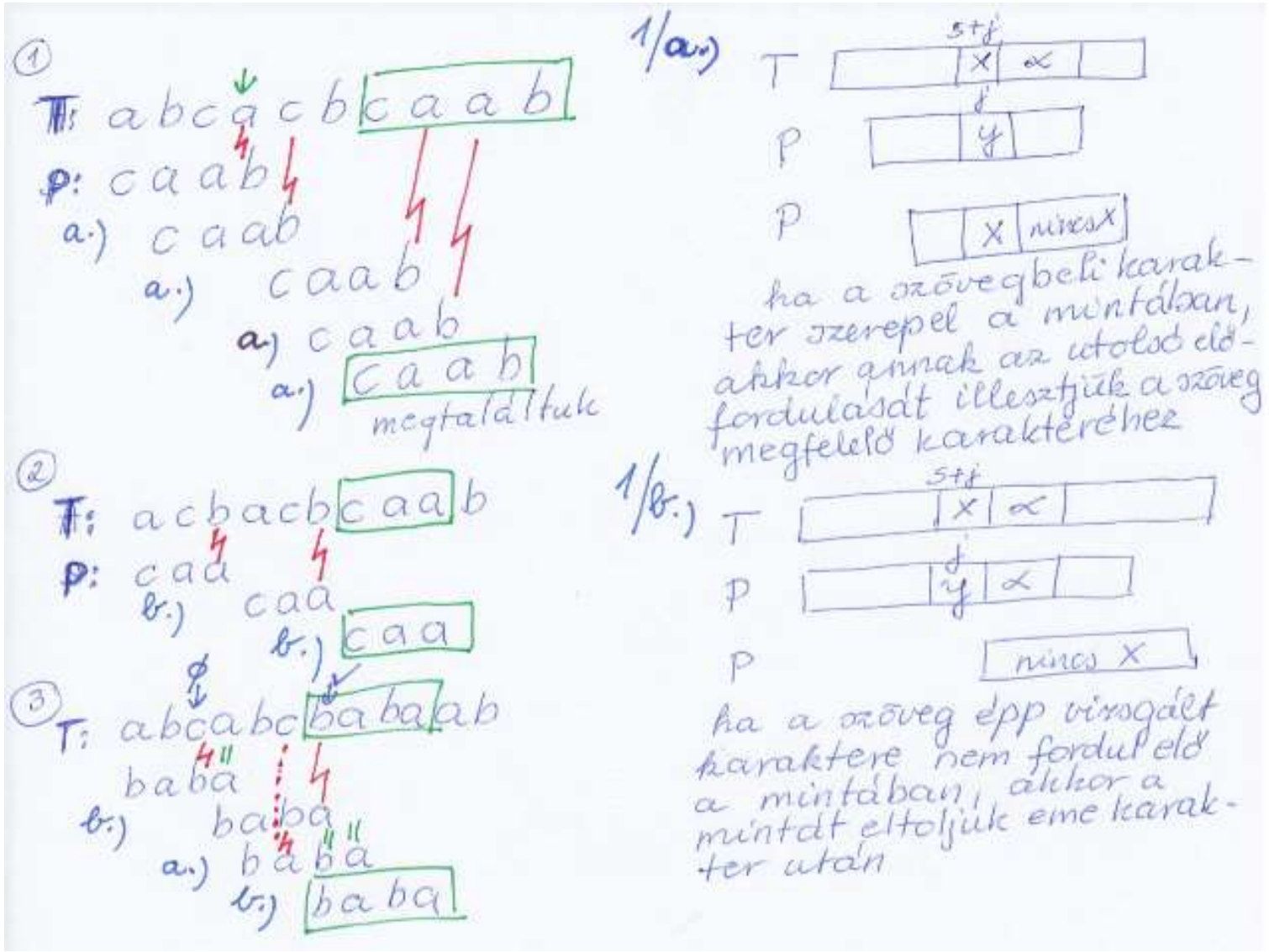
A mintát balról jobbra mozgatja, de **jobbról balra ellenőrzi az illeszkedést**.
Heurisztikák:
1. Az *"utolsó karakter" heurisztika* minden egyes betűhöz (karakterhez) megadja, hogy az hol fordult elő utoljára a mintában. Ha a szöveg épp vizsgált karaktere nem fordul elő a mintában, akkor a mintát eltolhatjuk eme karakter után. Ha viszont szerepel benne a karakter, akkor annak az utolsó előfordulását illesztjük a szöveg megfelelő karakteréhez.
Utolsó pozíció heurisztika, amikor a nem egyező karakter szerepel a mintában:



Utolsó pozíció heurisztika, amikor a nem egyező karakter nem szerepel a mintában:

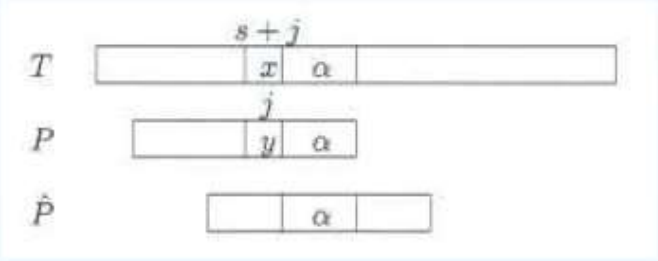


Példák

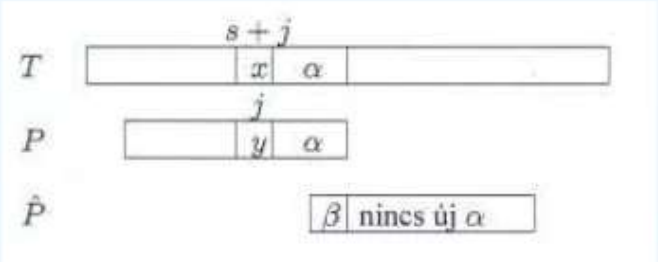

1/a.)
T
P
P
s+j
i
j
x
α
y
x
nincs x
ha a szövegbeli karakter szerepel a mintában, akkor annak az utolsó előfordulását illesztjük a szöveg megfelelő karakteréhez
1/b.)
T
P
P
s+j
i
j
x
α
y
α
nincs x
ha a szöveg épp vizsgált karaktere nem forduló elő a mintában, akkor a mintát eltoljuk eme karakter után
1
T: abcacbcaab
P: caab
a.) caab
a.) caab
a.) caab
a.) caab
megtaláltuk
2
T: acbacbcaab
P: caa
b.) caa
b.) caa
3
T: abcabcbabab
baba
b.) baba
a.) baba
b.) baba

