

Mintaillesztés

Feladat: Egy szövegben egy minta összes előfordulását keressük.

Jelölések:

- szöveg: $T[1..n]$
- minta: $P[1..m]$ (Feltesszük, hogy $1 \leq m \leq n$.)
- Érvényes eltolások halmaza: S
- ábécé: $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_d\}$ ($\sigma_1, \sigma_2, \dots, \sigma_d$ az ábécé betűi)

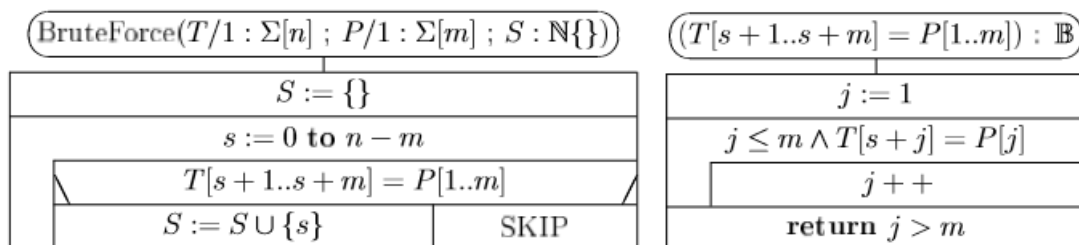
Mintaillesztő algoritmusok **összehasonlításánál** két fontos kérdéssel foglalkozunk:

- Műveletigény (hány összehasonlítást végzünk el)
- Mennyire „ugrál” a szövegben (Ez azért fontos, mert előfordulhat, hogy a mintaillesztést olyan adatszerkezetre kell megoldani (pl. szekvenciális fájl, bináris fa), ahol nem megengedett művelet az indexelés. Ekkor az olyan algoritmusoknál, ahol a szövegben ugrálunk, buffer segítségét kell igénybe venni.)

Brute-Force (BF) algoritmus:

A Brute-Force (nyers erő) algoritmus nem más, mint egy kiválogatásba ágyazott optimista lineáris keresés. Az algoritmust szemléletesen úgy lehet elképzelni, mintha a mintát tartalmazó sablont tolnánk végig a szövegben, és balról jobbra ellenőrizzük, hogy a minta karakterei egyeznek-e a lefedett szöveg karaktereivel. Amennyiben nem egyező karakterpárt találunk, a mintát eggyel jobbra toljuk a szövegben, és megint kezdjük a minta elejéről az összehasonlítást.

BF algoritmus stuktogramja



Műveletigény:

Legjobb eset: A minta első karaktere nincs a szövegben.

Ekkor $M\ddot{O}(n, m) = n - m + 1 \in \Theta(n)$ (A betűk összehasonlításait számoljuk, ami most lényegében véve megfelel a ciklusiterációk számának.)

Legrosszabb eset: A minta minden eltolásnál csak a minta utolsó karakterénél romlik el az illeszkedés.

Ekkor $M\ddot{O}(n, m) = (n - m + 1) * m \in \Theta(n * m)$ ($m \ll n$ esetén)

Szekvenciális sorozatokra, fájlokra való alkalmazhatóság:

A szövegben „ugrálunk”, ezért az olyan adatszerkezeteknél ahol nem megengedett az indexelés, szükség van buffer használatára.

Knuth-Morris-Pratt (KMP) algoritmus:

A KMP algoritmusnál nem szükséges minden esetben a minta elejéről kezdeni az illeszkedést. Amennyiben a mintával akkorát ugrunk, hogy a minta kezdőszelete (prefixe) egy valódi végszeletnél

