6. előadás
A sztring
A sztring adatszerkezet, sztringkereső algoritmusok

Adatszerkezetek és algoritmusok előadás
2011. március 16.

Kósa Márk és Pánovics János
Debreceni Egyetem
Informatikai Kar

A sztring adatszerkezet

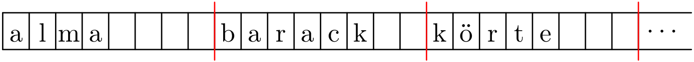
A sztring olyan szekvenciális lista, amelynek az elemei egy ábécé szimbólumai. Ezeket a szimbólumokat karaktereknek nevezzük.

Sztringgel végezhető műveletek

- Létrehozás: explicit módon felsoroljuk a sztring összes karakterét.
- Bővítés: bárhol bővíthető. Bővítéskor két részsztringet képzünk, majd konkatenáljuk azokat a beszúrandó sztringgel.
- Törlés: megvalósítható a fizikai törlés, melynek során két részsztringet képzünk (melyekben már nem szerepel a törlendő részsztring), majd konkatenáljuk azokat.
- Csere: cserélhetünk egy karaktert, de részsztring is cserélhető másik részsztringre. Két részsztringet képzünk, majd konkatenáljuk azokat az új értéket képviselő sztringgel (törlés+bővítés).
- Rendezés: nem értelmezett.
- Keresés: értelmezhető, kereshetünk egy karaktert vagy egy részsztringet.
- Elérés: soros vagy közvetlen.
- Bejárás: értelmezhető.

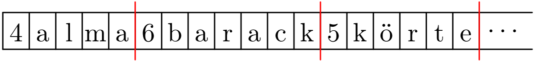
A sztring adatszerkezet folytonos reprezentációi

Fix hosszon:

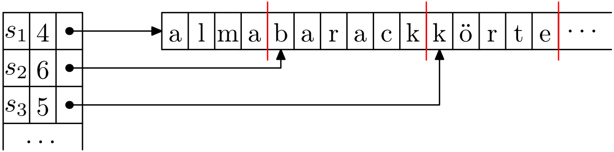


Változó hosszon:

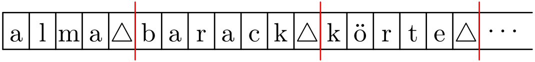
- hossz a sztring előtt:



- információs táblázattal:



- végjellel:



A sztring adatszerkezet szétszórt reprezentációja

A listaelemek tartalmazhatnak egy karaktert vagy egy részsztringet. Utóbbi esetben a részsztringek eltérő hosszúságúak lehetnek, és valamelyik folytonos reprezentációval ábrázoljuk őket.

Sztringkereső algoritmusok

Egy sztringben keresünk egy másik sztringet. Azt a sztringet, *amelyikben* keresünk, alapsztringnek, azt a sztringet pedig, *amit* keresünk, mintasztringnek nevezzük. A pszeudokódokban az alapsztringet A-val, a mintasztringet P-vel fogjuk jelölni.

Néhány sztringkereső algoritmus:

- mezítlábas (brute force) algoritmus
- Knuth–Morris–Pratt-algoritmus
- Boyer–Moore-algoritmus
- Rabin–Karp-algoritmus
- Shift-And (Dömölki Bálint-féle) és Shift-Or algoritmus

A mezítlábas (brute force) algoritmus

```
function BruteForce(A, P)
    n ← hossz(A)
    m ← hossz(P)
    i ← 0
    j ← 0
    while i < n and j < m do
        if A[i + 1] = P[j + 1] then
            i ← i + 1
            j ← j + 1
        else
            i ← i − j + 1
            j ← 0
        end if
    end while
    if j = m then
        return i − m + 1
    else
        return 0
    end if
end function
```


Kósa Márk
Pánovics János

Elméleti alapok

Prefix, szuffix

Legyen Σ egy ábécé és $x = x_1 \dots x_k$ ($k \in \mathbb{N}$) egy k hosszúságú sztring Σ felett! Az x-nek az u részsztring egy prefixe, ha

$$u = x_1 \dots x_b, \text{ahol } 0 \leq b \leq k,$$

azaz ha x u-val kezdődik. Az x-nek az u részsztring egy szuffixe, ha

$$u = x_{k-b+1} \dots x_k, \text{ahol } 0 \leq b \leq k,$$

azaz ha x u-val végződik.

