Vyhľadávanie reťazcov v textoch I

Problém

Text:
Reťazec:
Úloha: nájsť všetky výskyty daného reťazca v danom texte.

Použitie

- textové editory,
- internetové vyhľadávače,
- filtre na slová v správach pre spravodajské služby,
- prehľadávanie DNA
- ...

Označenia

Σ	abeceda
$T[1..n]$	text dĺžky n znakov, $T[i]$ $in\Sigma$
$P[1..m]$	vyhľadávaný reťazec dĺžky m znakov, $P[i] \in \Sigma$, $m \leq n$

Ak $T[s+1..s+m] = P[1..m]$, hovoríme o výskyte reťazca P v texte T na pozícii $s+1$, resp. o výskyte reťazca s posunutím s.

Označenia

Σ	abeceda
Σ^*	množina všetkých konečných reťazcov zo znakov abecedy Σ
ε	prázdny reťazec
xy	zreťazenie reťazcov x, y
$\|x\|$	dĺžka reťazca x
	Platí $\|xy\| = \|x\| + \|y\|$.

Prefix reťazca

Definition

Hovoríme, že w je prefixom reťazca x, označujeme $w \sqsubset x$, ak $|w| \leq |x|$ a $x = wy$ pre nejaký $y \in \Sigma^*$.

Example

Reťazec `po` je prefixom reťazca `počítač`.

Suffix reťazca

Definition

Hovoríme, že w je suffixom reťazca x, označujeme $w \sqsupset x$, ak $|w| \leq |x|$ a $x = yw$ pre nejaký $y \in \Sigma^*$.

Example

Reťazec `lýza` je suffixom reťazca `analýza`.

Prefix a suffix reťazca

Prázdny reťazec ε je prefixom i suffixom každého reťazca.

$\sqsubset, \sqsupset$ sú tranzitívne relácie.

Veta o prekrývajúcom suffixe

Theorem

Nech $x, y, z \in \Sigma^*$ *sú také reťazce, že* $x \sqsupset z$ *a* $y \sqsupset z$. *Potom:*

- *ak* $|x| \leq |y|$, *tak* $x \sqsupset y$,
- *ak* $|x| \geq |y|$, *tak* $y \sqsupset x$,
- *ak* $|x| = |y|$, *tak* $x = y$.

Známe algoritmy na vyhľadávanie reťazcov v textoch

Algoritmus	Zlož.prípr.fázy	Zlož.vyhľadávania
Naivný	0	$\mathcal{O}((n-m+1)m)$
Rabin-Kapr	$\Theta(m)$	$\mathcal{O}((n-m+1)m)$
Konečný automat	$Theta(m\|\Sigma\|)$	$\Theta(n)$
Knuth-Morris-Pratt	$\Theta(m)$	$\Theta(n)$

Naivný algoritmus

```
Naive-string-matching(T, P)
{
   n = length(T);
   m = length(P);
   for (s = 0; s <= n − m; s + +) {
      zhoda=1;
      for (i = 1; i <= m; i + +)
         if (P[i]! = T[s + i]) zhoda=0;
      if (zhoda)
         print "Výskyt reťazca s posunutím ", s
   }
```

Naivný algoritmus

Výpočtová zložitosť: $\mathcal{O}((n - m + 1)m)$.

Prečo?

Aký je najhorší prípad?

Aký je najlepší prípad?

V čom tkvie neefektivita naivného algoritmu?

Rabin-Karp algoritmus

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

$P[1..m] \rightarrow p,\ p \in N$.

$T[s+1..s+m] \rightarrow t_s,\ t_s \in N,\ s = 0, 1, 2, \ldots, n-m$.

Zrejme:
$t_s = p \Leftrightarrow T[s+1..s+m] = P[1..m]$.

Rabin-Karp algoritmus

Výpočet p:
pomocou Hornerovej schémy.

$$p = P[m]+10(P[m-1]+10(P[m-2]+\cdots+10(P[2]+10P[1])\ldots)).$$

Výpočtová zložitosť: $\Theta(m)$.