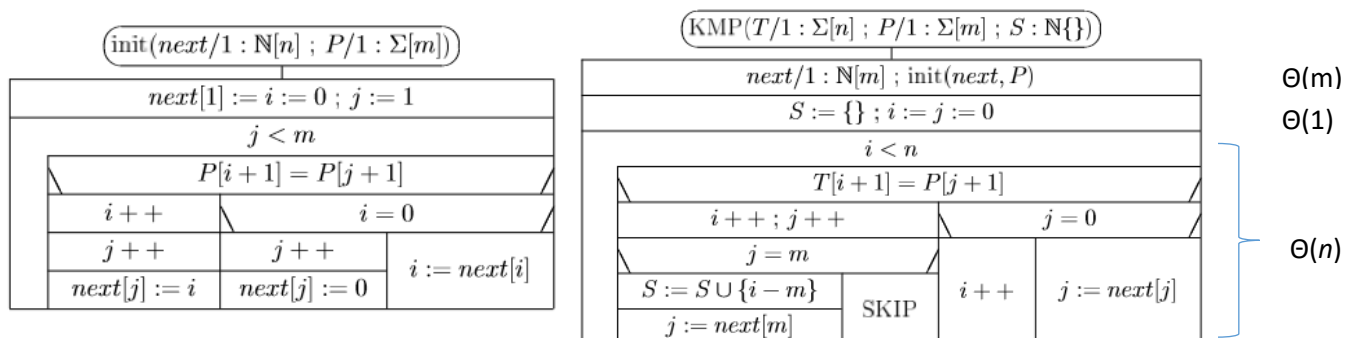
(szuffix) kezdődjön, azaz a prefix a szuffixel kerüljön fedésbe, a prefixet már nem kell újra vizsgálni. Ehhez a minta előfeldolgozására van szükség.

Az előfeldolgozás során definiálunk egy **next függvényt**, amely megadja a minta egyes kezdőrészleteire a leghosszabb egymással egyező prefix-szuffix párok hosszát. Ezt felhasználva tudjuk megadni a mintával való „ugrás” mértékét.

A next függvény alapvető tulajdonságai:

1. $next(j) \in 0..(j-1)$ ($j \in 1..m$)
2. $next(j+1) \leq next(j)+1$ ($j \in 1..m-1$)
3. $P_{h+1} \sqsupset T_{j+1} \Leftrightarrow P_h \sqsupset T_j \wedge P[h+1] = T[j+1]$
4. $0 \leq h < j \leq m$ és $P_j \sqsupset T_i$ esetén $P_h \sqsupset T_i \Leftrightarrow P_h \sqsupset P_j$
5. $\max_{l+1} H(j) = next(\max_l H(j))$ ($j \in 1..m, l \in 1..|H(j)|-1$)

KMP algoritmus stuktogramja:



Műveletigény:

Az *init* műveletigénye $\Theta(m)$. (Ahol m a minta hossza.)

Tegyük fel, hogy $m \leq n$, ekkor a *KMP* műveletigénye legjobb és legrosszabb esetben is $\Theta(n)$.

($T \in \Omega(n)$, mivel i növekedni egyesével tud, és n -ig nő \Rightarrow biztos van n lefutás)

$T \in O(n)$ $0 \leq j \leq i \leq n$, mivel a fő ciklus max $2n$ -szer fut le.)

KMP algoritmus előnye:

A szövegben nem kell visszaugrani. Ennek jelentősége pl szekvenciális sorozat/fájl formában adott szövegnél van, mivel ekkor buffer használata nélkül is tudjuk alkalmazni a KMP algoritmust.

1. példa: $P[1..6]=ABABAC, T[1..9]=ABABABACA$

Az $init(next, P)$ algoritmus szemléltetése az *ABABAC* mintán:

i	j	$next[j]$	1	2	3	4	5	6
			A	B	A	B	A	C
0	1	0		A				
0	2	0			A			
1	3	1			A	B		
2	4	2			A	B	A	
3	5	3			A	B	A	B
1	5	3					A	B
0	5	3						A
0	6	0						

A végeredmény:

$P[j]=$	A	B	A	B	A	C
$j=$	1	2	3	4	5	6
$next[j]=$	0	0	1	2	3	0

A $P[1..6] = ABABAC$ mintát keressük a $T[1..9] = ABABABACA$ szövegben.

