

Algoritmusok és adatszerkezetek II.

Sztring,-és számelméleti algoritmusok

Szegedi Tudományegyetem



$T = [t_1 \dots t_n]$ input tartalmazza-e a $P = [p_1 \dots p_m]$ mintát? Ha igen, mely inputpozíció(k)tól/eltolási érték(ek)től kezdődően?
Tipikusan $n \gg m$



$T = [t_1 \dots t_n]$ input tartalmazza-e a $P = [p_1 \dots p_m]$ mintát? Ha igen, mely inputpozíció(k)tól/eltolási érték(ek)től kezdődően?

Tipikusan $n \gg m$

Példa

Hányszor szerepel az 'orosz' szó a *Háború és béke* c. műben?

Nyers erőt használva legrosszabb esetben $O(mn)$ vizsgálat kell
Milyen hatékonyabb módszerek vannak?

- Mintaillesztés automatával
- Knuth-Morris-Prat
- Rabin-Karp algoritmus



- P_i jelölje P -nek az i hosszúságú prefixét (kezdőszeletét), azaz pl. $P_3 = bab$, $P_1 = b$, illetve $P_0 = \epsilon$ (ϵ az üres szót jelöli)



- P_i jelölje P -nek az i hosszúságú prefixét (kezdőszeletét), azaz pl. $P_3 = bab$, $P_1 = b$, illetve $P_0 = \epsilon$ (ϵ az üres szót jelöli)
- $X \sqsubset Y$ jelölje azt, ha X sztring szuffixe Y -nak (azaz Y végződése maga X)

Példa

aaba \sqsubset cacaaaba, ugyanakkor aaba $\not\sqsubset$ cacaab**b**

Megjegyzés: Az $Y \sqsubset Y$, valamint az $\epsilon \sqsubset Y$ relációk triviálisan teljesülnek minden Y -ra.



Automata alatt egy $M = (Q, q_0, A, \Sigma, \delta)$ rendezett ötöst értünk, ahol

- Q a lehetséges állapotok halmaza
- q_0 a kezdőállapot
- $A \subseteq Q$ a végállapotok halmaza
- Σ egy véges ábécé
- $\delta : Q \times \Sigma \rightarrow Q$ az állapotátmenet-függvény



Mintaillesztés véges állapotú automatákkal

- Q -t válasszuk $\{q_0, q_1, \dots, q_m\}$ -nak
- q_i állapot jelentése: az input aktuális pozíciójáig a minta első i karaktere illeszkedik
- q_m állapotba elérve elmondható, hogy megtaláltuk a P -nek egy T -beli előfordulását



```
ÁTMENETFÜGGVÉNYSZÁMÍTÁS(P,  $\Sigma$ ) {  
  m = P.length  
  for (q = 0 to m) {  
    for (a  $\in$   $\Sigma$ ) {  
      k = min(m + 1, q + 2)  
      repeat  
        k = k - 1  
      until  $P_k \sqsupseteq P_q a$   
       $\delta(q, a) = k$   
    }  
  }  
  return  $\delta$   
}
```

