Algoritmi in podatkovne strukture 2

Iskanje v nizih

Luka Fürst

Niz

- Končno zaporedje simbolov iz končne neprazne abecede Σ
- $S = a_1 a_2 \ldots a_n$
  - $a_i \in \Sigma$ za $i \in \{1, \ldots, n\}$
- $|S| = n$
  - dolžina niza S
- $S[i] = a_i$
  - indeksi se pričnejo z 1
  - se opravičujem, a tokrat mi to dejansko ustreza
- $S[i : j] = a_i a_{i+1} \ldots a_j$
- $S[i :] = a_i a_{i+1} \ldots a_n$
- $S[: i] = a_1 a_2 \ldots a_i$

} Odmislite python!

Niz

- Niz R je predpona niza S, če je $|R| \leq |S|$ in $R = S[: |R|]$
  - $R \sqsubseteq S$
  - $\texttt{aba} \sqsubseteq \texttt{aba} \sqsubseteq \texttt{abac}$
- Niz R je pripona niza S, če je $|R| \leq |S|$ in $R = S[|S| - |R| + 1 :]$
  - $R \sqsupseteq S$
  - $\texttt{aba} \sqsupseteq \texttt{aba} \sqsupseteq \texttt{caba}$

Iskanje vzorca v besedilu

- Naj bosta T in P niza nad abecedo Σ
- T: besedilo (*text*)
- P: vzorec (*pattern*)
- $|T| = n$
- $|P| = m$
- $1 \leq m \leq n$

Opis problema

- Poišči vse pojavitve niza P v nizu T
- Določi vse odmike s, tako da je $T[s+1 : s+m] = P$
  - $P \sqsupseteq T[s+m]$
- Primer
  - $T =$ `abbababacaba`
  - $P =$ `aba`
  - $s_1 = 3$ (`abbababacaba`)
  - $s_1 = 5$ (`abbababacaba`)
  - $s_1 = 9$ (`abbababacaba`)

Naivni pristop

- Za vsak $s \in \{0, 1, \ldots, n - m + 1\}$ preveri, ali je $T[s + 1 : s + m] = P$
- $n - m + 1$ odmikov (položajev vzorca v besedilu)
- $O(m)$ za primerjavo nizov $T[s + 1 : s + m]$ in P
- skupaj $O(m(n - m + 1))$ ali (enostavneje) $O(mn)$
- Se da bolje?
- Asimpotična spodnja meja je namreč $\Omega(n)$

Rabin-Karpov algoritem

- Naj bo $\Sigma = \{a_1, a_2, \ldots, a_B\}$
- Niz nad Σ si lahko predstavljamo kot število v sistemu z osnovo B
- Naj bo $f(S)$ število, ki pripada nizu S
- $f(a_i) = i - 1$
- $f(a_{i_1} a_{i_2} \ldots a_{i_n}) = (i_1 - 1)B^{n-1} + (i_2 - 1)B^{n-2} + \ldots + (i_n - 1)B^0$

Primer

- $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$
- $B = 3$
- $f(\mathtt{a}) = 0,\ f(\mathtt{b}) = 1,\ f(\mathtt{c}) = 2$
- $f(\mathtt{babca}) = 10120_{(3)} = 1 \cdot 3^4 + 0 \cdot 3^3 + 1 \cdot 3^2 + 2 \cdot 3^1 + 0 \cdot 3^0 = 96$

Rabin-Karpov algoritem

- Funkcija f je pri fiksni dolžini niza bijektivna
  - velja namreč $f(\texttt{ababca}) = f(\texttt{babca})$
- Ker je dolžina vzorca fiksna, lahko namesto P in $T[s+1:s+m]$ primerjamo $f(P)$ in $f(T[s+1:s+m])$

$$P = T[s+1:s+m] \iff f(P) = f(T[s+1:s+m])$$

Rabin-Karpov algoritem

- Potence B^0, B^1, ..., B^{m-1} izračunamo vnaprej
  - $O(m)$, enkratna operacija
- Za vzorec P izračunamo $f(P)$
  - $O(m)$, enkratna operacija
- Za vsak odmik s izračunamo $f(T[s+1:s+m])$
  - $O(m)$ za vsak odmik
  - $O(n-m+1)$ odmikov
  - skupaj $O((n-m+1)m)$
  - hm, to ni nič bolje kot naivno ...