$j=$	Hasonlítások száma	1	2	3	4	5	6	7	8	9	Magyarázat
$T[i]=$		A	B	A	B	A	B	A	C	A	
	6	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	€				Ekkor a szöveg és a minta 5. pozíciójánál vagyunk ($i=j=5$), és a szöveg 6. karakterét hasonlítjuk a minta 6. karakterével. Mivel nincs egyezés, a mintában az aktuális pozíciókat (5) felülírjuk a $next[5]$ -tel, azaz a szövegben az 5. pozíción maradunk, de a mintában a 3.-nál leszünk és a szöveg 6. pozícióját hasonlítjuk a mintánk 4. elemével.
$s=2$	3			A	B	A	<u>B</u>	<u>A</u>	<u>C</u>		A piros karaktereket nem hasonlítjuk, mivel a $next[5]=3$ azt jelenti, hogy a mintánk első három eleme megegyezik a 3.-5. elemével.
	1									<u>A</u>	

$S=\{2\}$

2. példa: $P[1..7]=BABABAB$, $T[1..9]=BABBABABABBBABABABAAB$

Az $init(next,P)$ algoritmus szemléltetése az $ABABAC$ mintán:

i	j	$next[j]$	1 B	2 A	3 B	4 A	5 B	6 A	7 B
0	1	0		B					
0	2	0			<u>B</u>				
1	3	1			B	<u>A</u>			
2	4	2			B	A	<u>B</u>		
3	5	3			B	A	B	<u>A</u>	
4	6	4			B	A	B	A	<u>B</u>
5	7	5							

A végeredmény:

$P[j]=$	B	A	B	A	B	A	B
$j=$	1	2	3	4	5	6	7
$next[j]=$	0	0	1	2	3	4	5

A $P[1..7] = \text{BABABAB}$ mintát keressük a $T[1..22] = \text{BABBABABAB ABBABABABAAB}$ szövegben.

$i=$	H.SZ.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$T[i]=$		B	A	B	B	A	B	A	B	A	B	A	B	B	A	B	A	B	A	B	A	A	B
	4	<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>																		
	1			<u>B</u>	<u>A</u>																		
$s=3$	7				<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>												
$s=5$	2						<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>										
	1								<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>									
	1									<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>										
	1											<u>B</u>	<u>A</u>										
$s=12$	7													<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>			
	2															<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>	
	1																<u>B</u>	<u>A</u>	<u>B</u>	<u>A</u>	<u>B</u>		
	1																		<u>B</u>	<u>A</u>	<u>B</u>		
	1																				<u>B</u>		
	1																						<u>B</u>

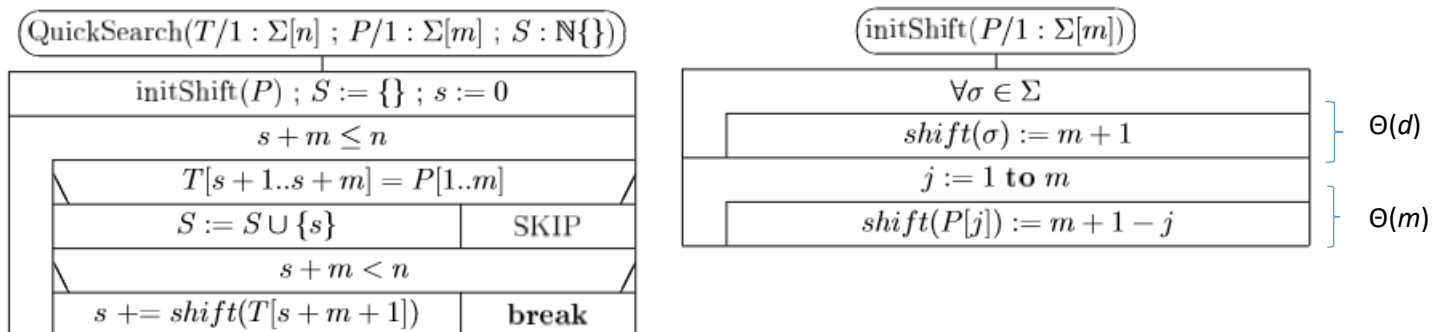
$S=\{3; 5; 12\}$

Quick Search (QS) algoritmus:

Alapötlet: Ha elromlik az illeszkedés, akkor nézzük a szövegben a minta utáni karaktert, és úgy toljuk el a mintát, hogy illeszkedjen a szöveg ezen karakteréhez. Ha a mintában nem szerepel ez a karakter, akkor átugorjuk a mintával. Az új vizsgálatot mindig a minta elejéről nézzük.

A mintával való „ugrás” végrehajtásához bevezetjük a shift függvényt. A shift függvény az ABC minden betűjére megadja az "ugrás" nagyságát, amelyet akkor tehetünk, ha az illeszkedés elromlása esetén az illető betű lenne a szöveg minta utáni első karaktere.

Quicksearch algoritmus stuktogramja:



Műveletigény:

Az initShift műveletigénye: $\Theta(d) + \Theta(m) \in \Theta(m)$ (d : az ábécé elemszáma, konstans)

legjobb eset: $M\ddot{O}(n, m) \in \Theta(n/(m+1))$ (A minta első karakterénél már elromlik az illeszkedés, továbbá a minta utáni karakter sem fordul elő a mintában, így azt „átugorjuk”.)

legrosszabb eset: $M\ddot{O}(n) \in \Theta(n * m)$ ($m \ll n$ esetén. A minta végén romlik el az illeszkedés, így kicsi az „ugrás”.)

Szekvenciális sorozatokra, fájlokra való alkalmazhatóság:

A szövegben „ugrálunk”, ezért az olyan adatszerkezeteknél ahol nem megengedett az indexelés, szükség van buffer használatára.

1. példa: $P=ABCA$, $T=ABDAEBBCBBCABCABABCA$

A szöveg alapján az ábécé: $\Sigma=\{A, B, C, D, E\}$

Előkészítő eljárás:

σ		A	B	C	D	E
		5	5	5	5	5
A	1	4				
B	2		3			
C	3			2		
A	4	1				
SHIFT(σ)		1	3	2	5	5

Mintaillesztő eljárás:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	B	D	A	E	B	B	C	B	B	C	A	B	C	B	A	B	C	A
A	B	C	A				C	A			A							
					A	B		A	B	C	A			A				
											A	B	C	C	A			
											A	A	B		A	B	C	A

- | | | |
|------------------|-----------------------------------|--|
| 1. elromlás: D-C | A minta utáni elem (5.) E | shift(E)=5, ezért a mintát 5-tel toljuk el. |
| 2. elromlás: B-A | A minta utáni elem (10.) B | shift(B)=3, ezért a mintát 3-mal toljuk el. |
| 3. elromlás: B-A | A minta utáni elem (13.) B | shift(B)=3, ezért a mintát 3-mal toljuk el. |
| 4. elromlás: B-A | A minta utáni elem (16.) A | shift(A)=1, ezért a mintát 1-gyel toljuk el. |
| 5. elromlás: B-A | A minta utáni elem (17.) B | shift(B)=3, ezért a mintát 3-mal toljuk el. |

$$\Sigma = \{A, B, C, D, E, G, H\}$$

σ	A	B	C	D	E	G	H
SHIFT(σ)	3	2	1	6	7	7	7

[illegible]

eltolások:

$$\text{shift}(B)=2$$

shift(A)=3

$$\text{shift}(C)=1$$

shift(A)=3

shift(A)=3

$$\text{shift}(B)=2$$

shift(A)=3