Valódi prefix, valódi szuffix

Az x egy u prefixét vagy x egy u szuffixét valódi prefixnek vagy szuffixnek nevezzük, ha $u \neq x$, azaz ha $b < k$. Ha $b = 0$, akkor $u = \varepsilon$ (üres sztring).

Elméleti alapok

Border

Legyen Σ egy ábécé és $x = x_1 \dots x_k$ ($k \in \mathbb{N}$) egy k hosszúságú sztring Σ felett! Az x-nek az r részsztring egy bordere, ha

$$r = x_1 \dots x_b \text{ és } r = x_{k-b+1} \dots x_k, \text{ahol } 0 \leq b < k.$$

Az x bordere egy olyan részsztring, amely valódi prefixe és valódi szuffixe is x-nek. Ekkor a részsztring b hosszát a border hosszának nevezzük. Ha $b = 0$, akkor $r = \varepsilon$ (üres sztring).

Elméleti alapok

Példa

Legyen $x = abacab$. Az x valódi prefixei:

$$\varepsilon, a, ab, aba, abac, abaca.$$

Az x valódi szuffixei:

$$\varepsilon, b, ab, cab, acab, bacab.$$

Az x borderei:

$$\varepsilon, ab.$$

Az ε border hossza 0, az ab border hossza 2.

Megjegyzés

Az ε üres sztring minden $x \in \Sigma^+$ sztringnek bordere. Az ε üres sztringnek nincs bordere.

Elméleti alapok

Példa

$$
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \dots \\
a & b & c & a & b & c & a & b & d & \\
\color{green}{a} & \color{green}{b} & \color{green}{c} & \color{green}{a} & \color{green}{b} & \color{red}{d} & & & & \\
 & & & a & b & c & a & b & d &
\end{array}
$$

Az 1,. . . ,5 pozíciókon lévő karakterek megegyeznek. A 6. pozíción a *c* és *d* karakterek eltérnek. A minta 3 pozícióval tovább léptethető, és az összehasonlítások a 6. pozíciótól folytathatók.
A léptetés mértékét a *p* egyező prefixének a legszélesebb bordere határozza meg. Ebben a példában az egyező prefix *abcab*, a hossza $j = 5$. Az ő legszélesebb bordere *ab*, amely $b = 2$ hosszúságú. A léptetés mértéke $j - b = 5 - 2 = 3$.

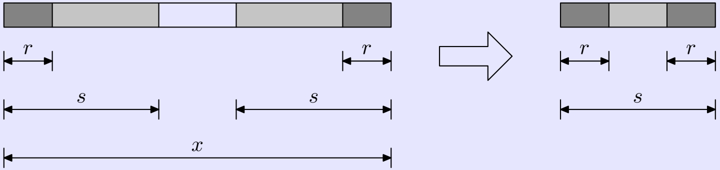
Előfeldolgozás

Az előfeldolgozási szakaszban a minta minden egyes prefixéhez meg kell határozni a legszélesebb border hosszát. Később a keresési szakaszban a léptetés mértéke az egyező prefixeknek megfelelően számítható.

Tétel

Legyen *r* és *s* egy *x* sztring bordere, ahol $|r| < |s|$. Ekkor *r* egy bordere *s*-nek.

