Algoritmusok és adatszerkezetek I. 11. Előadás

A mintaillesztési feladat. Egyszerű mintaillesztő algoritmus.

Tartalom

- Mintaillesztés (String-matching)
- Egyszerű mintaillesztés (Naive string-matching)
- Quicksearch és KMP algoritmusok
- Quicksearch: initShift
- Quicksearch algoritmus
- KMP algoritmus
- π prefix függvény
- Ellenőrző kérdések

Mintaillesztés (String-matching)

1. Jelölések. (Notations)

- $\mathbb{N} = \{ i \in \mathbb{Z} \mid i \geq 0 \}$ $i\,..\,k = \{ j \in \mathbb{N} \mid i \leq j \leq k \}$
- $[i\,..\,k) = \{ j \in \mathbb{N} \mid i \leq j < k \}$ $(i\,..\,k) = \{ j \in \mathbb{N} \mid i < j < k \}$
- $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_d\}$ az ábécé (alphabet), ahol $d \in \mathbb{N} \land d > 0$
- $T : \Sigma[n]$ a szöveg (text), ami betűk egy sztringje 0-tól $n{-}1$-ig indexelve.
- $T[i\,..\,j) = T[i\,..\,j{-}1]$
- $P : \Sigma[m]$ a minta (pattern), ahol $0 < m \leq n$.

- A továbbiakban feltesszük, hogy
  - $T : \Sigma[n]$ és $P : \Sigma[m]$,
  - a hosszuk, azaz m és n változatlanok, ahol $1 \leq m \leq n$ (a minta nem üres, és legfeljebb olyan hosszú, mint a szöveg)
- Feladat:
  - A $T : \Sigma[n]$ szövegben keressük a $P : \Sigma[m]$ minta előfordulásait (occurrences)
  - Más szavakkal a T szövegben P minta azon eltolásait keressük, amelyekre $T[s\,..\,s{+}m) = P[0\,..\,m)$
  - Ezekre az eltolásokra $s \in 0\,..\,n{-}m$ nyilvánvalóan teljesül.

Mintaillesztés (String-matching)

2. Definíció. $s \in 0 \, .. \, n{-}m$ a *P minta* lehetséges eltolása (possible shift) a *T szövegen*

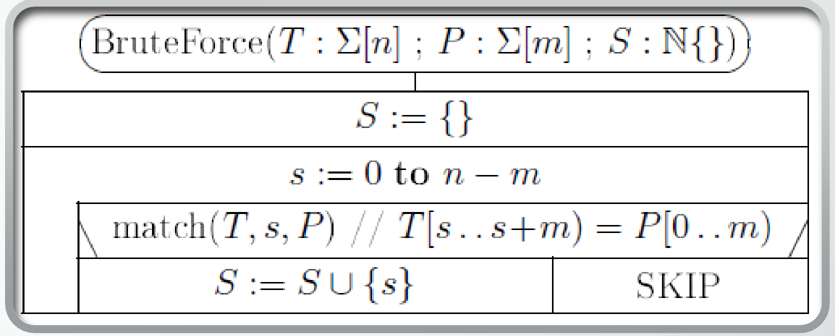
- $s \in 0 \, .. \, n{-}m$ **érvényes eltolás** (valid shift), ha
  - $T[s \, .. \, s{+}m) = P[0 \, .. \, m)$
- Különben $s \in 0 \, .. \, n{-}m$ **érvénytelen eltolás** (invalid shift)

3. Probléma. (Mintaillesztés.)

- Az érvényes eltolások V halmazát szeretnénk meghatározni
- Ehhez eltolások egy S halmazába gyűjtjük a mintaillesztő keresések futása során talált érvényes eltolásokat
- A problémát megoldó algoritmusok közös utófeltétele $S = V$, ahol
  - $V = \{ \, s \in 0 \, .. \, n{-}m \mid T[s \, .. \, s{+}m) = P[0 \, .. \, m) \, \}$

Egyszerű mintaillesztés (Naive string-matching)

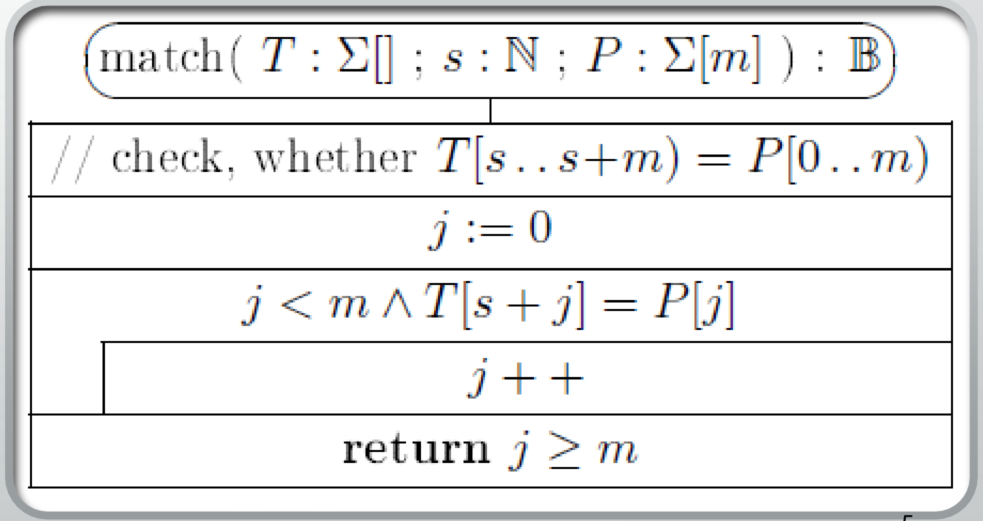
- **Példa:** A $P[0\,..\,4)$ = BABA mintát keressük a $T[0\,..\,11)$ = ABABBABABAB szövegben
  - $\underline{B}$: B-t sikeresen illesztette a szöveg megfelelő betűjére
  - $\cancel{B}$: sikertelenül illesztette.)



- Brute-force (nyers erő) algoritmus:
  - Sorban megvizsgáljuk a lehetséges eltolásokat, és kiszűrjük közülük az érvényeseket

$i =$	0	1	2	3	4	5	6	7	8	9	10
$T[i]=$	A	B	A	B	B	A	B	A	B	A	B
	$\cancel{B}$	A	B	A							
		$\underline{B}$	$\underline{A}$	$\underline{B}$	$\cancel{A}$						
			$\cancel{B}$	A	B	A					
				$\underline{B}$	$\cancel{A}$	B	A				
$s{=}4$					$\underline{B}$	$\underline{A}$	$\underline{B}$	$\underline{A}$			
						$\cancel{B}$	A	B	A		
$s{=}6$							$\underline{B}$	$\underline{A}$	$\underline{B}$	$\underline{A}$	
								$\cancel{B}$	A	B	A

$S = \{4; 6\} = V$



Algoritmus műveletigénye

- A $T[s\,..\,s+m) = P[0\,..\,m)$ egyenlőségvizsgálatra:
  - $MT_e(m) \in \Theta(m)$
  - $mT_e(m) \in \Theta(1)$
- ➢ Innét a teljes algoritmusra:
  - $MT_{BF}(n, m) \in \Theta((n - m + 1) * m)$
  - $mT_{BF}(n, m) \in \Theta(n - m + 1)$
- Ha egy feladatosztályon valamely $0 < k < 1$ konstansra $m \leq k * n$
  - ➢ $1 * n \geq n - m + 1 > n - k * n = (1 - k) * n$,
    ahol $1 > 1 - k > 0$ konstans
- ➢ A Θ(▪) függvényosztályok definíciója szerint:
  - $n - m + 1 \in \Theta(n)$
  - $(n - m + 1) * m \in \Theta(n * m)$

$\mathrm{match}(\ T : \Sigma[]\ ;\ s : \mathbb{N}\ ;\ P : \Sigma[m]\ ) : \mathbb{B}$

// check, whether $T[s\,..\,s+m) = P[0\,..\,m)$
$j := 0$
$j < m \wedge T[s+j] = P[j]$
$\quad j{+}{+}$
return $j \geq m$

$\mathrm{BruteForce}(T : \Sigma[n]\ ;\ P : \Sigma[m]\ ;\ S : \mathbb{N}\{\})$

$S := \{\}$	
$s := 0$ **to** $n - m$	
$\mathrm{match}(T, s, P)$ // $T[s\,..\,s+m) = P[0\,..\,m)$	
$S := S \cup \{s\}$	SKIP