Učinkovito posodabljanje $f(.)$

- Recimo, da smo v besedilu pravkar odkrili niz babca
- $f(\texttt{babca}) = f(\texttt{b}) \cdot 3^4 + f(\texttt{a}) \cdot 3^3 + f(\texttt{b}) \cdot 3^2 + f(\texttt{c}) \cdot 3^1 + f(\texttt{a})$
- Recimo, da je naslednji znak besedila enak c
- Ali lahko učinkovito izračunamo $f(\texttt{abcac})$ na podlagi $f(\texttt{babca})$?

Učinkovito posodabljanje $f(.)$

- Da!
- Naj bo $F = f(\texttt{babca})$
- Odbijemo prvi b
  - $F' = F - f(\texttt{b}) \cdot 3^4$
- Preostanek ($\texttt{abca}$) premaknemo za eno mesto v levo
  - $F'' = 3F'$
- Prištejemo $f(\texttt{c})$
- $f(\texttt{abcac}) = 3\left(f(\texttt{babca}) - f(\texttt{b}) \cdot 3^4\right) + f(\texttt{c})$

Učinkovito posodabljanje $f(.)$

- V splošnem:

$$f(T[s+2:s+m+1]) =$$
$$B\left(f(T[s+1:s+m]) - f(T[s+1])B^{m-1}\right) + f(T[s+m+1])$$

- $O(m)$ časa za enkratno predobdelavo vzorca
- $O(n-m+1)$ ali (enostavneje) $O(n)$ časa za iskanje vzorca v besedilu

Kaj pa, če . . .

- . . . sta abeceda in/ali vzorec P prevelika, da bi lahko $f(P)$ učinkovito izračunali?
- Spomnimo se na množenje velikih števil!

Računanje $f(.)$ po modulu

- V tem primeru nimamo druge možnosti, kot da $f(.)$ računamo po nekem modulu q
- Modul naj bo praštevilo in naj bo karseda velik
- Kljub temu se situacijam, ko je $f(S) = f(R)$ za $S \neq R$, v splošnem ne moremo izogniti
- Nizov dolžine m nad abecedo velikosti B je B^m, to pa je v realnih primerih praviloma bistveno več od smiselnega modula

Računanje $f(.)$ po modulu

- Če je $f(T[s+1:s+m]) \neq f(P)$, potem se P gotovo ne nahaja na odmiku s
- Če je $f(T[s+1:s+m]) = f(P)$, potem se P »zelo verjetno« nahaja na odmiku s, a moramo še preveriti, ali je res $T[s+1:s+m] = P$
- Na ta način z $O(n)$ ponovno pridemo na $O(mn)$, kljub temu pa je v praksi Rabin-Karp hitrejši od naivnega algoritma

Rabin-Karpov algoritem

function RABIN-KARP(T, P, B, q)
    $n \leftarrow |T|$
    $m \leftarrow |P|$
    $f \leftarrow 0$
    $g \leftarrow 0$
    $h \leftarrow B^{m-1} \bmod q$
    for $i \leftarrow 1$ **to** m **do**
        $f \leftarrow (Bf + P[i]) \bmod q$
        $g \leftarrow (Bg + T[i]) \bmod q$
    for $s \leftarrow 0$ **to** $n - m$ **do**
        if $f = g$ **then**
            if $P = T[s + 1 : s + m]$ **then**
                PRINT(s)
        if $s < n - m$ **then**
            $g \leftarrow (B(g - T[s + 1]h) + T[s + m + 1]) \bmod q$

Vrnimo se na osnovno metodo . . .

- Kako bi lahko povečali učinkovitost osnovne metode?
- Naj bo $P =$ `abcd`
- Recimo, da se `abc` ujema z besedilom na odmiku s, `d` pa se ne ujema

  T: . . .`abcx`. . .
  P: `abcd`

- Ali je nujno, da v naslednjem koraku poskusimo z odmikom $s + 1$?