Bizonyítás



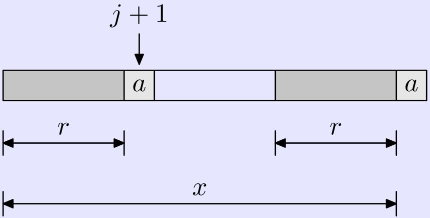
Előfeldolgozás

Megjegyzés

Ha s a legszélesebb bordere x-nek, akkor x következő legszélesebb r borderét megkapjuk s legszélesebb bordereként, és így tovább. . .

Definíció

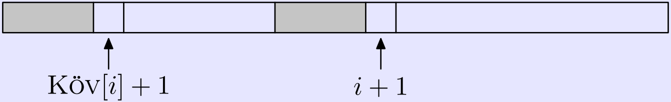
Legyen x egy sztring és $a \in \Sigma$ egy karakter. Az x egy r bordere **bővíthető** a-val, ha ra egy bordere xa-nak.



Előfeldolgozás

A Köv tömb

Az előfeldolgozási szakaszban egy $m + 1$ elemű Köv tömböt számítunk ki. A tömb Köv$[i]$ eleme a mintasztring i hosszúságú prefixéhez tartozó legszélesebb border hossza $(i = 0, \ldots, m)$. Mivel az $i = 0$ hosszúságú ε sztringnek nincsen bordere, ezért Köv$[0] = -1$.



Feltéve, hogy a Köv$[0], \ldots,$ Köv$[i]$ értékeket már ismerjük, a Köv$[i + 1]$ értékét kiszámíthatjuk, ha ellenőrizzük, hogy a $p_1 \ldots p_i$ prefix egy bordere bővíthető-e a p_{i+1} karakterrel. Ez abban az esetben tehető meg, ha $p_{\text{Köv}[i]+1} = p_{i+1}$. A bordereket a Köv$[i]$, Köv$[$Köv$[i]]$, … értékek csökkenő sorrendjében kell megvizsgálni.

Az előfeldolgozó algoritmus

```
function KövFeltölt(P)
    m ← hossz(P)
    i ← 0
    j ← −1
    Köv[0] ← −1
    while i < m do
        if j = −1 or P[i + 1] = P[j + 1] then
            i ← i + 1
            j ← j + 1
            Köv[i] ← j
        else
            j ← Köv[j]
        end if
    end while
    return Köv
end function
```

Előfeldolgozás

Példa

A $p = ababaa$ minta esetén a borderek szélességei a következő értékeket veszik fel a Köv tömbben:

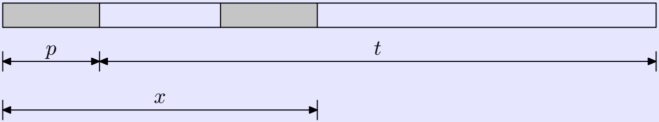
$$\begin{array}{rccccccc} i: & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ p[i]: & & a & b & a & b & a & a \\ \text{Köv}[i]: & -1 & 0 & 0 & 1 & 2 & 3 & 1 \end{array}$$

Láthatjuk például, hogy Köv[5] $= 3$, mivel az 5 hosszúságú *ababa* prefixnek van egy 3 hosszúságú bordere.

Keresés

Megjegyzés

Elméletileg semmi akadálya annak, hogy az előző előfeldolgozó algoritmust a *pt* sztringre alkalmazzuk a *p* helyett. Ha a bordereket csak a *p* minta *m* szélességéig számoljuk ki, akkor a *pt* valamely *x* prefixének egy *m* szélességű bordere megfelel a minta egy előfordulásának *t*-ben (feltéve, hogy a border nem önátfedő).