Boyer-Moore algoritmus

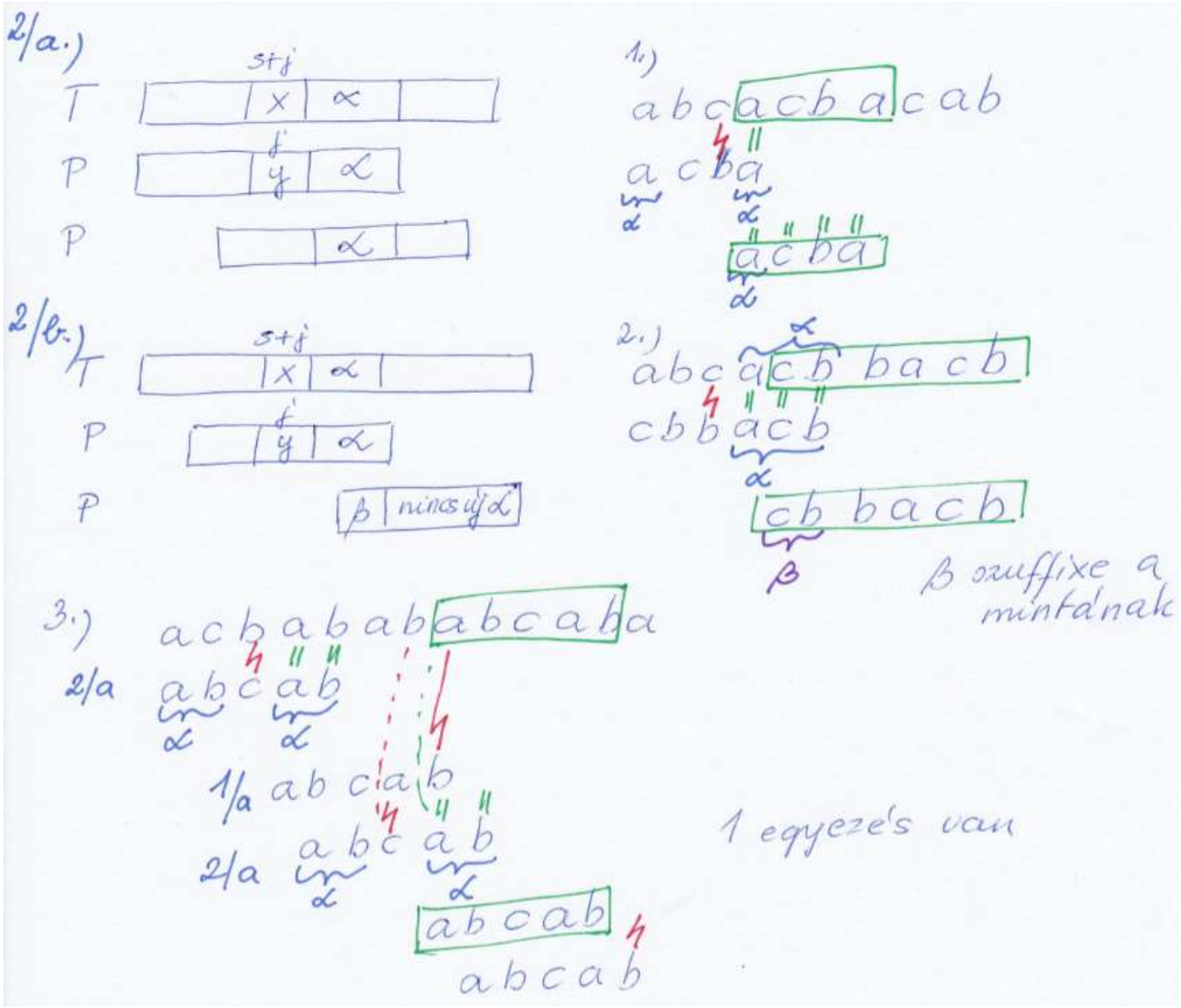
2. A *"jó szuffix" heurisztikánál* úgy mozgatjuk a mintát, hogy a szövegnek arra a részére, amelyre már illeszkedett a minta vizsgált része, újra (legalább részben) illeszkedjék. Ha az illeszkedő részminta újra előfordul a szövegben, akkor ezt a részt kell illeszteni, egyébként eme minta egy szuffixét kell illeszteni a szöveghez.
A jó szuffix heurisztika, amikor az α újra előfordul:



A jó szuffix heurisztika, amikor az α nem fordul elő újra:



Példák


2/a.)
s+j
T
x
α
P
j
y
α
P
α
2/b.)
s+j
T
x
α
P
j
y
α
P
β
nincs új α
1.)
a b c a c b a c a b
a c b a
α
α
a c b a
α
2.)
α
a b c a c b b a c b
c b b a c b
α
c b b a c b
β
β suffixe a mintának
3.)
a c b a b a b a b c a b a
2/a
a b c a b
α
α
1/a a b c a b
2/a a b c a b
α
a b c a b
a b c a b
1 egyezés van

Függvény1

Function UTOLSÓ-POZÍCIÓ(P,m,S)
Input: P minta, m minta hossza, S ábécé
Output: u utolsó pozíció heurisztika értékei

