

Algoritmi i strukture podataka

Traženje uzorka u tekstu

Mirko Stojadinović

11. decembar 2015

1 Naivni (elementarni) algoritam

Napomena: U oba algoritma indeksi kreću od 1.

Elementarni algoritam traženja uzorka se zasniva na traženju uzorka u tekstu sleva udesno, i to tako što se poredi karakter po karakter (u parovima-karakter iz uzorka sa karakterom iz teksta). Prepostavka je da se svaki dokument predstavlja kao jedna niska karaktera (s) koja ima ukupno n karaktera (slova, cifre, razni specijalni znaci, ...). Uzorak ili obrazac (p) predstavlja nisku karaktera dužine m . Tipično je da je tekst mnogo duži od uzorka ($n >> m$). Elementarno traženje uzorka prolazi kroz s i kroz p i polazi od krajnje leve pozicije.

```
//Funkcija proverava da li string s sadrzi string p.  
//Vraca poziciju na kojoj p pocinje, odnosno -1 ukoliko ga nema  
int Sravni_niske(char s[], char p[]){  
    int i, j;  
  
    /* Proveravamo da li p pocinje na svakoj poziciji i */  
    for (i = 1; s[i]; i++)  
        /* Poredimo p sa s pocevsi od znaka p[1] sve dok ne nadjemo na razliku */  
        for (j = 0; s[i+j] == p[j+1]; j++)  
            /* Nismo naisli na razliku, a ispitali smo sve karaktere niske p */  
            if (p[j+2]=='\0')  
                return i;  
  
    /* Uzorak nije nadjen. */  
    return -1;  
}
```

Kod traženja uzorka u tekstu broj koraka se izračunava kao broj poređenja karatkera. U prethodnom algoritmu može biti najviše $O(n \cdot m)$ poređenja pa se ovaj algoritam smatra neefikasnim (uprkos tome što se ovaj najgori slučaj vrlo retko javlja).

2 KMP algoritam (Knuth-Morris-Pratt)

U prethodnom primeru postupak se zasniva na tome da se uzorak pomera za jednu poziciju u odnosu na tekst i za svaku poziciju u tekstu se prepostavlja da može da bude početak uzorka. Ništa nije unapred poznato. Međutim, ako se uzorak koji se sravnjuje prvo analizira, dobija se određeno saznanje o uzorku koje nam omogućava da se izbegne testiranje pretpostavke na svakoj poziciji teksta. Te pozicije su strateški odabrane i na njima postoji teoretska mogućnost da se uzorak pronađe. Broj poređenja u ovom algoritmu je $O(n + m)$ što je značajno efikasnije od elementarnog algoritma.

KMP algoritam se sastoji iz dva dela: prvi je popunjavanje niza h (tabele pomeranja uzorka) koja zavisi samo od uzorka, a drugi deo je sam algoritam traženja uzorka u tekstu.

Napomena: Obratiti pažnju da svi indeksi kreću od 1.

2.1 Prvi deo algoritma - popunjavanje tabele

```
Algoritam Pomaci
Ulaz: p (uzorak, string duzine m).
Izlaz: h (niz duzine m).
begin
    i := 1;
    j := 0;
    h[1] := 0; g[1] := 0;

    while i < m do
        while j > 0 and p[i] != p[j]
            j := h[j];

        i := i+1; j := j+1;
        g[i] := j;

        if p[i] == p[j]
            h[i] := h[j];
        else
            h[i] := j;
end
```

Smisao algoritma Pomaci: indeks i označava koju poziciju u tabeli sledeću popunjavamo. Ovaj indeks takođe označava koji deo niske trenutno razmatramo (od početka pa zaključno sa tim indeksom). Cilj je da se u svakom trenutku čuva najduži zajednički sufiks koji je jednak prefiksu (oba posmatramo u trenutno razmatranoj niski, tj. od indeksa 1 pa do i). Dužina tog sufiksa označena je indeksom j (na početku je 0). Kada se produži trenutno razmatrana niska (i se

uveća za 1), onda se posmatra da li se do tada nađeni zajednički prefiks i sufiks i dalje poklapaju. Postoje 2 opcije:

1. Poklapaju se - to znači da bi pri pretrazi u potencijalnom tekstu u slučaju neslaganja na poziciji i niske, automatski bio nemoguć i korišćenje sufiksa prethodno srađenog dela niske kao novog početka. Zato se prepisuje akcija koja je bila već izračunata za poziciju j ($h[i] := h[j]$).
2. Ne poklapaju se - do sada je (možda) bilo poklapanja zajedničkog prefiksa i sufiksa, ali od ove pozicije više nema poklapanja. U slučaju da u tekstu dođe do neslaganja na poziciji i niske (na nekoj poziciji teksta nije karakter $h[i]$, ne znamo da li je na toj poziciji možda karakter j niske ($h[i] \neq h[j]$)) se razlikuju. Zato nisku treba pomeriti da se sada porede j -ti elemnt niske i karakter teksta gde je ustanovljeno da nije $h[i]$. Dakle, u tabelu h se smešta j .

Smisao while petlje gde se vrši naredba $j := h[j]$ je da se u slučaju kada dođe do razlike na pozicijama i i j niske, automatski traži neki novi sufiks maksimalne dužine koji je jednak prefiksu.

2.2 Drugi deo algoritma - traženje uzorka

Algoritam KMP

Ulaz: s (tekst, string duzine n) i p (uzorak duzine m).