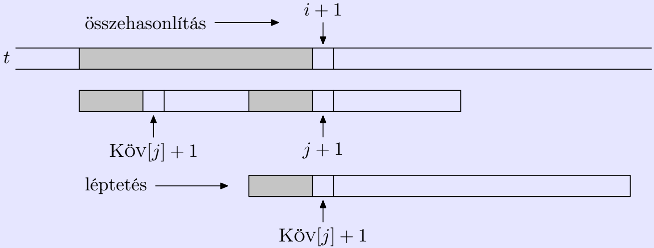
Kósa Márk
Pánovics János

A kereső algoritmus

```
function KMP-KERESÉS(A, P)
    n ← hossz(A)
    m ← hossz(P)
    KÖV ← KÖVFELTÖLT(P)
    i ← 0
    j ← 0
    while i < n and j < m do
        if j = −1 or A[i + 1] = P[j + 1] then
            i ← i + 1
            j ← j + 1
        else
            j ← KÖV[j]
        end if
    end while
    if j = m then
        return i − m + 1
    else
        return 0
    end if
end function
```

Keresés

A minta léptetése



A Shift-And (Dömölki-féle) algoritmus

Az ötlet

Legyen a p mintasztring hossza m! Vegyünk egy m elemű D vektort, amelynek j-edik eleme akkor és csak akkor 1, ha $p_1 \dots p_j$ szuffixe $a_1 \dots a_i$-nek, egyébként 0! (i-vel az alapsztring aktuálisan vizsgált karakterének az indexét jelöljük.)

Megjegyzés

Ha p mérete kisebb, mint egy gépi szó hossza, akkor ez a vektor a számítógép egy regiszterében is tárolható, így gyorsítható a majdani keresés.

Amikor az alapsztring következő, $(i+1)$-edik karakterét olvassuk, meg kell határoznunk egy új D' vektort. Ehhez a következő megfigyelést használjuk fel: a D' vektor $(j+1)$-edik elemének értéke akkor és csak akkor lesz 1, ha

1. a j-edik eleme 1 volt D-nek, azaz $p_1 \dots p_j$ szuffixe volt $a_1 \dots a_i$-nek, és
2. a_{i+1} megegyezik p_{j+1}-gyel.

A Shift-And (Dömölki-féle) algoritmus

A megvalósítás

Az algoritmus az előfeldolgozás során felépít egy B bitmátrixot, amelynek az oszlopait a p mintasztring karaktereivel (illetve ezen karakterek indexeivel), a sorait pedig az ábécé (egymástól különböző) karaktereivel címkézi. Egy c karakterhez tartozó sorban

$$B_{c,j} = \begin{cases} 1, & \text{ha } p_j = c, \\ 0 & \text{egyébként.} \end{cases}$$

A kereséshez az algoritmus három m elemű segédvektort (D, U és V) használ, melyeket a következőképpen definiálunk:

$$\begin{aligned} D_j &= 0 \qquad 1 \leq j \leq m \text{ esetén}, \\ U_j &= \begin{cases} 1, & \text{ha } j = 1, \\ 0 & \text{egyébként,} \end{cases} \\ V_j &= \begin{cases} 1, & \text{ha } j = m, \\ 0 & \text{egyébként.} \end{cases} \end{aligned}$$

A Shift-And (Dömölki-féle) algoritmus

A SHIFT művelet

Legyen $x = (x_1, x_2, \ldots, x_{m-1}, x_m)$! Jelölje $\textsc{Shift}(x)$ azt a vektort, melyre

$$\textsc{Shift}(x) = \textsc{Shift}(x_1, x_2, \ldots, x_{m-1}, x_m) = (0, x_1, x_2, \ldots, x_{m-1})!$$

Az a alapsztringet karakterenként vizsgáljuk végig, és minden a_i karakter érintésekor frissítjük a D vektort a következő formula felhasználásával:

$$D = (\textsc{Shift}(D) \vee U) \wedge B_{a_i}.$$

Ha a keresés során az i-edik karakter feldolgozása után teljesül a

$$D \wedge V \neq 0^m = \underbrace{(0, 0, \ldots, 0)}_{m}$$

feltétel, akkor megtaláltuk a p mintasztring egy előfordulását az a alapsztringben. A minta első karaktere ekkor az alapsztring $(i - m + 1)$-edik karakterére illeszkedik.