Műveletigény

- az $\cdot \in \Theta(\cdot)$ reláció tranzitivitása miatt:

4. Következmény. Ha egy feladatosztályon a k és az m konstansokra $0 < k < 1 \land m \le k * n$

- $mT_{BF}(n,m) \in \Theta(n)$
- $MT_{BF}(n,m) \in \Theta(n * m)$

- Ha egy feladatosztályon m az n-hez képest nem is elhanyagolható
  - azaz valamely $\varepsilon > 0$ konstansra $m \ge \varepsilon * n \Rightarrow \varepsilon * n * n \le n * m \le n * n$
- Következésképp $n * m \in \Theta(n^2)$
  - Ebből + 4 következménnyel adódik az alábbi eredmény:

5. Következmény.
Ha egy feladatosztályon az ε és a k konstansokra $0 < \varepsilon \le k < 1 \land \varepsilon * n \le m \le k * n \Rightarrow MT_{BF}(n,m) \in \Theta(n^2)$

Quicksearch és KMP algoritmusok

- A gyorsabb keresés érdekében:
  - általában egynél nagyobb lépésekben növeljük a P[0 . .*m*) minta eltolását a T[0 . . *n*) szöveghez képest úgy
    - biztosan ne ugorjunk át egyetlen érvényes eltolást sem
- A tényleges mintaillesztés előtt egy előkészítő fázist hajt végre
  - nem függ a szövegtől, csak a mintától

Quicksearch és KMP algoritmusok

- Quicksearch előkészítő fázisában
  - az ábécé σ elemeihez shift(σ) $\in 1\,..\,m+1$ címkéket társítunk, ahol $P[0\,..\,m)$ a keresett minta
  - Tfh $\sigma = T[s + m]$
  - shift(σ) érték: megmondja, hogy a $T[s\,..\,s + m) = P[0\,..\,m)$ öh után legalább mennyivel kell (jobbra) eltolni a P mintát a szövegen ahhoz
    - a $T[s + m]$ alapján legyen esély a mintának a megfelelő szövegrészhez való illeszkedésére
  - $\sigma \in P[0\,..\,m)$ esetén a shift(σ) $\in 1\,..\,m$ érték azt mondja meg, hogy legalább mennyivel kell tovább tolni a P mintát ahhoz, hogy a $T[s + m]$ betűhöz kerülő karaktere maga is σ legyen. Világos, hogy a σ legjobboldali P-beli előfordulásához tartozik a legkisebb ilyen eltolás.
  - $\sigma \notin P[0\,..\,m)$ esetén shift(σ) $= m + 1$ lesz, azaz a minta átugorja a $T[s + m]$ karaktert.

Quicksearch: initShift

- shift(σ) jelentése
  - $\sigma \in P[0\,..m)$ esetén: $\text{shift}(\sigma) \in 1\,..m$
    - legalább mennyivel kell tovább tolni a P mintát ahhoz, hogy a $T[s + m]$ betűhöz kerülő karaktere maga is σ legyen
    - a σ legjobboldali P-beli előfordulásához tartozik a legkisebb ilyen eltolás
  - $\sigma \notin P[0\,..m)$ esetén: $\text{shift}(\sigma) = m + 1$
    - azaz a minta átugorja a $T[s + m]$ karaktert
- Az ábécé méretét konstansnak tekintve: $T_{\text{initShift}}(m) \in \Theta(m)$
- A $P[0\,..\,4)$=CADA mintával az initShift() működése:

$\text{initShift}(P : \Sigma[m])$

$\forall \sigma \in \Sigma$

  $shift(\sigma) := m + 1$

$j := 0 \text{ to } m-1$

  $shift(P[j]) := m - j$

σ	A	B	C	D
initial $shift(\sigma)$	5	5	5	5
C			4	
A	3			
D				2
A	1			
final $shift(\sigma)$	1	5	4	2

Quicksearch algoritmus

σ	A	B	C	D
$shift(\sigma)$	1	5	4	2

$i =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$T[i]=$	A	D	A	B	A	B	C	A	D	A	B	C	A	B	A	D	A	C	A	D	A	D	A
	$\cancel{C}$	A	D	A																			
		$\cancel{C}$	A	D	A																		
$s = 6$							$\underline{C}$	$\underline{A}$	$\underline{D}$	$\underline{A}$													
												$\underline{C}$	$\underline{A}$	$\cancel{D}$	A								
														$\cancel{C}$	A	D	A						
$s = 17$																		$\underline{C}$	$\underline{A}$	$\underline{D}$	$\underline{A}$		
																				$\cancel{C}$	A	D	A

$S = \{6; 17\} = V$

- $mT(n,m) \in \Theta\left(\frac{n}{m+1} + m\right)$
  - (pl. ha $T[0\,..\,n)$ és $P[0\,..\,m)$ diszjunktak)
- $MT(n,m) \in \Theta((n - m + 2) * m)$
  - (pl. ha $T = AA...A$ és $P = A...A$)

- Összehasonlítás az egyszerű mintaillesztéssel
  - $mT(n)$ nagyságrenddel jobb
  - $MT(n)$ egy kicsit rosszabb
  - Tapasztaltok szerint az átlagos futási idő inkább a legjobb esethez áll közel
- Időkritikus alkalmazásokban
  - Egy stabilabb futási idejű, minden esetben hatékony algoritmusra lenne szükségünk

QuickSearch($T : \Sigma[n]$; $P : \Sigma[m]$; $S : \mathbb{N}\{\}$)

initShift(P) ; $S := \{\}$; $s := 0$

$s + m \leq n$

match(T, s, P) // $T[s\,..\,s+m) = P[0\,..\,m)$

$S := S \cup \{s\}$	SKIP

$s + m < n$

$s += shift(T[s + m])$	break

Mintaillesztés lineáris időben (Knuth-Morris-Pratt, azaz KMP algoritmus)

- A KMP algoritmus futási ideje : $\Theta(n)$ (minden esetben)
  - a legjobb és a legrosszabb eset között körülbelül kétszeres szorzóval
  - ➢ hatékony működés és nagyon stabil futási idő
- Sohasem lép vissza a *T* szövegen
  - Ellentétben az előző megoldásokkal
  - ➢ a KMP algoritmus könnyen implementálható szekvenciális fájlokra.

Speciális jelölések:

6. Jelölések . *Ha x és y két sztring =>*

- *$x + y$ a konkatenáltjuk.*
  - *Pl. $x = AB$ és $y = BA$ esetén $x + y = ABBA$ és $y + x = BAAB$.*
- *ε az üres sztring.*
  - *Nyilván tetszőleges x sztringre $x + \varepsilon = x = \varepsilon + x$.*
- *$x \sqsubseteq y$ (x az y prefixe) jelentése: $\exists z$ sztring : $x + z = y$.*
  - *(ε, A,AB,ABB,ABBA $\sqsubseteq$ ABBA)*
- *$x \sqsubset y$ (x az y valódi prefixe [proper prefix]) jelentése: $x \sqsubseteq y \wedge x \neq y$.*
  - *(ε, A,AB,ABB $\sqsubset$ ABBA)*

Speciális jelölések:

6. Jelölések. *Ha x és y két sztring* $\Rightarrow$

- $x \sqsupseteq y$ *(x az y szuffixe) jelentése:* $\exists z$ *sztring :* $z + x = y$.
  - *(ABBA,BBA,BA, A, ε* $\sqsupseteq$ *ABBA)*
- $x \sqsupset y$ *(x az y valódi szuffixe [proper suffix]) jelentése:* $x \sqsupseteq y \wedge x \neq y$.
  - *(BBA,BA, A, ε* $\sqsupset$ *ABBA)*
- $P_{:j} = P[0\,..\,j) = P[0\,..\,j-1]$ *(csak ebben az alfejezetben)*
  - $P_{:j}$ *a P sztring j hosszúságú prefixe, azaz kezdőszelete.*
  - $P_{:0} = \varepsilon$ *az üres prefixe.*
  - ≈ $T_{:i} = T[0\,..\,i)$.
  - ➢ *minden sztringnek szuffixe az üres sztring, azaz* $P_{:0} \sqsupseteq P_{:j}$ $(j \in 0\,..\,m)$,
  - ➢ *minden nemüres sztringnek valódi szuffixe az üres sztring, azaz* $P_{:0} \sqsupset P_{:j}$ $(j \in 1\,..\,m)$.]