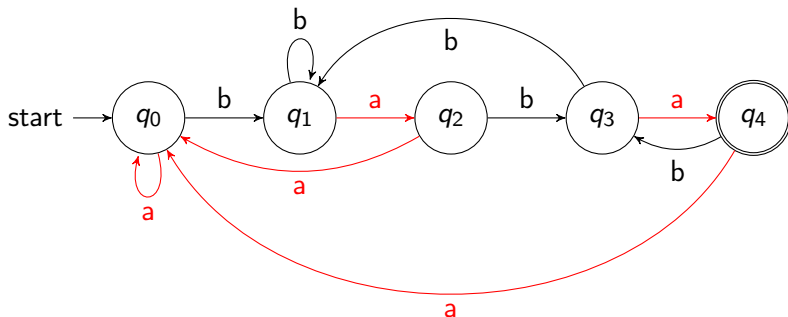


$P = baba$ minta felismerését végző véges automata

Az állapotátmenet-függvény táblázatos formában

	q_0	q_1	q_2	q_3	q_4
a	q_0	q_2	q_0	q_4	q_0
b	q_1	q_1	q_3	q_1	q_3

Kitöltése és tárolása egyaránt $O(m|\Sigma|)$ költségű



Knuth-Morris-Pratt (KMP) algoritmus

A véges automatával történő feldolgozás korlátai

Előfeldolgozás gyanánt ki kell tölteni egy $m|\Sigma|$ méretű táblázatot
(és persze tárolni is kell azt)



A véges automatával történő feldolgozás korlátai

Előfeldolgozás gyanánt ki kell tölteni egy $m|\Sigma|$ méretű táblázatot (és persze tárolni is kell azt)

Észrevételek

- 1 Ha $P = aab$ minta illeszkedett az input i -edik pozíciójára, akkor az $i + 1$ pozíciótól kezdődően biztos nem beszélhetünk illeszkedésről
- 2 Ha $P = aaa$ minta nem illeszkedett az input i -edik pozíciójára, akkor az $i + 1$ pozíciótól kezdődően biztos nem beszélhetünk illeszkedésről



- $\pi : \{1, 2, \dots, m\} \rightarrow \{0, 1, \dots, m-1\}$ $P = [p_1 \dots p_m]$ minta prefixfüggvénye, ha $\pi[q] = \max\{k : k < q \wedge P_k \sqsubseteq P_q\}$
- Azaz $\pi[q]$ megadja P azon leghosszabb (q -nál rövidebb) prefixének hosszát, ami valódi szuffixe $P_q = [p_0 \dots p_q]$ -nak
- Értelmezése: ha **nem sikerül továbbillesszük** a mintát az inputra, az **legalább** mely állapotig vet minket vissza



- $\pi : \{1, 2, \dots, m\} \rightarrow \{0, 1, \dots, m-1\}$ $P = [p_1 \dots p_m]$ minta prefixfüggvénye, ha $\pi[q] = \max\{k : k < q \wedge P_k \sqsubset P_q\}$
- Azaz $\pi[q]$ megadja P azon leghosszabb (q -nál rövidebb) prefixének hosszát, ami valódi szuffixe $P_q = [p_0 \dots p_q]$ -nak
- Értelmezése: ha **nem sikerül továbbilleszük** a mintát az inputra, az **legalább** mely állapotig vet minket vissza

Példa

$T = abaabababaca$ inputban keressük a $P = ababababca$ mintát

i	1	2	3	4	5	6	7	8	9	10
$P[i]$	a	b	a	b	a	b	a	b	c	a
$\pi[i]$	0	0	1	2	3	4	5	6	0	1



Indirekt belátható, hogy $\pi[i + 1] - \pi[i] \leq 1$ minden esetben teljesül



A prefixfüggvény növekedése

Indirekt belátható, hogy $\pi[i+1] - \pi[i] \leq 1$ minden esetben teljesül

$$\begin{array}{ccccccc} \overbrace{s_1 \ s_2}^{\pi[i]=2} & s_3 & s_4 & \dots & s_{i-2} & \overbrace{s_{i-1} \ s_i}^{\pi[i]=2} & s_{i+1} \\ \underbrace{\hspace{1.5cm}}_{\pi[i+1]=4} & & & & & \underbrace{\hspace{1.5cm}}_{\pi[i+1]=4} & \end{array}$$

Azaz $\pi[i+1] - \pi[i] > 1$ csak úgy teljesülhet, ha $\pi[i]$ nem P_i leghosszabb valódi szuffixének a hossza, ami ellentmondás.