Az algoritmus futási ideje $O(n)$, feltéve, hogy a D kiszámításához szükséges műveletek konstans időben elvégezhetők.

A kereső algoritmus

function SHIFT-AND(A, P)
  $n \leftarrow \text{hossz}(A)$
  $m \leftarrow \text{hossz}(P)$
  for all $c \in \Sigma$ **do** ▷ Σ jelöli az ábécé karaktereinek halmazát
    for $j \leftarrow 1$ **to** m **do**
      if $P[j] = c$ **then**
        $B_{c,j} \leftarrow 1$
      else
        $B_{c,j} \leftarrow 0$
      end if
    end for
  end for
  for $j \leftarrow 1$ **to** m **do**
    $D_j \leftarrow U_j \leftarrow V_j \leftarrow 0$
  end for
  $U_1 \leftarrow V_m \leftarrow 1$
  for $i \leftarrow 1$ **to** n **do**
    $D \leftarrow (\text{SHIFT}(D) \vee U) \wedge B_{A[i]}$
    if $D \wedge V \neq 0^m$ **then**
      return $i - m + 1$
    end if
  end for
  return 0
end function

A Shift-And (Dömölki-féle) algoritmus

Legyen $a = atacgatatata$ és $p = atat$! A Shift-And algoritmus által használt B bitmátrix a következő lesz:

	a	t	a	t
a	1	0	1	0
t	0	1	0	1
$*$	0	0	0	0

A mátrixban $*$ jelzi az ábécé a-tól és t-től különböző összes többi karakterét.

A Shift-And (Dömölki-féle) algoritmus

i	a_i	D	$\textsc{Sh}(D)$	$\textsc{Sh}(D) \vee U$	B_{a_i}	D'
1	a	$(0,0,0,0)$	$(0,0,0,0)$	$(1,0,0,0)$	$(1,0,1,0)$	$(1,0,0,0)$
2	t	$(1,0,0,0)$	$(0,1,0,0)$	$(1,1,0,0)$	$(0,1,0,1)$	$(0,1,0,0)$
3	a	$(0,1,0,0)$	$(0,0,1,0)$	$(1,0,1,0)$	$(1,0,1,0)$	$(1,0,1,0)$
4	c	$(1,0,1,0)$	$(0,1,0,1)$	$(1,1,0,1)$	$(0,0,0,0)$	$(0,0,0,0)$
5	g	$(0,0,0,0)$	$(0,0,0,0)$	$(1,0,0,0)$	$(0,0,0,0)$	$(0,0,0,0)$
6	a	$(0,0,0,0)$	$(0,0,0,0)$	$(1,0,0,0)$	$(1,0,1,0)$	$(1,0,0,0)$
7	t	$(1,0,0,0)$	$(0,1,0,0)$	$(1,1,0,0)$	$(0,1,0,1)$	$(0,1,0,0)$
8	a	$(0,1,0,0)$	$(0,0,1,0)$	$(1,0,1,0)$	$(1,0,1,0)$	$(1,0,1,0)$
9	t	$(1,0,1,0)$	$(0,1,0,1)$	$(1,1,0,1)$	$(0,1,0,1)$	$\boxed{(0,1,0,1)}$
10	a	$(0,1,0,1)$	$(0,0,1,0)$	$(1,0,1,0)$	$(1,0,1,0)$	$(1,0,1,0)$
11	t	$(1,0,1,0)$	$(0,1,0,1)$	$(1,1,0,1)$	$(0,1,0,1)$	$\boxed{(0,1,0,1)}$
12	a	$(0,1,0,1)$	$(0,0,1,0)$	$(1,0,1,0)$	$(1,0,1,0)$	$(1,0,1,0)$

A táblázatból látható, hogy az algoritmus kétszer, a 9. és a 11. karakter feldolgozása után állapíthatja meg, hogy $D' \wedge V \neq 0^m$. A mintasztring így a $9-4+1=6$. és $11-4+1=8$. pozícióktól kezdve fordul elő az alapsztringben.