Sztringek tulajdonságai

7. **Lemma.** A szuffixum reláció tranzitivitása (Transitivity of the suffix relation)
   - $x \sqsupset y \land y \sqsupseteq z \Rightarrow x \sqsupset z$.
8. **Lemma.** Átfedő szuffix (Overlapping-suffix) lemma
   - Tegyük fel, hogy x, y és z olyan sztringek, amelyekre $x \sqsupseteq z$ és $y \sqsupseteq z$
   - $|x| \leq |y| \Rightarrow x \sqsupseteq y$
   - $|x| < |y| \Rightarrow x \sqsupset y$
   - $|x| = |y| \Rightarrow x = y$
9. **Lemma.** Szuffix kiterjesztési (Suffix-extension) lemma
   - $P_j \sqsupseteq T_i \land P[j] = T[i] \Leftrightarrow P_{:j+1} \sqsupseteq T_{i+1}$
   - $P_{:i} \sqsupset P_{:j} \land P[i] = P[j] \Leftrightarrow P_{:i+1} \sqsupset P_{:j+1}$

Bevezetés a KMP algoritmushoz

- Példa:
  - Feltesszük, hogy van egy hosszabb *T* szövegünk, de most ennek csak a *T*[*i*−5 . . *i*+2] = *BABABABB* részét vesszük figyelembe.
  - A minta a $P_{:6}$ = *BABABB*
  - Az aktuális eltolás *i* – 5
  - A sikeresen illesztett betűket aláhúztuk
  - A sikertelenül illesztett karaktert pedig áthúztuk

$\dots$	T_{i-5}	T_{i-4}	T_{i-3}	T_{i-2}	T_{i-1}	T_i	T_{i+1}	T_{i+2}
$\dots$	B	A	B	A	B	A	B	B
$P_{:6} =$	$\underline{B}$	$\underline{A}$	$\underline{B}$	$\underline{A}$	$\underline{B}$	$\cancel{B}$		
			B	A	B	$\underline{A}$	$\underline{B}$	$\underline{B}$

Bevezetés a KMP algoritmushoz

…	T_{i-5}	T_{i-4}	T_{i-3}	T_{i-2}	T_{i-1}	T_i	T_{i+1}	T_{i+2}
…	B	A	B	A	B	A	B	B
$P_{:6} =$	$\underline{B}$	$\underline{A}$	$\underline{B}$	$\underline{A}$	$\underline{B}$	$\cancel{B}$		
			B	A	B	$\underline{A}$	$\underline{B}$	$\underline{B}$

- Példa megoldása:
  - $P_{:5} = T[i-5\,.\,.\,i)$, de $P[6] \neq T[i]$
    - ➢ $i-5$ egy érvénytelen eltolás (3. sor)
  - Most keressük a legkisebb további eltolást, hogy $P_{:k} = T[i-k\,.\,.\,i)$ teljesüljön (4. sor)
    - Azaz a minta $P_{:5}$ kezdőszeletének az a $P_{:k}$ (0 ≤ k < 5) prefixe, ami még a szöveg $T[i-k\,.\,.\,i)$ része alatt van, illeszkedjen arra
  - Ez a $P_{:k}$ -t meghatározó legkisebb további eltolás legfeljebb 5
    - mert $P_{:0} \sqsupset T_{:i.}$
    - Ezzel a $P_{:k}$-t meghatározó eltolással biztosan nem ugrunk át egyetlen érvényes eltolást sem.
  - A példában két betűvel kell tovább tolni a mintát és $k = 3$
    - $P[3] = T[\mathrm{i}]$, $P[4] = [i+1]$ és $P[5] = T[i+2]$
    - ➢ $i-3$ egy érvényes eltolás
    - Bármely ennél nagyobb eltolás átugorta volna az $i-3$ érvényes eltolást.

Bevezetés a KMP algoritmushoz

- Hogyan lehetne a *k* értékét hatékonyan meghatározni általános esetben?
  - ≈ az előző gondolatmenethez
  - Általánosítás:
    - Minta: $P_{:m}$
    - Szöveg: $T_{:n}$
    - $j \in 1..m$, ahol
      - az aktuális eltolás : $i-j$
      - $P_{:j} = T[i-j..i)$ azaz $P_{:j} \sqsupseteq T_{:i}$
      - $(P[j] \neq T[i] \vee j = m)$
  - A *további eltolás* + $k = j$
    - ➢ Nagyobb *további eltolás*hoz kisebb *k* érték, míg kisebb további eltoláshoz nagyobb *k* érték tartozik

Bevezetés a KMP algoritmushoz

- k a legkisebb *további eltolás*hoz tartozik, amelyre $P_{:k} \sqsupset T_{:i}$
  - ahol ez a *további eltolás* $\geq j$
    - ui. $P_{:0} \sqsupset T_{:i}$
  - ➢ k a legnagyobb h, amire $P_{:h} \sqsupset T_{:i} \land 0 \leq h < j$ teljesül
- Továbbá $P_{:h} \sqsupset T_{:i}$ ekvivalens a $P_{:h} \sqsupset P_{:j}$ relációval
  - mert $P_{:j} \sqsupseteq T_{:i} \land 0 \leq h < j$.
  - Más szavakkal: $P[0\,..\,h) = T[i-h\,..\,i) \Leftrightarrow P[0\,..\,h) = P[j-h\,..\,j) \Rightarrow P[0\,..\,j) = T[i-j\,..\,i) \land 0 \leq h < j$
- ➢ k csak $P_{:j}$ -től függ
  - ➢ $P_{:m}$ fix $\Rightarrow$ k csak j-től függ
- Más szavakkal: a leghosszabb $P_{:h}$ -t keressük, amire $P_{:h} \sqsubset P_{:j} \land P_{:h} \sqsupset P_{:j}$
  - Ennek hosszát a π prefix függvény határozza meg.

π prefix függvény

11. Definíció. $\pi(j) = \max\{\, h \in 0\,..\,j{-}1 \mid P_{:h} \sqsupset P_{:j} \,\}$ $(j \in 1\,..\,m)$

12. Tulajdonság. $j \in 1\,..\,m \Rightarrow \pi(j) \in 0\,..\,j-1$

- **Bizonyítás.**
  - 11. definíció szerint: $\pi(j)$ a $0\,..\,j-1$ egy részhalmazának a maximuma.

13. Lemma. $j \in [1\,..\,m) \Rightarrow \pi(j+1) \le \pi(j) + 1$

- **Bizonyítás.**
  - Ha $\pi(j+1) = 0 \Rightarrow \pi(j+1) = 0 \le 0{+}1 \le \pi(j){+}1$
    - mivel $\pi(j) \ge 0$ (12. tul.)
  - Ha $\pi(j+1) > 0$
    - (11. def. szerint) $P_{:(\pi(j+1)-1)} = P_{:\pi(j+1)} \sqsupset P_{:j+1}$
    - (9. lemma) $P_{:\pi(j+1)-1} \sqsupset P_{:j}$
    - (11. def) $\pi(j+1) - 1 \le \pi(j)$
    - $\pi(j+1) \le \pi(j) + 1$.

Példa: π függvény kiszámítása

$P[j-1] =$	B	A	B	A	B	B	A	B
$j =$	1	2	3	4	5	6	7	8
$\pi(j) =$	0	0	1	2	3	1	2	3

- $P_{:8} = P[0\,..\,8) =$ BABABBAB mintára
  - Alkalmazzuk:

    11. Definíció. $\pi(j) = \max\{\, h \in 0\,..\,j{-}1 \mid P_{:h} \sqsupset P_{:j} \,\}$ $(j \in 1\,..\,m)$

    12. Tulajdonság. $j \in 1\,..\,m \Rightarrow \pi(j) \in 0\,..\,j-1$

    13. Lemma. $j \in [1\,..\,m) \Rightarrow \pi(j+1) \le \pi(j) + 1$
  - Megoldás:
    - *j=1 B*
    - *j=2 BA*
    - *j=3 BAB*
    - *j=4 BABA*
    - *j=5 BABAB*
    - *j=6 BABABB*
    - *j=7 BABABBA*
    - *j=8 BABABBAB*

Példa:

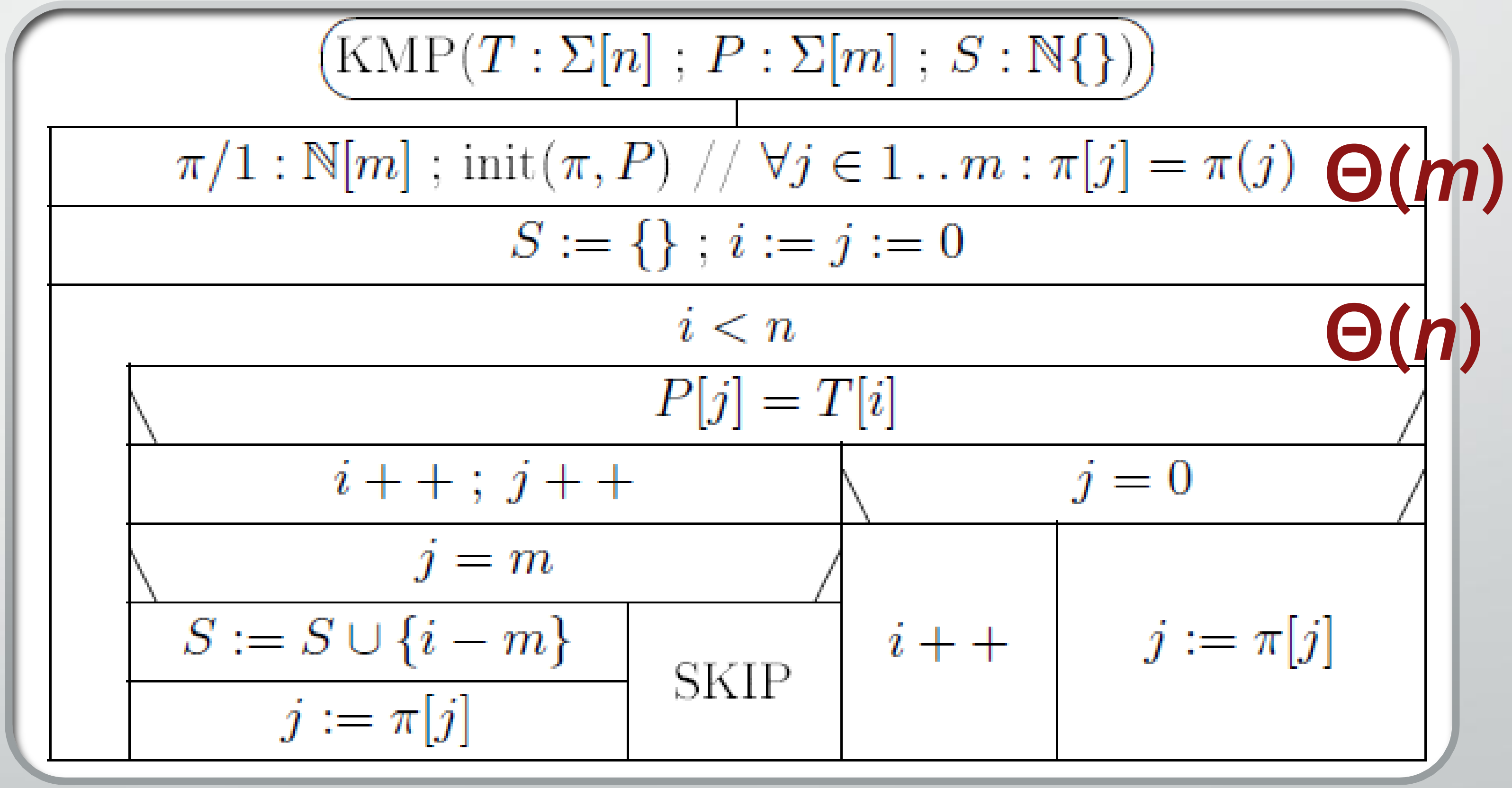
- Minta: $P_{:8}$ = P[0 . . 8) = BABABBAB
- Szöveg: $T_{:18}$ = T[0 . . 18) = ABABABABBABABABBAB

$i =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$T[i]=$	A	B	A	B	A	B	A	B	B	A	B	A	B	A	B	B	A	B
	$\cancel{B}$																	
		$\underline{B}$	$\underline{A}$	$\underline{B}$	$\underline{A}$	$\underline{B}$	$\cancel{B}$											
$s=3$				B	A	B	$\underline{A}$	$\underline{B}$	$\underline{B}$	$\underline{A}$	$\underline{B}$							
									B	A	B	$\underline{A}$	$\underline{B}$	$\cancel{B}$				
$s=10$											B	A	B	$\underline{A}$	$\underline{B}$	$\underline{B}$	$\underline{A}$	$\underline{B}$
																B	A	B

$$S = \{3; 10\} = V$$

A KMP algoritmus fő eljárása

- init(π, P)
  - π prefix-függvény értékeit összegyűjti a π tömbbe
  - Műveletigény: $\Theta(m)$
  - Utófeltétele:
    - $\forall j \in 1\,..\,m : \pi[j] = \pi(j)$)
- Fő eljárás:
  - a ciklusa 17. invariánsán alapszik
  - ahol i–j a P minta aktuális eltolása a T szövegen



Invariáns helyességéhez szükséges lemma

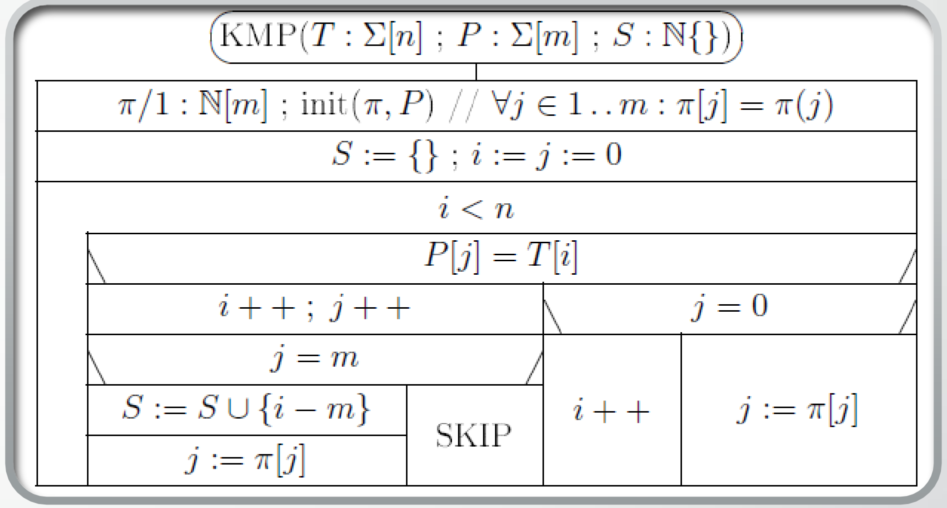
16. Lemma. $j \in 1\,..\,m \wedge P_{:j} \sqsupseteq T_{:i} \Rightarrow$ nincs érvényes eltolás az $(i\text{–}j\,..\,i\text{–}\pi(j))$ intervallumban

- **Bizonyítás.**
  - Indirekt: $k \in (\pi(j)\,..\,j)$ és $i\text{–}k$ érvényes eltolás
  - ➢ $T[i\text{–}k\,..\,i\text{–}k+m = P[0\,..\,m)$
  - Nyilván $k < j \leq m$
  - ➢ $k < m$
  - ➢ $i\text{–}k+m > i$
  - ➢ $T[i\text{–}k\,..\,i) = P[0\,..\,k)$, vagy másképp írva $P_{:k} \sqsupseteq T_{:i}$
  - Továbbá $P_{:j} \sqsupseteq T_{:i} \wedge k < j$
  - ➢ Következésképpen $P_{:k} \sqsupset P_{:j}$ (8. Átfedő szuffix lemma miatt)
  - ➢ $k > \pi(j)$
  - ➢ $P_{:k} \not\sqsupset P_{:j}$
    - ➢ ui. $P_{:\pi(j)}$ a $P_{:j}$ leghosszabb valódi prefixe, ami egyben szuffixe is, a π prefix függvény definíciója (11) alapján.