Izlaz: Start (indeks pocetka prvog podstringa s jednakog p ,
ako takav postoji, odnosno 0 u protivnom

begin

```

    Start := 0;
    i := 1;                      // pokazivac na uzorak
    j := 1;                      // pokazivac na tekst

    while i <= m and j <= n do  // dok ima nade da se nadje podstring jednak uzorku
        while i > 0 cand p[i] != s[j] do
            i := h[i];           // pomeranje uzorka udesno za i - h[i]
            i := i + 1; j := j + 1; // napredovanje u tekstu i u uzorku

        if i > m then
            Start := j - i + 1;   // pronadjen je podstring jednak uzorku
        end
    
```

Napomena: Niz g je niz koji se može koristiti kao i niz h , ali daje manje pomeraje, pa se zbog toga koristi niz h . U narednim zadacima nije neophodno konstruisati niz g .

2.3 Zadaci

1. Traži se prva pojava uzorka $p=abcabca$ u tekstu $s=babcabcaabca$.

- (a) Izračunati brojeve pomeranja uzorka - tabelu koja se koristi za algoritam KMP.
- (b) Izračunati broj pokušaja i broj poređenja karaktera?

Rešenje:

(a)

Uzorak p sa indeksima:

```
1 2 3 4 5 6 7 8 9 10
a b c a b c a c a b
```

Brojači u prvom algoritmu i nizovi g i h koji se formiraju:

```
i: 1 2 3 4 5 6 7 8 9 10
j: 0 1 0 1 0 1 2 3 4 5 1 0 1 2
```

```
i: 1 2 3 4 5 6 7 8 9 10
h[i]: 0 1 1 0 1 1 0 5 0 1
i-h[i]: 1 1 2 4 4 5 7 3 9 9 (pomeraj)
g[i]: 0 1 1 1 2 3 4 5 1 2
```

(b) Tekst s sa indeksima zajedno sa pokušajima:

```
indeks: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
s: b a b c b a b c a b c a
pokusaj 1: a
pokusaj 2: a b c a
pokusaj 3: a b c a b c a c
pokusaj 4: a b c a b
pokusaj 5: a b c a
```

pokusaj 1 pa pomeraj za $1-h[1] = 1$ poziciju;
 pokusaj 2 pa pomeraj za $4-h[4] = 4$ pozicije;
 pokusaj 3 pa pomeraj za $8-h[8] = 3$ pozicije;
 pokusaj 4 pa pomeraj za $5-h[5] = 4$ pozicije;

Brojači u algoritmu KMP:

```
i: 1 0 1 2 3 4 0 1 2 3 4 5 6 7 8 5 1 2 3 4
j: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

Pokušaja: 5 (za 1 više od broja pomeraja).

Poređenja karaktera: 18 (jedan način da se ovo vidi je broj vrednosti koje uzima brojač i ne računajući nule, drugi je iz gornjeg prikaza pokušaja).

- 2.** Traži se prva pojava uzorka $p=atcacatcatca$ u tekstu $s=gatcgatcacatcatcacaaaaaa$.

(a) Izračunati brojeve pomeranja uzorka - tabelu koja se koristi za algoritam KMP.

(b) Izračunati broj pokušaja i broj poređenja karaktera?

Rešenje:

(a)

```
i: 1 2 3 4 5 6 7 8 9 10 11 12  
h[i]: 0 1 1 0 2 0 1 1 0 5 1 0
```

(b)

```
indeks: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  
s: h a t c g a t c a c a t c a t c a c g a a a a a  
p1: a  
p2: a t c a  
p3: a t c a c a t c a t c a
```

Uzorak pronadjen, Start = 6. Pokušaja: 3, poređenja karaktera: 17.

Napomena: U narednim zadacima dato je samo rešenje. Pri radu na kolo-
kvijumu/ispitu potrebno je detaljno navesti postupak (obrazloženje može biti
najmanje dužine kao u zadatku 2).

3. Traži se prva pojava uzorka $p=abacab$ u tekstu $s=acabcacb$.

(a) Izračunati brojeve pomeranja uzorka - tabelu koja se koristi za algoritam KMP.

(b) Izračunati broj pokušaja i broj poređenja karaktera?

Rešenje:

(a)

```
i: 1 2 3 4 5 6  
h[i]: 0 1 0 2 0 1
```

(b) Pokušaja: 6, poređenja karaktera: 10.

4. Traži se prva pojava uzorka $p=prepreden$ u tekstu $s=kadsuprelaziliprekoprekenasmejaseprepredeno$.

(a) Izračunati brojeve pomeranja uzorka - tabelu koja se koristi za algoritam KMP.

(b) Izračunati broj pokušaja i broj poređenja karaktera?

Rešenje:

(a)

i: 1 2 3 4 5 6 7 8 9
h[i]: 0 1 1 0 1 1 4 1 1

(b) Pokušaja: 26, poređenja karaktera: 46.

5. Traži se prva pojava uzorka p=abacab u tekstu s=aacabadababcabac.

(a) Izračunati brojeve pomeranja uzorka - tabelu koja se koristi za algoritam KMP.

(b) Izračunati broj pokušaja i broj poređenja karaktera?

Rešenje: (b)

0. aacabadababcabac
1. ab
2. ab
3. a
4. abac
5. ab
6. a
7. abac
8. aba
9. abac

Pokušaja: 9, poređenja karaktera: 21.

6. Traži se prva pojava uzorka p=abcabd u tekstu s=abcabbabbcaab.

(a) Izračunati brojeve pomeranja uzorka - tabelu koja se koristi za algoritam KMP.

(b) Izračunati broj pokušaja i broj poređenja karaktera?

Rešenje: Pokušaja: 7, poređenja karaktera: 19.