A Shift-And (Dömölki-féle) algoritmus

Az algoritmus hatalmas előnye, hogy párhuzamosan több mintát is kereshetünk a segítségével. Hogyan?

- A B bitmátrixnak annyi oszlopa lesz, ahány karakterből állnak a mintasztringek összesen. Kitöltése ugyanúgy történik, mint eddig.
- Az U bitvektor is annyi elemű lesz, ahány karakterből állnak a mintasztringek összesen. A vektorban az egyes minták első karakterének megfelelő pozíciókba 1-et írunk, a többibe 0-t.
- Az V bitvektor is annyi elemű lesz, ahány karakterből állnak a mintasztringek összesen. A vektorban az egyes minták utolsó karakterének megfelelő pozíciókba 1-et írunk, a többibe 0-t.
- Maga az algoritmus alapvetően nem változik. Ha a $D \wedge V \neq 0^m$ feltétel teljesül, akkor az(oka)t a mintá(ka)t találtuk meg, amely(ek)nek az utolsó karakteréhez tartozó bitpozícióban 1 szerepel a $D \wedge V$ vektorban.

A Shift-And (Dömölki-féle) algoritmus

Legyen $a = atacgatatata$, $p_1 = atat$, $p_2 = gat$ és $p_3 = tata$! Az algoritmus által használt B bitmátrix a következő lesz:

	a	t	a	t	g	a	t	t	a	t	a
a	1	0	1	0	0	1	0	0	1	0	1
g	0	0	0	0	1	0	0	0	0	0	0
t	0	1	0	1	0	0	1	1	0	1	0
$*$	0	0	0	0	0	0	0	0	0	0	0

A mátrixban $*$ jelzi az ábécé a-tól, g-től és t-től különböző összes többi karakterét.

Az U, V és D vektorok pedig így fognak kinézni:

$$U = (1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0)$$
$$V = (0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1)$$
$$D = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

A Shift-Or algoritmus

Az ötlet

Ha a Shift-And algoritmus által használt minden bitet negálunk, és felcseréljük egymással az algoritmusban a konjunkció ($\wedge$) és a diszjunkció ($\vee$) műveleteket, akkor ugyanazt az algoritmust kapjuk. Mivel azonban a SHIFT művelet mindig egy 0 bitet léptet be balról a D vektorban, az eltolás után elhagyható az U vektorral végzendő konjunkciós művelet (mivel az úgyis csak az első bitet nullázná ki), és ezért maga az U vektor is.

Megjegyzés

Ha párhuzamosan több mintát szeretnénk keresni, akkor nem hagyható el sem az U vektor, sem a vele végzendő konjunkciós művelet. Ebben az esetben a Shift-Or algoritmus egyenértékű a Shift-And algoritmussal.

A Shift-Or algoritmus

A megvalósítás

Az előfeldolgozás során felépített B bitmátrix egy c karakterhez tartozó sorában

$$B_{c,j} = \begin{cases} 0, & \text{ha } p_j = c, \\ 1 & \text{egyébként.} \end{cases}$$

A kereséshez az algoritmus most csak két m elemű segédvektort (D és V) használ, melyeket a következőképpen definiálunk:

$$\begin{aligned} D_j &= 1 \qquad 1 \leq j \leq m \text{ esetén,} \\ V_j &= \begin{cases} 0, & \text{ha } j = m, \\ 1 & \text{egyébként.} \end{cases} \end{aligned}$$

A Shift-Or algoritmus

A SHIFT művelet

Legyen $x = (x_1, x_2, \ldots, x_{m-1}, x_m)$! Jelölje $\textsc{Shift}(x)$ azt a vektort, melyre

$$\textsc{Shift}(x) = \textsc{Shift}(x_1, x_2, \ldots, x_{m-1}, x_m) = (0, x_1, x_2, \ldots, x_{m-1})!$$

Az a alapsztringet karakterenként vizsgáljuk végig, és minden a_i karakter érintésekor frissítjük a D vektort a következő formula felhasználásával:

$$D = \textsc{Shift}(D) \vee B_{a_i}.$$

Ha a keresés során az i-edik karakter feldolgozása után teljesül a

$$D \vee V \neq 1^m = (\underbrace{1, 1, \ldots, 1}_{m})$$

feltétel, akkor megtaláltuk a p mintasztring egy előfordulását az a alapsztringben. A minta első karaktere ekkor az alapsztring $(i - m + 1)$-edik karakterére illeszkedik.