1. **forall the** $a \in S$ **do** $u[a] \leftarrow 0$
2. **for** $j \leftarrow 1$ **to** m **do** $u[P[j]] \leftarrow j$
3. **return** u

Függvény2

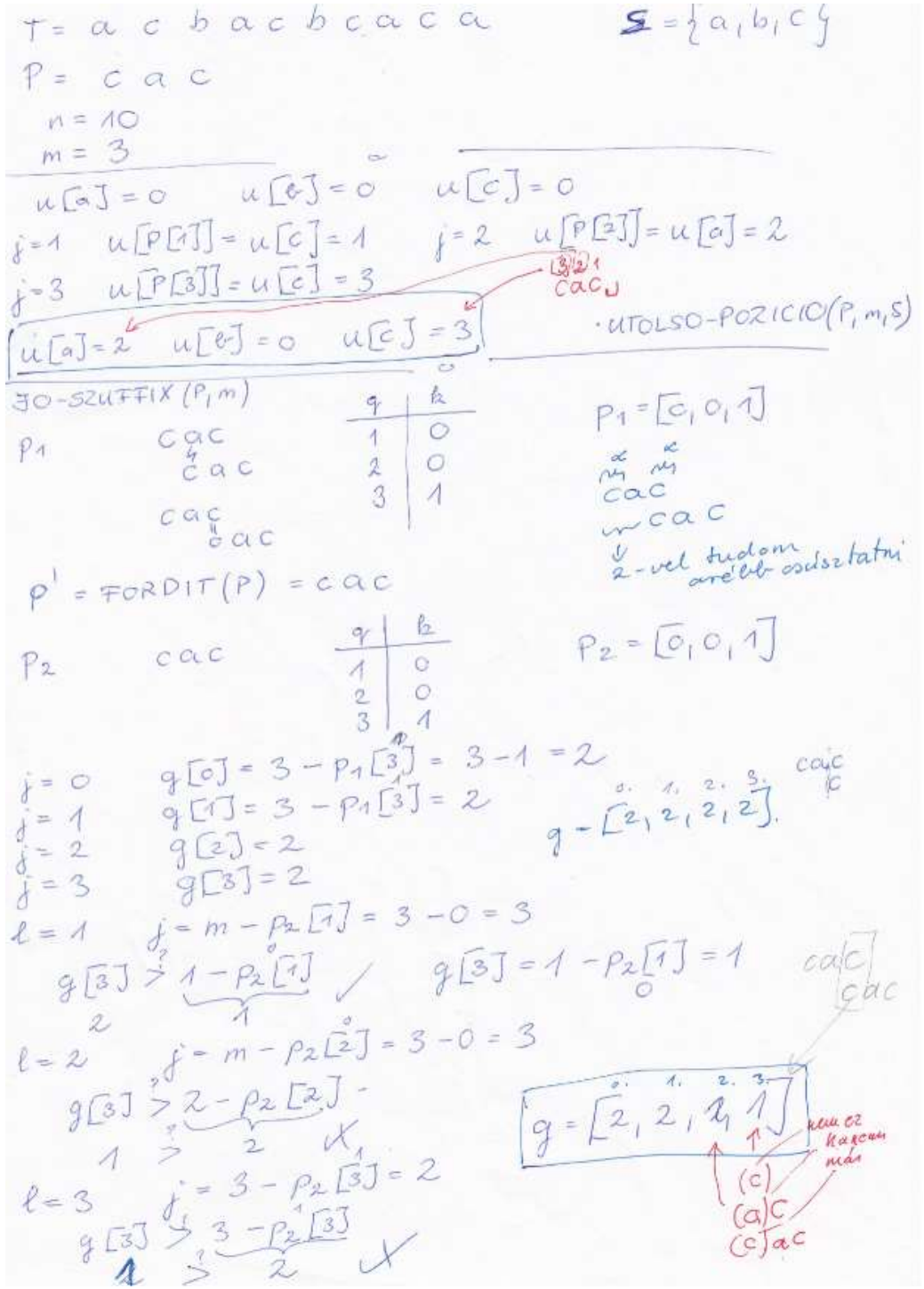
Function JÓ-SZUFFIX(P,m)
Input: P minta, m minta hossza
Output: g jó szuffix heurisztika értékei