Ključna ideja za izboljšavo osnovne metode

- Ne!
- Odmik s lahko povečamo za 4

  T: `...abcx.....` ==> `...abcx.....`
  P: `abcd` `abcd`

- Če bi se del `ab` ujemal z besedilom, `c` pa ne, bi odmik povečali za 3

  T: `...abx.....` ==> `...abx.....`
  P: `abcd` `abcd`

- Ali je povečanje odmika torej kar enako indeksu neujemanja (oz. m, če je $P = T[s+1 : s+m]$)?

Ključna ideja za izboljšavo osnovne metode

- Ne, žal ni tako enostavno
- Maksimalno varno povečanje odmika ni odvisno samo od položaja neujemanja, ampak tudi od znaka besedila na točki neujemanja in od strukture vzorca
- Analizirajmo dva primera ...

Ključna ideja za izboljšavo osnovne metode

- Naj bo r indeks prvega znaka v P, ki se pri trenutnem odmiku s razlikuje od istoležnega znaka v T
- $T[s+1:s+r] = P[1:r]$
- $T[s+r+1] \neq P[r+1]$
- $r = 0$, če imamo razliko že pri prvem znaku
- $r = m$, če imamo ujemanje med P in $T[s+1:s+m]$

- Za koliko lahko povečamo odmik (Δs) v odvisnosti od P, r in $T[s+r+1]$?

Ključna ideja za izboljšavo osnovne metode

- Naj bo $P = \texttt{abcd}$

r	$T[s+r+1]$	Δs
0	$\Sigma \setminus \{\texttt{a}\}$	1
1	$\Sigma \setminus \{\texttt{a}, \texttt{b}\}$	2
1	$\texttt{a}$	1
2	$\Sigma \setminus \{\texttt{a}, \texttt{c}\}$	3
2	$\texttt{a}$	2
3	$\Sigma \setminus \{\texttt{a}, \texttt{d}\}$	4
3	$\texttt{a}$	3
4	Σ	4

Ključna ideja za izboljšavo osnovne metode

- Naj bo $P = \mathtt{abac}$

r	$T[s+r+1]$	Δs
0	$\Sigma \setminus \{\mathtt{a}\}$	1
1	$\Sigma \setminus \{\mathtt{a}, \mathtt{b}\}$	2
1	$\mathtt{a}$	1
2	$\Sigma \setminus \{\mathtt{a}\}$	3
3	$\Sigma \setminus \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$	4
3	$\mathtt{a}$	3
3	$\mathtt{b}$	2
4	Σ	4

Ključna ideja za izboljšavo osnovne metode

- Naj bo $P =$ abac
- Recimo, da je $T[s+1 : s+4] =$ abab

T: ...abab..... ==> ...abab.....
P: abac abac

- Odmik lahko povečamo samo za 2, ker obstaja možnost, da je $T[s+3 : s+4] =$ ab začetek naslednje pojavitve P
- ab je pripona $T[: s+4]$ in hkrati predpona P

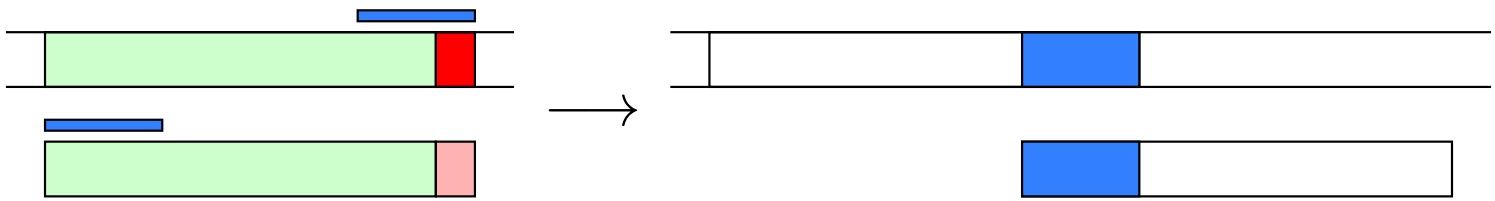
Priponska funkcija (glede na vzorec P)