A kereső algoritmus

function SHIFT-OR(A, P)
  $n \leftarrow \text{hossz}(A)$
  $m \leftarrow \text{hossz}(P)$
  for all $c \in \Sigma$ **do** ▷ Σ jelöli az ábécé karaktereinek halmazát
    for $j \leftarrow 1$ **to** m **do**
      if $P[j] = c$ **then**
        $B_{c,j} \leftarrow 0$
      else
        $B_{c,j} \leftarrow 1$
      end if
    end for
  end for
  for $j \leftarrow 1$ **to** m **do**
    $D_j \leftarrow V_j \leftarrow 1$
  end for
  $V_m \leftarrow 0$
  for $i \leftarrow 1$ **to** n **do**
    $D \leftarrow \text{SHIFT}(D) \vee B_{A[i]}$
    if $D \vee V \neq 1^m$ **then**
      return $i - m + 1$
    end if
  end for
  return 0
end function

A Shift-Or algoritmus

Legyen $a = atacgatatata$ és $p = atat$! A Shift-Or algoritmus által használt B bitmátrix a következő lesz:

	a	t	a	t
a	0	1	0	1
t	1	0	1	0
$*$	1	1	1	1

A mátrixban $*$ jelzi az ábécé a-tól és t-től különböző összes többi karakterét.

A Shift-Or algoritmus

i	a_i	D	$\textsc{Sh}(D)$	B_{a_i}	D'
1	a	$(1,1,1,1)$	$(0,1,1,1)$	$(0,1,0,1)$	$(0,1,1,1)$
2	t	$(0,1,1,1)$	$(0,0,1,1)$	$(1,0,1,0)$	$(1,0,1,1)$
3	a	$(1,0,1,1)$	$(0,1,0,1)$	$(0,1,0,1)$	$(0,1,0,1)$
4	c	$(0,1,0,1)$	$(0,0,1,0)$	$(1,1,1,1)$	$(1,1,1,1)$
5	g	$(1,1,1,1)$	$(0,1,1,1)$	$(1,1,1,1)$	$(1,1,1,1)$
6	a	$(1,1,1,1)$	$(0,1,1,1)$	$(0,1,0,1)$	$(0,1,1,1)$
7	t	$(0,1,1,1)$	$(0,0,1,1)$	$(1,0,1,0)$	$(1,0,1,1)$
8	a	$(1,0,1,1)$	$(0,1,0,1)$	$(0,1,0,1)$	$(0,1,0,1)$
9	t	$(0,1,0,1)$	$(0,0,1,0)$	$(1,0,1,0)$	$\boxed{(1,0,1,0)}$
10	a	$(1,0,1,0)$	$(0,1,0,1)$	$(0,1,0,1)$	$(0,1,0,1)$
11	t	$(0,1,0,1)$	$(0,0,1,0)$	$(1,0,1,0)$	$\boxed{(1,0,1,0)}$
12	a	$(1,0,1,0)$	$(0,1,0,1)$	$(0,1,0,1)$	$(0,1,0,1)$

A táblázatból látható, hogy az algoritmus kétszer, a 9. és a 11. karakter feldolgozása után állapíthatja meg, hogy $D' \vee V \neq 1^m$. A mintasztring így a $9 - 4 + 1 = 6$. és $11 - 4 + 1 = 8$. pozícióktól kezdve fordul elő az alapsztringben.