Invariáns helyessége

17. Tétel. Az (Inv) állítás a KMP(T, P, S) eljárás ciklusának invariánsa.

1. $P_{:j} \sqsupseteq T_{:i} \wedge$
2. $0 \le j \le i \le n \wedge$
3. $j < m \wedge$
4. $S = V \cap [0\,..\,i\text{–}j)$



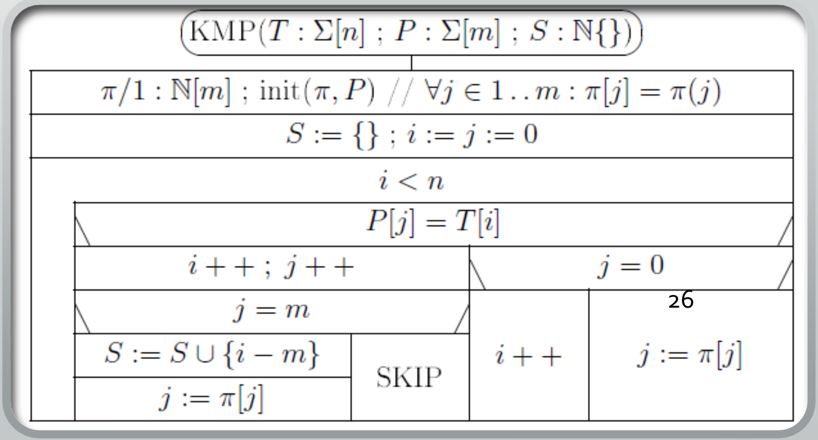
- **Bizonyítás.**
  - Kezdetben:
    - $i = j = 0 \Rightarrow P_{:0} \sqsupseteq T_{:0} \wedge 0 \le 0 \le 0 \le n \wedge 0 < m \wedge S = V \cap [0\,..\,0) = V \cap \{\} = \{\}$
  - A továbbiakban:
    - igazoljuk, hogy a ciklusmag végrehajtása (mind a négy programágon) tartja (Inv)-et.
    - Állításainkhoz mindig hozzáértjük init(π, P) utófeltételét.
    - Ha $i < n$ => belépünk a ciklusba => (Inv1) $P_{:j} \sqsupseteq T_{:i} \wedge 0 \le j \le i < n \wedge j < m \wedge S = V \cap [0\,..\,i\text{–}j)$ teljesül

Invariáns helyességének bizonyítása

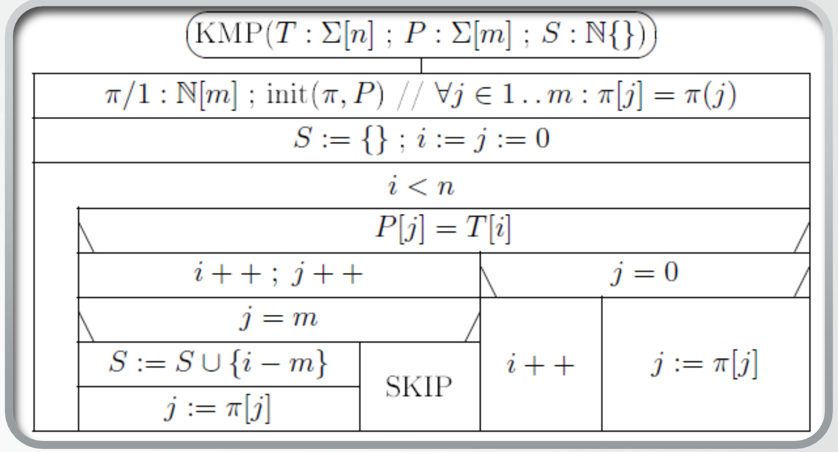
- Ha $P[j] = T[i]$ =>(9. Szufix kiterjesztési lemma szerint) $P_{:j+1} \sqsupseteq T_{:i+1}$
  - majd i és j növelése után (Inv2) $P_{:j} \sqsupseteq T_{:i} \wedge 0 < j \le i \le n \wedge j \le m \wedge S = V \cap [0\,..\,i-j)$

1. Ha $j = m$ => $P_{:m} \sqsupseteq T_{:i}$ (másképpen írva: $P[0\,..\,m) = T[i-m\,..\,i)$)
   - ➢ $i-m$ érvényes eltolás, amit S-hez hozzávéve (Inv3) $P_{:j} \sqsupseteq T_{:i} \wedge 0 < j \le i \le n \wedge j \le m \wedge S = V \cap [0\,..\,i-j]$
   - ➢ $j \in 1\,..\,m \wedge P_{:j} \sqsupseteq T_{:i}$, =>(16. lemma szerint) nincs érvényes eltolás az $(i-j\,..\,i-\pi(j))$ intervallumban
   - ➢ $P_{:j} \sqsupseteq T_{:i} \wedge 0 < j \le i \le n \wedge j \le m \wedge S = V \cap [0\,..\,i-\pi(j))$
   - $P_{:\pi(j)} \sqsupset P_{:j} \sqsupseteq T_{:i}$,
   - (a szufixum reláció tranzitivitása 7. lemma)
   - $\pi[j] = \pi(j)$

     ⇒ $P_{:\pi(j)} \sqsupseteq T_{:i} \wedge 0 < j \le i \le n \wedge j \le m \wedge S = V \cap [0\,..\,i-\pi[j])$
   - ➢ $\pi[j] = \pi(j) \in [0\,..\,j)$ => $j := \pi[j]$ értékadás után már $0 \le j < i \wedge j < m$ teljesül
   - ➢ $P_{:j} \sqsupset T_{:i} \wedge 0 \le j < i \le n \wedge j < m \wedge S = V \cap [0\,..\,i-j)$
   - ➢ az első programág végén közvetlenül következik (Inv).
2. Ha $j \neq m$ => (Inv2 szerint) $j < m$
   - ➢ a második programág végén is teljesül (Inv).



Invariáns helyességének bizonyítása



- Ha P[j]≠T[i]
  - ➢ (9. Szufix kiterjesztési lemma szerint) $P_{:j+1} \not\sqsupseteq T_{:i+1}$ (másképpen írva: $P[0\,..\,j] \neq T[i-j\,..\,i]$
    - ➢ Inv1-ből $j < m$ miatt => $P[0\,..\,m) \neq T[i-j\,..\,i-j+m)$
    - ➢ $i-j$ érvénytelen eltolás
    - \+ Inv1 (azaz a $P_{:j} \sqsupseteq T_{:i} \land 0 \leq j \leq i < n \land j < m \land S = V \cap [0\,..\,i-j)$ tulajdonsággal
    - ➢ (Inv4) $P_{:j} \sqsupseteq T_{:i} \land 0 \leq j \leq i < n \land j < m \land S = V \cap [0\,..\,i-j]$
    3. Ha $j = 0$ => (Inv4 figyelembevételével) $P_{:0} \sqsupseteq T_{:i} \land 0 = j \leq i < n \land j < m \land S = V \cap [0\,..\,i-j]$
       - ➢ i növelése után => $P_{:0} \sqsupseteq T_{:i} \land 0 = j \leq i \leq n \land j < m \land S = V \cap [0\,..\,i-j)$
       - ➢ a harmadik programág végén is teljesül (Inv) $P_{:j} \sqsupseteq T_{:i} \land 0 \leq j \leq i \leq n \land j < m \land S = V \cap [0\,..\,i-j)$
    4. Ha $j \neq 0$ 7> (Inv4 figyelembevételével) $P_{:j} \sqsupseteq T_{:i} \land 0 < j \leq i < n \land j < m \land S = V \cap [0\,..\,i-j]$
       - Ennek közvetlen következménye (Inv3) $P_{:j} \sqsupseteq T_{:i} \land 0 < j \leq i \leq n \land j \leq m \land S = V \cap [0\,..\,i-j]$
       - (lásd 1. pr.ág vizsgálata) (Inv3) esetén a $j := \pi[j]$ értékadást végrehajtva teljesül (Inv)
       - ➢ 4. programág végén is igaz.