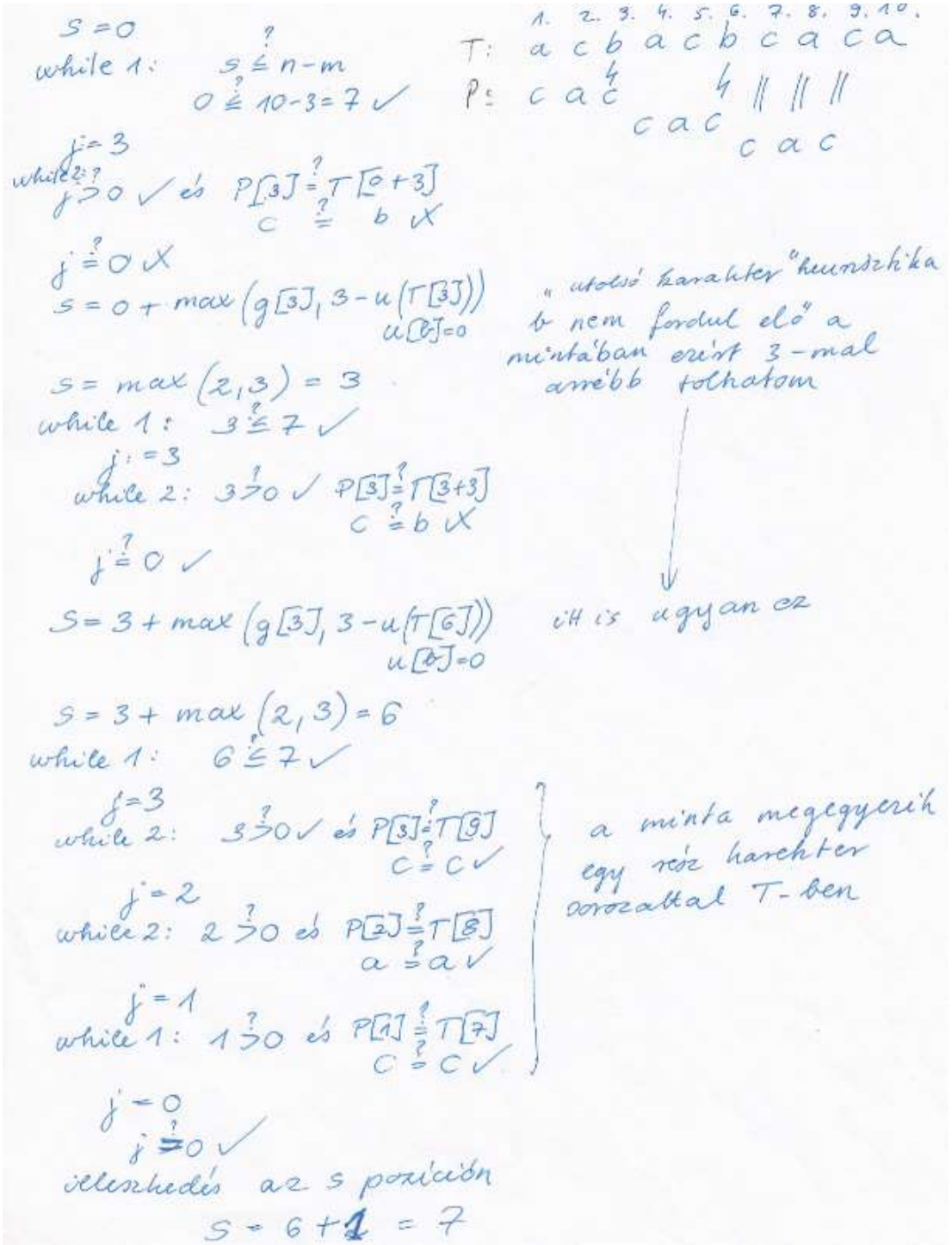
1. $p_1 \leftarrow$ PREFIX-FÜGGVÉNY-SZÁMÍTÁS(P)
2. $P' \leftarrow$ FORDÍT(P)
3. $p_2 \leftarrow$ PREFIX-FÜGGVÉNY-SZÁMÍTÁS(P')
4. **for** $j \leftarrow 0$ **to** m **do** $g[j] \leftarrow m - p_1[m]$
5. **for** $l \leftarrow 1$ **to** m **do**
6. $j \leftarrow m - p_2[l]$
7. **if** $g[j] > l - p_2[l]$ **then** $g[j] \leftarrow l - p_2[l]$
8. **endfor**
9. **return** g