- Folytatódó illeszkedés

$$\begin{array}{c}
 \overbrace{s_1 \ s_2 \ s_3}^{\pi[i]} \quad s_4 = s_{i+1} \quad \overbrace{s_{i-2} \ s_{i-1} \ s_i}^{\pi[i]} \quad s_{i+1} = s_i + 1 \\
 \underbrace{\hspace{1.5cm}}_{\pi[i+1] = \pi[i] + 1} \quad \dots \quad \underbrace{\hspace{1.5cm}}_{\pi[i+1] = \pi[i] + 1} \\
 \text{Red: } s_4, s_{i+1}
 \end{array}$$

- Nem (közvetlen) folytatódó illeszkedés

$$\begin{array}{c}
 \overbrace{s_1 \ s_2 \ s_3 \ s_4}^{\pi[i]} \quad \dots \quad \overbrace{s_{i-3} \ s_{i-2} \ s_{i-1} \ s_i}^{\pi[i]} \quad s_{i+1} \\
 \underbrace{\hspace{1.5cm}}_j \quad \underbrace{\hspace{1.5cm}}_{j = \pi[\pi[i]]}
 \end{array}$$



Prefixfüggvény kiszámítása

```
PREFIXSZÁMÍT(P) {  
    m = P.length  
     $\pi[1] = 0$   
    i = 0  
    for (q=2 to m) {  
        while i > 0 and P[i+1]  $\neq$  P[q]  
            i = P[q]  
        if P[i+1] = P[q]  
            i = i+1  
         $\pi[q]=i$   
    }  
    return  $\pi$   
}
```



KMP algoritmus

```
KMP-ILLESZTŐ(T, P) {  
    n = T.length  
    m = P.length  
     $\pi$  = PREFIXSZÁMÍT(P)  
    q = 0  
    for (i = 1 to n)  
        while q > 0 and P[q + 1]  $\neq$  T[i]  
            q =  $\pi$ [q]  
        if P[q + 1] = T[i]  
            q = q + 1  
        if q = m  
            print("Illeszkedés a %d. pozícióval bezárólag" % i)  
            q =  $\pi$ [m]  
}
```



Az egyszerűség kedvéért a mintára és az inputra tekintsünk 10-es számrendszerbeli számokként

Alapötlet

P mintára alkalmazzunk egy $h_q(x) = x \bmod q$ hasítófüggvényt.

T -nek csak azon $S = [t_j]_{j=i}^{i+m-1}$ résztringjei egyezhetnek meg

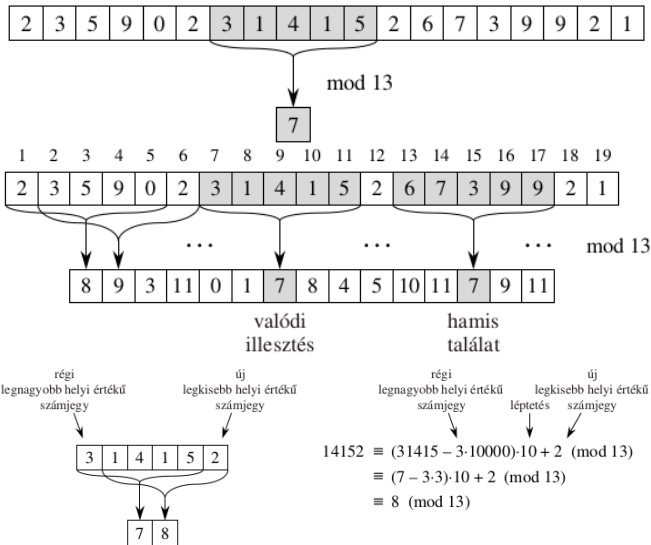
P -vel, melyekre $h_q(S) = h_q(P)$

$h_q(S) = h_q(P)$ természetesen csak szükségességi feltételt támaszt



Rabin-Karp algoritmus illusztráció (Forrás: CLRS 32.5 ábra)

$$P = 31415, h_q(x) = x \bmod 13$$



Észrevétel

h_q -nak az input minden indexén kezdődő kiszámítása költséges



Észrevétel

h_q -nak az input minden indexén kezdődő kiszámítása költséges

A hatékonyság záloga

Ahogy abc és bcd számpárral teljesül

$$bcd = 10 * (abc - 100 * a) + d$$

egyenlőség, hasonlóan igaz

$$h_q(bcd) = h_q(10 * (h_q(abc) - h_q(100) * a) + d)$$

összefüggés is.