A KMP(T, P, S) eljárás parciális helyessége

18. Tétel. Ha a KMP algoritmus megáll, akkor megoldja az 3. (mintaillesztési) problémát,

- azaz $S = V$ teljesül, amikor a KMP(T, P, S) eljárás befejeződik

- **Bizonyítás.**
  - A KMP(T, P, S) eljárás ciklusának (Inv) invariánsa (17)
    - azaz $P_{:j} \sqsupseteq T_{:i} \wedge 0 \le j \le i \le n \wedge j < m \wedge S = V \cap [0\,..\,i{-}j)$
  - a ciklusfeltétel tagadása
    - vagyis $i \ge n$ szerint

$\Rightarrow$ $i = n \wedge j < m \wedge S = V \cap [0\,..\,n{-}j)$ teljesül, mikor a ciklus befejeződik

+ $j < m \Rightarrow n{-}j > n{-}m \Rightarrow [0\,..\,n{-}j) \supseteq 0\,..\,n{-}m \supseteq \{\, s \in 0\,..\,n{-}m \mid T[s\,..\,s+m) = P[0\,..\,m) \,\} = V \Rightarrow [0\,..\,n{-}j) \supseteq V$

➢ $S = V \cap [0\,..\,n{-}j) = V$

➢ $S = V$ teljesül, amikor a KMP(T, P, S) eljárás befejeződik

A KMP(T, P, S) eljárás megállása és hatékonysága

- A terminálás igazolása
  - init(π, P) eljárás műveletigénye: $\Theta(m)$ (bizonyítás később)
  - KMP(T, P, S) eljárás műveletigénye: $\Theta(n)$
- KMP(T, P, S) eljárás ciklusa legfeljebb $2n$ iterációt hajt végre
  - Elég belátni, hogy a fő ciklusának a futási ideje $\Theta(n)$
    - ➢ $m \in 1 \,..\, n$ => az init(π, P) eljárás és egyéb inicializálások $\Theta(m)$ műveletigénye aszimptotikus nagyságrendben már nem módosít
  - A fő ciklus legalább n és legfeljebb $2n$ iterációt hajt végre
    - A ciklus (Inv) invariánsa (17) szerint: $i \in 0 \,..\, n \wedge j \in 0 \,..\, (m-1) \wedge j \leq i$

A KMP(T, P, S) eljárás megállása és hatékonysága

- legalább n iteráció
  - az első iteráció előtt $i = 0$
  - mindegyik iteráció legfeljebb eggyel növeli az i változót
  - a ciklusfeltétel $i < n$

legalább n iteráció szükséges ahhoz, hogy $i = n$ legyen, és az eljárás befejeződjék

- legfeljebb $2n$ iteráció
  - Tekintsük a $2i–j$ kifejezést!
    - Az $i \in 0\,.\,.\,n \wedge j \in 0\,.\,.\,(m–1) \wedge j \leq i$ invariáns tulajdonság => $2i–j \in 0\,.\,.\,2n$
    - Az első iteráció előtt $2i–j = 0$
    - mindegyik iterációval (mind a négy programágon) szigorúan monoton nő
    - végig $2i–j \leq 2n$

legfeljebb $2n$ iteráció után megáll a ciklus

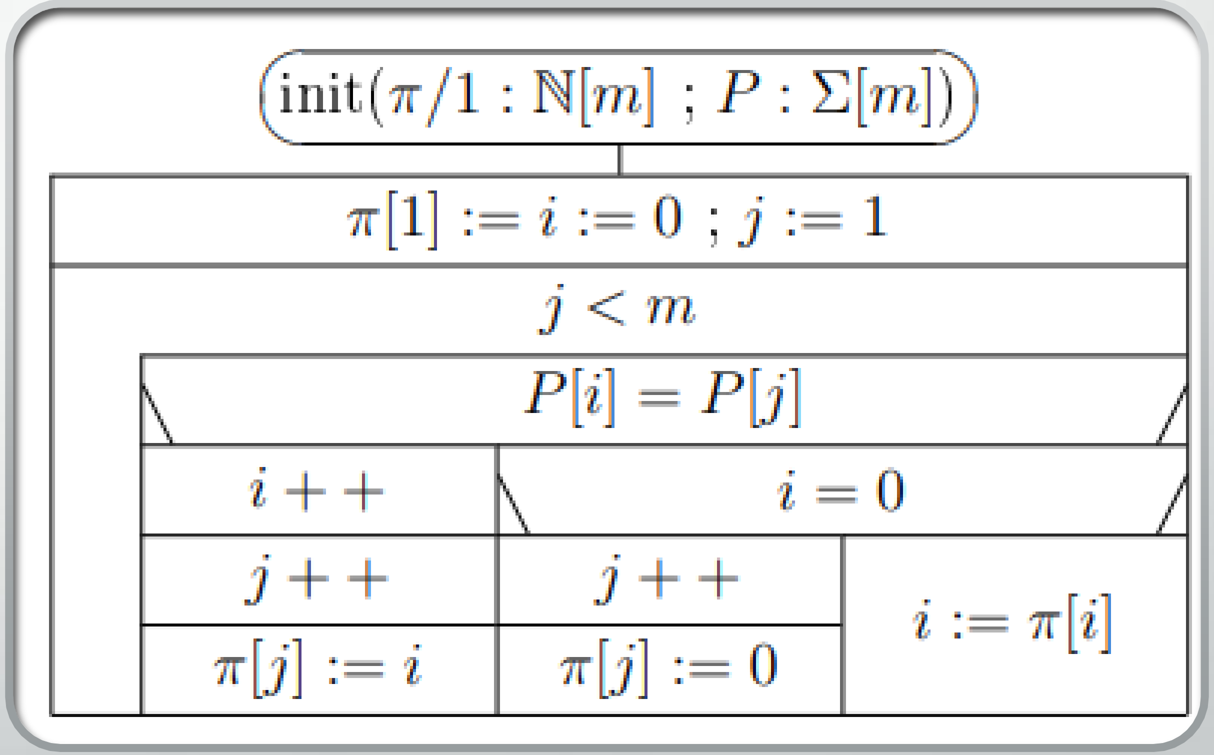
A π prefix tömb inicializálása

19. Lemma. $P_{:i} \sqsupset P_{:j} \wedge 0 < i < j < m \wedge \pi(j+1) \leq i \Rightarrow \pi(j+1) \leq \pi(i)+1$

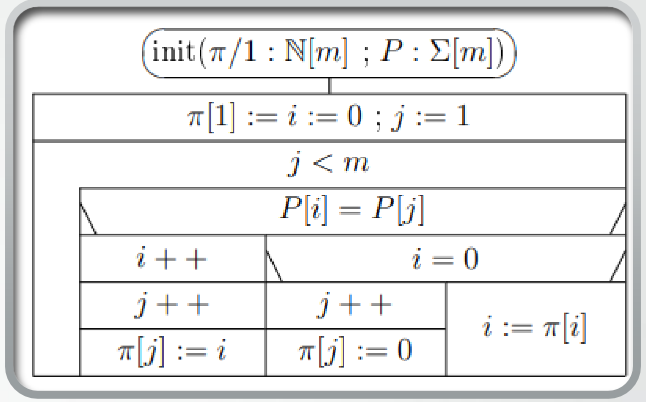
- **Bizonyítás.**
  - Ha $\pi(j+1) = 0 \Rightarrow \pi(j+1) < 0+1 \leq \pi(i)+1$
    - ➢ definíció szerint $\pi(i) \geq 0$
  - Ha $\pi(j + 1) > 0 \Rightarrow k := \pi(j + 1) - 1$
  - ➢ $k \geq 0$ és $k + 1 = \pi(j + 1)$
  - A π függvény definíciója szerint: $P_{:k+1} \sqsupset P_{:j+1} \Rightarrow P_{:k} \sqsupset P_{:j}$
  - \+ $P_{:i} \sqsupset P_{:j}$ és $k < i$
  - ➢ $P_{:k} \sqsupset P_{:i}$
  - ➢ $k \leq \pi(i)$
  - ➢ $\pi(j + 1) = k + 1 \leq \pi(i) + 1$ adódik

Az init(π, P) eljárás

- A ciklus invariánsa
  - $P_{:i} \sqsupset P_{:j} \wedge$
  - $[\ \forall k \in 1\,..\,j : \pi[k] = \pi(k)\ ] \wedge$
  - $0 \leq i < j \leq m \wedge$
  - $(\ j < m \longrightarrow \pi(j+1) \leq i + 1\ )$



Az init(π, P) eljárás ciklusinvariánsa

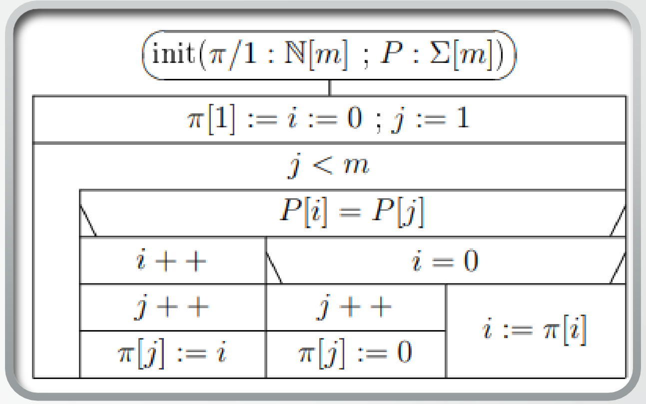


20. Tétel. Az (inv) állítás az init(π, P) eljárás ciklusának invariánsa.

- (inv) $P_{:i} \sqsupset P_{:j} \wedge (\forall k \in 1..j : \pi[k] = \pi(k)) \wedge 0 \le i < j \le m \wedge (j < m \rightarrow \pi(j+1) \le i+1)$