- $\sigma(S)$ je dolžina najdaljše predpone vzorca P, ki je hkrati pripona niza S
- $\sigma(S) = \max\{k \mid P[:k] \sqsupseteq S\}$
- Naj bo $P = \texttt{abac}$

S	$\sigma(S)$
`xy`	0
`xya`	1
`xyab`	2
`xyaba`	3
`xyabac`	4

Priponska funkcija in največje varno povečanje odmika

- Naj bo $T[s+1:s+r] = P[1:r]$ in $T[s+r+1] \neq P[r+1]$
  - $r = 0$, če imamo razliko že pri prvem znaku
  - $r = m$, če imamo ujemanje med P in $T[s+1:s+m]$
- Potem je $\Delta s = r + 1 - \sigma(T[s+1:s+r+1])$



- Lahko bi pri vsakem neujemanju računali funkcijo σ, a tako bi vnovič prišli do $O(mn)$, lahko pa ...

Nekoliko drugačen pogled

- ... upoštevamo, da je maksimalni varni Δs odvisen samo od strukture vzorca in od znaka neujemanja besedila, ne pa od celotnega besedila
- Zadošča torej, da izračunamo Δs za vsak par (položaj neujemanja, znak neujemanja)
- Če trenutni položaj v vzorcu obravnavamo kot stanje, povečanje odmika pa kot spremembo stanja, celotna zgodba zadiši (zasmrdi?) po ...

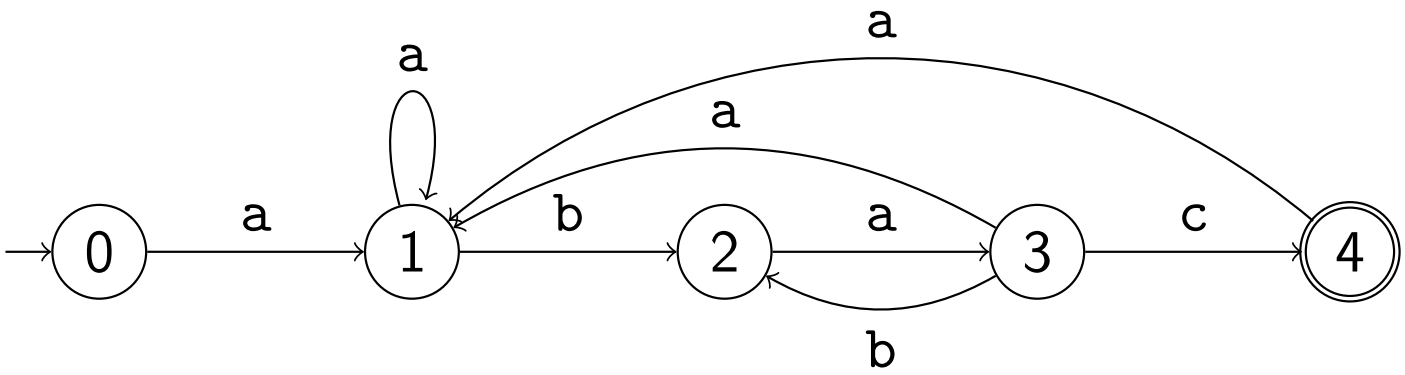
... deterministič̌nem končnem avtomatu, kakopak!

- $Q = \{0, 1, \ldots, m\}$
- $q_0 = 0$
- $F = \{m\}$
- $\delta(q, a) = \sigma(P[: q]a)$