- Probabilisztikus adatszerkezet a Keres műveletre nézve
- Implementációja: (egyenletesen szóró) h hasítófüggvény és egy (kezdetben csupa 0) bitvektor
 - x elem beszúrásakor a bitvektor $h(x)$ indexét 1-re állítjuk
 - x elem keresésnél ha a bitvektor $h(x)$ indexe 1, akkor azt mondjuk, hogy x -et tartalmazza a bloom filterünk
 - Hamis elutasítás soha nem teszünk, ugyanakkor hamis pozitív választ adhatunk
- Hatékonysága függ h hasítófüggvényről, valamint az eltárolt elemek számától (ami kihat a bitvektor kitöltöttségére)
- Linkek: [Bloom filter demo](#) és [Guava API](#)



Bloom filterek tévesztési arányának elemzése

- Egy bloom filter olyan arányban ad egy elem tartalmazására nézve igenlő választ, mint amennyi bitje egyesre lett állítva a beszúrások hatására
 - Feltesszük, hogy az alkalmazott h hasítófüggvény jól szór, azaz m ekvivalenciaosztályba való sorolás esetén az objektumok közelítőleg $\frac{1}{m}$ részét sorolja az összes osztályba

Hány bitje lesz egy m hosszú bloom filternek 1-es n beszúrás után?

1 beszúrás során $\frac{1}{m}$ valószínűséggel töltünk föl egy bitet

1 beszúrás során $1 - \frac{1}{m}$ valószínűséggel *nem* töltünk föl egy bitet

n beszúrás során $\left(1 - \frac{1}{m}\right)^n$ valószínűséggel *nem* töltünk föl egy bitet



Bloom filterek tévesztési arányának elemzése

- Egy bloom filter olyan arányban ad egy elem tartalmazására nézve igenlő választ, mint amennyi bitje egyesre lett állítva a beszúrások hatására
 - Feltesszük, hogy az alkalmazott h hasítófüggvény jól szór, azaz m ekvivalenciaosztályba való sorolás esetén az objektumok közelítőleg $\frac{1}{m}$ részét sorolja az összes osztályba

Hány bitje lesz egy m hosszú bloom filternek 1-es n beszúrás után?

1 beszúrás során $\frac{1}{m}$ valószínűséggel töltünk föl egy bitet

1 beszúrás során $1 - \frac{1}{m}$ valószínűséggel *nem* töltünk föl egy bitet

n beszúrás során $\left(1 - \frac{1}{m}\right)^n$ valószínűséggel *nem* töltünk föl egy bitet

$$\left(1 - \frac{1}{m}\right)^n = \left[\left(1 - \frac{1}{m}\right)^m\right]^{\frac{n}{m}} \approx e^{-\frac{n}{m}} \text{ (nagy } m \text{ esetén)}$$

n beszúrás követően $1 - e^{-\frac{n}{m}}$ valószínűséggel lesz a bloom filter valamely bitje 1-esre állítva



Mintaillesztés főbb megközelítései

Algoritmus	Előfeldolgozás	Illesztés ¹
Nyers erő	0	$O(mn)$
Rabin-Karp	$\Theta(m)$	$O(mn)$
Véges automata	$O(m \Sigma)$	$\Theta(n)$
Knuth-Morris-Prat	$\Theta(m)$	$\Theta(n)$

- Rabin-Karp a nyers erő módszerének heurisztikus kiterjesztéseként tekinthető
- KMP pedig a véges automatákkal való illesztés egy hatékonyabb verziója

¹legrosszabb eset



A moduláris hatványozás

Alapprobléma: Mi az $a^b \bmod n$ kifejezés értéke?