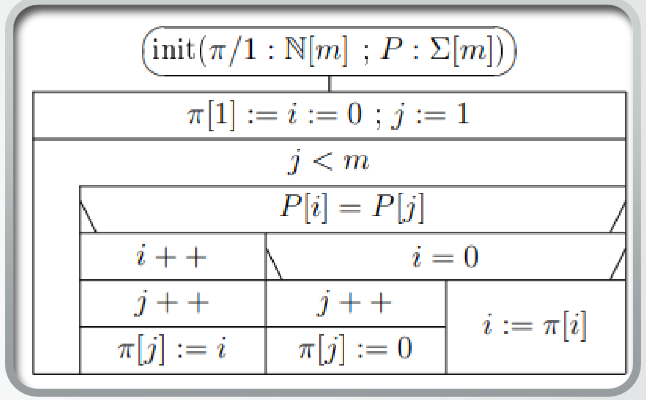
- Bizonyítás.
  - Közvetlenül az 1. ciklusiteráció előtt a $\pi[1] := i := 0 \; ; \; j := 1$ inicializálások miatt (inv):
    - $P_{:0} \sqsupset P_{:1} \wedge (\forall k \in 1..1 : \pi[k] = \pi(k) = \pi(1) = 0) \wedge 0 \le 0 < 1 \le m \wedge (1 < m \rightarrow \pi(2) \le 1)$
    - Igazak, mert
      - $P_{:0}$ üres sztring bármely nem nemüres sztringnek valódi szufixe
      - (12. tulajdonság szerint) $\pi(1) = 0$
      - a P minta m mérete nem nulla
      - (13. lemma szerint) $\pi(1+1) \le \pi(1) + 1 = 1$, feltéve, hogy $m > 1$

Az init(π, P) eljárás ciklusinvariánsa



- Bizonyítás folytatása
  - A ciklusiterációk tartják az (inv) tulajdonságot
    - azaz, ha tetszőleges ciklusiteráció előtt (inv) és a ciklusfeltétel igaz => a ciklusmag bármelyik ágának a végére jutva ugyancsak igaz lesz
    - Amikor belépünk a ciklusmagba => (inv) és a ciklusfeltétel alapján (inv1) teljesül:
      - (inv1) $P_{:i} \sqsupset P_{:j} \wedge (\forall k \in 1..j : \pi[k] = \pi(k)) \wedge 0 \leq i < j < m \wedge \pi(j + 1) \leq i + 1$.

1. Ha $P[i] = P[j]$
   - (9. lemma és inv1-ből a $P_{:i} \sqsupset P_{:j}$ alapján) $P_{:i+1} \sqsupset P_{:j+1}$
   - (a π prefix függvény definíciója (11) szerint) $\pi(j + 1) \geq i + 1$
   - (inv1 szerint) $\pi(j + 1) \leq i + 1$.
   - ➢ $\pi(j + 1) = i + 1$



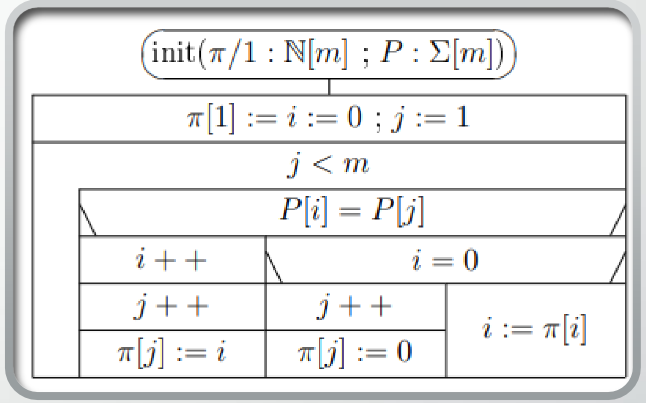
Az init(π, P) eljárás ciklusinvariánsa

- Bizonyítás folytatása
  - Az $i++; j++; \pi[j] := i$ értékadások után pedig:
    - $P_{:i} \sqsupset P_{:j} \wedge (\forall k \in 1..j : \pi[k] = \pi(k)) \wedge 0 < i < j \leq m \wedge \pi(j) = i$
  - Ha j < m => (13. lemma és π(j) = i szerint) π(j + 1) ≤ π(j)+1 = i+1
  - A ciklusmag első ágának végén tehát teljesül az (inv) állítás

2-3. Ha P[i]≠P[j]

- (1.9. lemma szerint) $P_{:i+1} \not\sqsupset P_{:j+1}$
- (a π prefix függvény definíciója (11) szerint) $\pi(j+1) \neq i+1$
- (inv1 alapján) $\pi(j + 1) \leq i + 1$
- $\pi(j + 1) \leq i$
- Ezt (inv1)-gyel összevetve, a belső elágazás előtt (inv2) teljesül
  - (inv2) $P_{:i} \sqsupset P_{:j} \wedge (\forall k \in 1..j : \pi[k] = \pi(k)) \wedge 0 \leq i < j < m \wedge \pi(j + 1) \leq i$.

Az init(π, P) eljárás ciklusinvariánsa



- Bizonyítás folytatása

  2. Ha $i = 0$

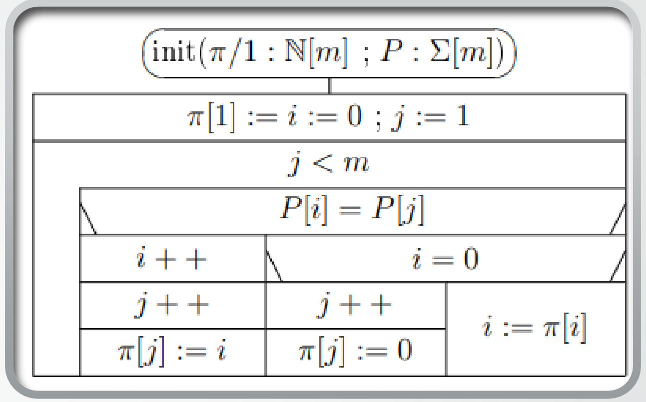
     - (inv2)-ből + (a $\pi(j + 1) \leq i$ állítás) => $\pi(j + 1) = 0$
       - mivel a π függvény nemnegatív

     + (inv2) a j+ +; $\pi[j]$:= 0 értékadások után:
       - $P_{:i} \sqsupset P_{:j} \land (\forall k \in 1..j : \pi[k] = \pi(k)) \land 0 = i < j \leq m \land \pi(j) = i.$

     - Ha $j < m$ => (13. lemma és $\pi(j) = i$ szerint) $\pi(j+1) \leq \pi(j) + 1 = i + 1$

     - A ciklusmag második ágának végén is teljesül az (inv) állítás.

Az init(π, P) eljárás ciklusinvariánsa



- Bizonyítás folytatása
  2. Ha $i \neq 0$
     - (inv2 szerint) inv3 teljesül
       - (inv3) $P_{:i} \sqsupset P_{:j} \wedge (\ \forall k \in 1\,..\,j : \pi[k] = \pi(k)\ ) \wedge 0 < i < j < m \wedge \pi(j+1) \leq i$
     - $i > 0$ => az 1.19. lemma feltételei teljesülnek => $\pi(j{+}1) \leq \pi(i) + 1$
     - (inv3 2. és 3. tényezője figyelembevételével) $\pi[i] = \pi(i)$
     - \+ π prefix függvény definíciója (11) => $P_{:\pi[j]} \sqsupset P_{:i}$
     - \+ (az inv3 $P_{:i} \sqsupset P_{:j}$ állítása + az 7. tranzitivitási lemma => $P_{:\pi[j]} \sqsupset P_{:j}$
     - (inv3) alapján, az $i := \pi[i]$ értékadás után:
       - $P_{:i} \sqsupset P_{:j}\ \wedge\ (\ \forall k \in 1\,..\,j : \pi[k] = \pi(k)\ ) \wedge\ 0 \leq i < j < m\ \wedge\ \pi(j+1) \leq i + 1$
     - A ciklusmag utolsó ágának a végén is teljesül az (inv) állítás