Podobne výpočet t_0 :
$t_0 =$
$$T[m]+10(T[m-1]+10(T[m-2]+\cdots+10(T[2]+10T[1])\ldots)).$$

Rabin-Karp algoritmus

Výpočet t_{s+1} pomocou t_s:
$t_{s+1} = 10(t_s - 10^{m-1} T[s+1]) + T[s+m+1]$.

$t_1, t_2, \ldots, t_{n-m}$ možno počítať počas samotného hľadania.

Rabin-Karp algoritmus 1

```
Rabin-Karp-matching1(T, P)
{
    n = length(T);
    m = length(P);
    // prípravná fáza
    p = 0;
    t_0 = 0;
    for (s = 1; s <= m; ++s) {
        p = 10p + P[s];
        t_0 = 10t_0 + T[s];
    }
```

Rabin-Karp algoritmus 1 (pokračovanie)

```
    // vyhľadávanie
    for (s = 0; s <= n − m; s + +) {
        if (p == t_s)
            printf("Výskyt reťazca s posunutím %d", s);
        if (s < n − m)
            t_{s+1} = 10(t_s − 10^{m−1} T[s + 1]) + T[s + m + 1];
    }
}
```

Rabin-Karp algoritmus

Teoreticky očakávaná výpočtová zložitosť:
prípravná fáza: $\Theta(m)$,
hľadanie: $\Theta(n - m + 1)$.

Prax:
p, t_s môžu byť veľké čísla, t.j. predpoklad, že operácie s nimi možno vykonať v jednotkovom čase nie je správny.

Rabin-Karp algoritmus

Riešenie:
počítať p, t_s modulo vhodný modul q.

Problém tohto riešenia:
$t_s \equiv p \pmod{q}$ neimplikuje $t_s = p$.
(V takomto prípade treba porovnať znaky reťazca so znakmi textu pri danom posunutí.)
Avšak platí:
$t_s \neq p \pmod{q} \Rightarrow t_s \neq p$.

Rabin-Karp algoritmus

Vo všeobecnosti:
d-árna abeceda $\{0, 1, 2, \ldots, d-1\}$.
Volíme q tak, aby sa dq zmestilo do registra PC.
$t_{s+1} = (d(t_s - hT[s+1]) + T[s+m+1]) \mod q$.
$h \equiv d^{m-1} \pmod q$.

Rabin-Karp algoritmus 2

Rabin-Karp-matching2(T, P, d, q)
{
  $n = \text{length}(T)$;
  $m = \text{length}(P)$;
  $h = d^{m-1} \bmod q$;
  $p = 0$;
  $t_0 = 0$;
  // prípravná fáza
  for ($i = 1; i <= m; ++i$) {
    $p = (dp + P[i]) \bmod q$;
    $t_0 = (dt_0 + T[i]) \bmod q$;
  }

Rabin-Karp algoritmus 2 (pokračovanie)

```
    // vyhľadávanie
    for (s = 0; s <= n − m; s + +) {
        if (p == t_s)
            if (P[1..m] == T[s + 1..s + m + 1])
                printf("Výskyt reťazca s posunutím %d", s);
        if (s < n − m)
            t_{s+1} = (d(t_s − hT[s + 1]) + T[s + m + 1]) mod q;
    }
}
```

Rabin-Karp algoritmus 2

Teoreticky očakávaná výpočtová zložitosť:
prípravná fáza: $\Theta(m)$,
hľadanie (najhorší prípad): $\Theta(n - m + 1)m$.

Prax:
očakáva sa len malý počet (c) výskytov hľadaného reťazca
hľadanie: $\mathcal{O}((n - m + 1) + cm)$

Príklad - Rabin-Karp algoritmus 2

$q = 11$

$T = 3141592653589793$

$P = 26,\ p = 26 \bmod 11 = 4$

s	0	1	2	3	4	5	6	7
$T[s+1, s+2]$	31	14	41	15	59	92	26	65
t_s	9	3	8	4	4	4	4	10

s	8	9	10	11	12	13	14	
$T[s+1, s+2]$	53	35	58	89	97	79	93	
t_s	9	2	3	1	9	2	5	