Gyakorlati jelentőség: titkosító eljárások (pl. RSA) használata során szükségünk lehet ilyen számítások elvégzésére

Már viszonylag kis b -re is rengeteg bitműveletet kell elvégezzünk



A moduláris hatványozás

Alapprobléma: Mi az $a^b \bmod n$ kifejezés értéke?

Gyakorlati jelentőség: titkosító eljárások (pl. RSA) használata során szükségünk lehet ilyen számítások elvégzésére

Már viszonylag kis b -re is rengeteg bitműveletet kell elvégezzünk

Példa – Mi lesz $7^{560} \bmod 561$ értéke? Avagy

179846672920572906577258722224560336083856608137007342561
612636144396089769552955665549135995604075608096691162101
184113545972526638255004784055311390598542305958357097082
391225061077433281620117130013826448606281708665937931659
796736755253074977366471063146923373865223501532185753076
292710887401801774392779475679510556311966981952826025487
057204699261913973664257857993744045143447531121540574215
969802961003872086163860991702035506591312847029674260362
509156745965136001 mod 561=?



Moduláris hatványozás

```
MODULÁRIS-HATVÁNYOZÓ(a, b, n){
```

```
    c = 0
```

```
    d = 1
```

```
    B = [bk ... b1 b0]
```

```
    for (i=k; k >=0; --k) {
```

```
        c = 2*c
```

```
        d = (d*d) mod n
```

```
        if (B[i] == 1) {
```

```
            c = c+1
```

```
            d = (d*a) mod n
```

```
        }
```

```
    }
```

```
    return d
```

```
}
```

Megjegyzések

- 1 *b*-nek *k* bites bináris alakja *B*
- 2 *c* csak egy segédváltozó



```
MODULÁRIS-HATVÁNYOZÓ(a, b, n){
```

```
    c = 0
```

```
    d = 1
```

```
    B = [bk ... b1 b0]
```

```
    for (i=k; k >=0; --k) {
```

```
        c = 2*c
```

```
        d = (d*d) mod n
```

```
        if (B[i] == 1) {
```

```
            c = c+1
```

```
            d = (d*a) mod n
```

```
        }
```

```
    }
```

```
    return d
```

```
}
```

Megjegyzések

- 1 b -nek k bites bináris alakja B
- 2 c csak egy segédváltozó

Kérdés

- Mi lesz d és c értéke a for ciklus első végrehajtása után?



Moduláris hatványozás

```
MODULÁRIS-HATVÁNYOZÓ(a, b, n){
```

```
    c = 0
```

```
    d = 1
```

```
    B = [bk ... b1 b0]
```

```
    for (i=k; k >=0; --k) {
```

```
        c = 2*c
```

```
        d = (d*d) mod n
```

```
        if (B[i] == 1) {
```

```
            c = c+1
```

```
            d = (d*a) mod n
```

```
        }
```

```
    }
```

```
    return d
```

```
}
```

Megjegyzések

- 1 b -nek k bites bináris alakja B
- 2 c csak egy segédváltozó

Kérdés

- Mi lesz d és c értéke a for ciklus első végrehajtása után?

Műveletigény

Ha a, b és n k biten elfér, mindez $O(k)$ aritmetikai műveletet és $O(k^3)$ bitműveletet jelent



A moduláris hatványozás működése

- Az algoritmus for ciklusában minden egyes iteráció megkezdése előtt a c és d változók értékeire teljesül, hogy:
 - 1 c változó értéke megegyezik a $[b_k \dots b_{i+1}]$ bináris értékkel
 - 2 $d = a^c \bmod n$
- A fenti ciklusinvariáns teljesül, megmarad és befejeződik \rightarrow



Moduláris hatványozás példa

$$7^{560} \bmod 561 = ?$$

$$a = 7, b = 560, n = 561 \ (B = [1000110000])$$

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