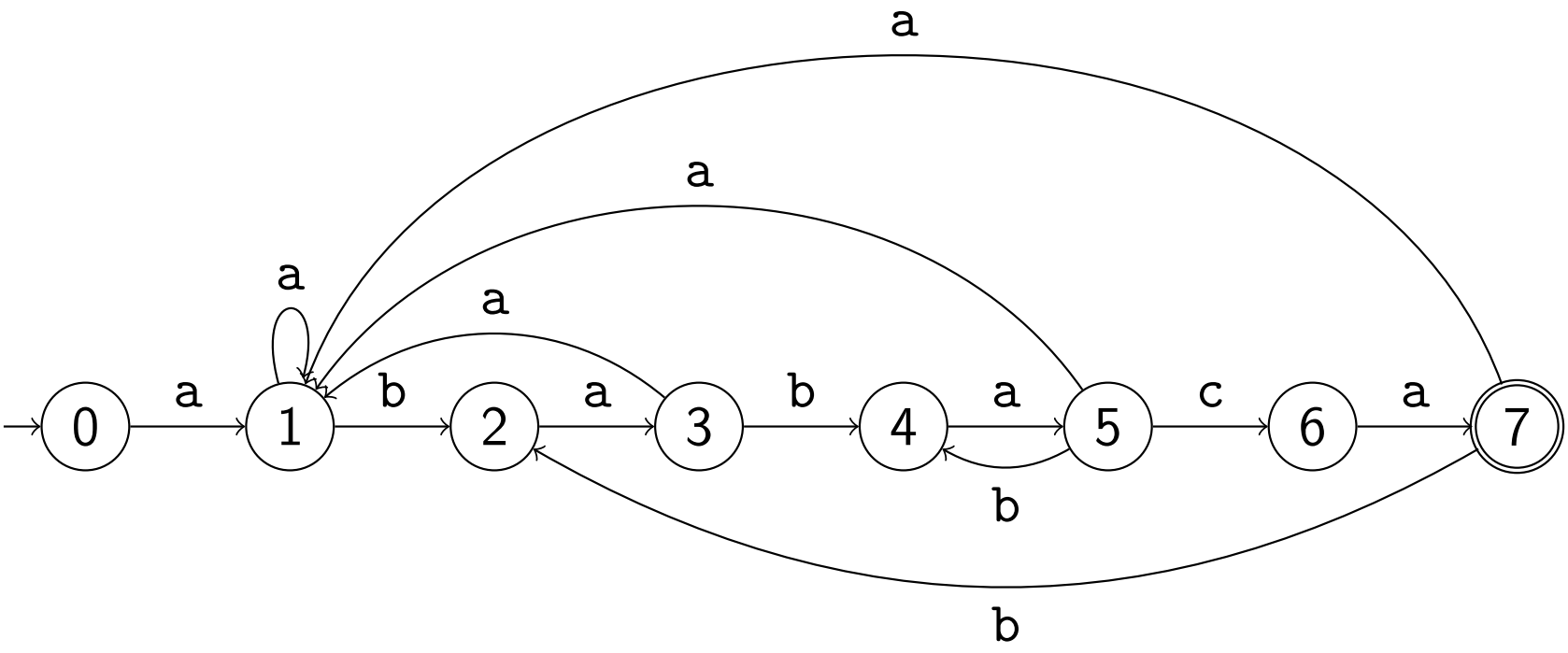
- Stanje nam pove število ujemajočih znakov besedila in vzorca
- Pričnemo v stanju 0, beremo znake besedila in potujemo po avtomatu
- Kadarkoli prispemo v končno stanje, najdemo pojavitev vzorca v besedilu

Primer 1: $P = \mathtt{abac}$

q	$\delta(q, \mathtt{a})$	$\delta(q, \mathtt{b})$	$\delta(q, \mathtt{c})$
0	1	0	0
1	1	2	0
2	3	0	0
3	1	2	4
4	1	0	0



Primer 2: $P =$ ababaca



Potovanje po DKA

- Recimo, da smo v stanju q
- To pomeni, da velja $T[s+1 : s+q] = P[: q]$
  - prvih q znakov vzorca se ujema z besedilom
- Naj bo $T[s+q+1] = a$
- Če je tudi $P[q+1] = a$, preidemo v stanje $q+1$
  - sedaj se prvih $q+1$ znakov vzorca ujema z besedilom
- Če je $P[q+1] \neq a$, preidemo v stanje $\sigma(P[: q]a) = q+1-\Delta s$, kjer je Δs največje varno povečanje odmika

Časovna zahtevnost

- Sprehod po besedilu zahteva le še $O(n)$ časa
- Gradnja avtomata je razmeroma zahtevna, a jo izvršimo samo enkrat
- $O(m^3|\Sigma|)$ po naivnem postopku
- Lahko kaj prihranimo tudi v fazi preodbdelave?