Az init(π, P) eljárás parciális helyessége

21. Következmény. Ha az init(π, P) eljárás megáll, akkor a visszatérésekor teljesül az utófeltétele:

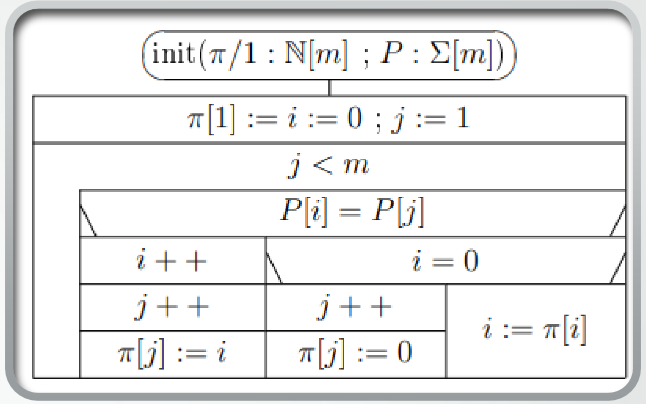
- $\forall k \in 1..m : \pi[k] = \pi(k)$

- **Bizonyítás.**
  - Az init(π, P) eljárásnak a 20. tételből ismerős (inv) invariánsa és a ciklusfeltétel tagadása:
    - azaz $j \geq m$ alapján $j = m$

  \+ az (inv) invariáns ($\forall k \in 1..j : \pi[k] = \pi(k)$) tényezője

  ➢ az init(π, P) eljárás fenti utófeltétele azonnal adódik.

Az init(π, P) eljárás megállása és hatékonysága:



- Az eljárás ciklusa legfeljebb 2m–2 iterációt hajt végre: ($\Theta(m)$ műveletigény)
  - A ciklus minden egyes iterációja $\Theta(1)$ műveletigényű
  - Az init(π, P) eljárás ciklusa legalább m–1 iterációt hajt végre
    - A ciklus (inv) invariánsa szerint $0 \leq i < j \leq m$
    - Az első iteráció előtt $j = 1$
    - \+ Mindegyik iteráció legfeljebb eggyel növeli a j változót
    - \+ a ciklusfeltétel $j < m$
    - ➢ legalább m–1 iteráció szükséges ahhoz, hogy $j = m$ legyen, és az eljárás befejeződjék
  - Az init(π, P) eljárás ciklusa legfeljebb 2m–2 iterációt hajt végre

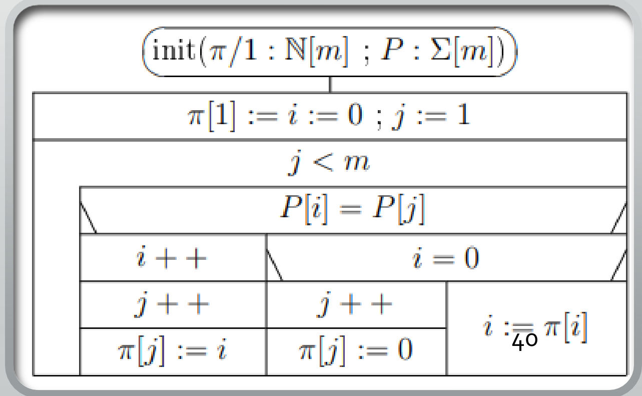
Az init(π, P) eljárás megállása és hatékonysága:

- Az init(π, P) eljárás ciklusa legfeljebb $2m-2$ iterációt hajt végre
  - Tekintsük a $2j-i$ kifejezést!
    - $0 \leq i < j \leq m \Rightarrow 2j\text{-}i \in 2\,..\,2\text{m}$
      - $j-i \geq 1 \land j \geq 1 \Rightarrow 2j-i \geq 2$
      - $j \leq m \Rightarrow 2j \leq 2m \Rightarrow 2j-i \leq 2m$, mivel $i \geq 0$
    - Az első iteráció előtt $2j-i = 2$

    \+ mindegyik iterációval (3 programágon) szigorúan monoton nő

    \+ végig $2j-i \leq 2m$

    ➢ legfeljebb $2m-2$ iteráció után megáll a ciklus



A KMP algoritmus összegzése

- KMP algoritmusnál legjobb és a legrosszabb eset absztrakt műveletigénye között kétszeres szorzó van
    - a futási idő is nagyon stabil
    - a legrosszabb esetben is meglepően hatékony, a szöveg hosszának közelítőleg lineáris függvénye
- Online alkalmazásoknál
    - a hatékonyság (a legrosszabb esetben is) és a futási idő stabilitása együtt, határozott előny a másik két algoritmussal szemben
- Offline alkalmazásoknál
    - a Quicksearch várható futási ideje jobb, mint a KMP algoritmusé => ez lehet előnyösebb
- A KMP(T, P, S) eljárás i változója sohasem csökken => a szövegen sohasem kell visszalépni
    - Mivel a T[0 . . n) szövegre csak a T[i] kifejezésen keresztül hivatkozunk
    - a KMP algoritmus kényelmesen, hatékonyan implementálható akkor is, ha a szöveg egy szekvenciális fájlban van
    - a Brute-force és a Quicksearch algoritmusoknál van visszalépés
        - a szövegen esetleg $m-2$ karakternyit is vissza kell lépni
        - szekvenciális input fájl: az utoljára beolvasott $m-1$ karakterét folyamatosan nyilván kell tartani egy átmeneti tárolóban

A KMP algoritmus szemléltetése

- Az init(π, P) algoritmus szemléltetése az ABABBABA mintán:
- (A három programág mindegyikének az elején kezdünk új sort.)

i	j	$\pi[j]$	$\overset{0}{A}$	$\overset{1}{B}$	$\overset{2}{A}$	$\overset{3}{B}$	$\overset{4}{B}$	$\overset{5}{A}$	$\overset{6}{B}$	$\overset{7}{A}$
0	1	0		$\cancel{A}$						
0	2	0			$\underline{A}$					
1	3	1			A	$\underline{B}$				
2	4	2			A	B	$\cancel{A}$			
0	4	2					$\cancel{A}$			
0	5	0						$\underline{A}$		
1	6	1						A	$\underline{B}$	
2	7	2						A	B	$\underline{A}$
3	8	3								

A végeredmény:

$P[j-1] =$	A	B	A	B	B	A	B	A
$j =$	1	2	3	4	5	6	7	8
$\pi[j] =$	0	0	1	2	0	1	2	3

Példa:

- A P[0 . . 8) = ABABBABA mintát keressük
- A T[0 . . 17) = ABABABBABABBABABA szövegben
- A mintához tartozó $\pi/1 : \mathbb{N}[8]$ tömböt fentebb már kiszámoltuk.

A végeredmény:

$P[j-1] =$	A	B	A	B	B	A	B	A
$j =$	1	2	3	4	5	6	7	8
$\pi[j] =$	0	0	1	2	0	1	2	3

A keresés:

$i = 0$		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$T[i]=$	A	B	A	B	A	B	B	A	B	A	B	B	A	B	A	B	A
	A	B	A	B	~~B~~												
s=2			A	B	A	B	B	A	B	A							
s=7								A	B	A	B	B	A	B	A		
													A	B	A	B	~~B~~
															A	B	A

$$S = \{\,2; 7\,\} = V$$

Ellenőrző kérdések: Quick-search

1. Mit számol ki a Quick Search algoritmus?
   - Mi az előnye, illetve hátránya a naiv mintaillesztő algoritmussal összehasonlítva?
   - Mutassa be a Quick Search algoritmus (a) előkészítő eljárásának működését
     - az {A;B;C;D} ábécé-vel
     - az ABACABA mintán
   - És e mintát illesztő eljárását
     - az ABABACABACABADABACABABA szövegen!
   - Mekkora az egyes eljárások aszimptotikus műveletigénye?

Ellenőrző kérdések: Knuth-Morris-Pratt (KMP)

1. Deniálja a KMP algoritmusban a keresett P[1::m] mintához tartozó π függvényt (nem a struktogramot)!
   - Adja meg a π függvény három alapvető tulajdonságát, és kettőt bizonyítson is be!
   - Adja meg a π függvényt a definíciója alapján
     - a BABAABAB mintán!
2. Szemléltesse a KMP algoritmus működését,
   - ahogy a fenti minta előfordulásait keresi
   - az ABABABAABABAABABABAABABBABA szövegben!

Köszönöm a figyelmet!

Pusztai Kinga

A bemutató Ásványi Tibor: Algoritmusok és adatszerkezetek II. eladásjegyzet:Mintaillesztés, tömörítés alapján készült.