Függvény3

BOYER-MOORE-ILLESZTŐ(T,P,S)
Input: T szöveg, P minta, S ábécé
Output: illeszkedés esetén üzenet

1. $n \leftarrow T.hossz$
2. $m \leftarrow P.hossz$
3. $u \leftarrow$ UTOLSÓ-POZÍCIÓ(P, m, S)
4. $g \leftarrow$ JÓ-SZUFFIX(P, m)
5. $s \leftarrow 0$
6. **while** $s \leq n - m$ **do**
7. $\quad j \leftarrow m$
8. $\quad$ **while** $j > 0$ **and** $P[j] = T[s + j]$ **do** $j \leftarrow j - 1$
9. $\quad$ **if** $j = 0$ **then** ***kiír*** illeszkedés az $s + 1$ pozíción
10. $\quad$ **else** $s \leftarrow s + max(g[j], j - u(T[s + j]))$
11. $\quad$ **endif**
12. **endwhile**


T = a c b a c b c a c a
Σ = {a, b, c}
P = c a c
n = 10
m = 3
u[a] = 0 u[b] = 0 u[c] = 0
j = 1 u[P[1]] = u[c] = 1
j = 2 u[P[2]] = u[a] = 2
j = 3 u[P[3]] = u[c] = 3
u[a] = 2 u[b] = 0 u[c] = 3
UTOLSO-POZICIO(P, m, Σ)
JO-SZUFFIX(P, m)
P1 = [c, 0, 1]
P' = FORDIT(P) = cac
P2 = [0, 0, 1]
g = [2, 2, 2, 2]
g[3] = 1 - P2[1] = 1
g = [2, 2, 2, 1]
(c)
(a)c
(c)ac


s = 0
while 1: s ≤? n - m
0 ≤? 10 - 3 = 7 ✓
j := 3
while 2: j >? 0 ✓ és P[3] =? T[0+3]
c =? b ✗
j =? 0 ✗
s = 0 + max(g[3], 3 - u(T[3]))
u[b] = 0
s = max(2,3) = 3
while 1: 3 ≤? 7 ✓
j := 3
while 2: 3 >? 0 ✓ P[3] =? T[3+3]
c =? b ✗
j =? 0 ✓
s = 3 + max(g[3], 3 - u(T[6]))
u[b] = 0
s = 3 + max(2,3) = 6
while 1: 6 ≤? 7 ✓
j = 3
while 2: 3 >? 0 ✓ és P[3] =? T[9]
c = c ✓
j = 2
while 2: 2 >? 0 és P[2] =? T[8]
a = a ✓
j = 1
while 1: 1 >? 0 és P[1] =? T[7]
c = c ✓
j = 0
j ≥? 0 ✓
illeszkedés az s pozíción
s = 6 + 1 = 7
T: a c b a c b c a c a
1. 2. 3. 4. 5. 6. 7. 8. 9. 10.
P: c a c
c a c
c a c
„utolsó karakter" heurisztika
b nem fordul elő a mintában ezért 3-mal arrébb tolhatom
ill is ugyan ez
a minta megegyezik egy rész karakter sorozattal T-ben