Vyhľadávanie podreťazcov v reťazci (String matching)

1. Naive method (Priama metóda)

Pre každý z možných posunov sa pokúsi vyhľadať reťazec P.
Zložitosť: $O((n-m+1)m)$
Najhorší prípad: vyhľadávanie podreťazca a^m v a^n, $m < n$
T - reťazec v ktorom sa vyhľadáva
P - hľadáný podreťazec
$|\cdot|$ - dĺžka reťazca
$P[a..b] = a.$ - $b.$ prvok v reťazci P

```
n := |T|
m := |P|

for s:=0 to n-m do
  if P[1..m] = T[s+1..s+m] then
    print "Podreťazec sa našiel pri posunutí", s
```

2. Rabin-Karp algorithm (Rabin-Karpov algoritmus)

Použitá abeceda: $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$
Toto nám umožní brať podreťazce T a samotné P ako číslo v desiatkovej sústave. Podmienka $P[1..m] = T[s+1..s+m]$ sa tak prevedie na porovnávanie dvoch čísel. Hodnota sa vypočítava z Hornerovej schémy. Pre každý posun v T stačí vynechať prvú cifru, celé vynásobiť 10 a pripočítať nasledujúcu cifu.

		h						h'		
a_k	a_{k+1}	$\dots$	a_{k+m-1}	a_{k+m}	$\rightarrow$	a_k	a_{k+1}	$\dots$	a_{k+m-1}	a_{k+m}

$h = 10^{m-1}a_k + 10^{m-2}a_{k-1} + \ldots + 10a_{k+m-2} + a_{k+m-1} \rightarrow$
$h' = 10 * (h - 10^{m-1}a_k) + a_{k+m}$

Pre dlhé podreťazce P môžeme dostávať veľmi veľké čísla. (Efektívne by bolo dobré dostávať čísla, ktoré sú menšie ako dĺžka slova na procesore - $0 - 2^{32} - 1$, (32bit), $0 - 2^{64} - 1$ (64bit).)

Preto sa to rieši pomocou výpočtu modulo q. Najprv sa porovnajú získané čísla modulo q a v prípade rovnosti sa ešte porovnajú člen, po člene. (Pretože, v prípade nízkej hodnoty modula sa kľudne môže stať, že dva rôzne podreťazce zodpovedajú rovnakému číslu modulo q.)

- Podobne ako pri základe 10, možno predošlé urobiť pre iný číselný základ d.
  Zložitosť:
  Preprocessing (príprava) - $\Theta(m)$
  Výpočet (Computation) - $O((n-m+1)m)$
  Najhorší prípad: vyhľadávanie podreťazca a^m v a^n, $m < n$
  - Oproti Naive dáva lepšie priemerné časy.

```
Rabin-Karp Matcher(T, P, d, q)
n := |T|
m := |P|
h := d^(m-1) mod q
p := 0
t_0 := 0
for i:=1 to m do {preprocessing}
    p := (d * p + P[i]) mod q
    t_0 := (d * t_0 + T[i]) mod q
for s:=0 to n - m do {computation}
    if p = t_s then
        if P[1..m] = T[s + 1..s + m] then
            print "Podreťacec sa našiel s posunom", s
    if s < n - m then
        t_{s+1} := (d * (t_s - T[s + 1]h) + T[s + m + 1]) mod q
```

3. Konečný automat

Def.: *Konečný automat* (*finite automaton*) je usporiadaná 5-tica $(Q, q_0, A, \Sigma, \delta)$, kde

- Q je konečná množina stavov
- q_0 počiatočný stav
- A množina koncových stavov (akceptujúce)
- Σ je použitá abeceda
- δ je tzv. *prechodová funkcia* z $Q \times \Sigma$ do Q.

Rozšírenie δ funkcie - $\delta^* : Q \times \Sigma^* \to Q$ je definované induktívne:
$\delta^*(q, \epsilon) = q$
$\delta^*(q, wa) = \delta(\delta^*(q, w), a)$
Final-state function - vracia stav automatu po spracovaní nejakého slova

Suffix funkcia pre P, $|P| = m$ je $\sigma : \Sigma^* \to \{0, 1, \ldots, m\}$ definovaná ako

$$\sigma(x) = \max\{k : P_k \sqsupset x\},$$

kde $u \sqsupset v$ znamená, že u je sufixom v a $P_k = P[1..k]$.
Definícia automatu: pre P, $|P| = m$
$Q = \{0, 1, \ldots, m\}$, $q_0 = 0$, $A = \{m\}$, $\delta(q, a) = \sigma(P_q a)$.

Vždy, keď sa počas simulácie vstupného slova T na automate dostaneme do stavu m, našiel sa podvýraz P a jeho posun je rovný o m menej ako je aktuálna pozícia v reťazci.

Platia nasledujúce vety:
V (suffix-function inequality): Pre každý reťazec x a znak a platí: $\sigma(xa) \leq \sigma(x) + 1$.

V (suffix-function recursion lemma): Pre každý reťazec x a znak a, ak $q = \sigma(x)$, tak $\sigma(xa) = \sigma(P_q a)$.

VÝPOČET PRECHODOVEJ FUNKCIE (P, Σ)
$m := |P|$
for $q := 0$ to m do
    for each symbol $a \in \Sigma$ do
        $k := \min(m+1, q+2)$
        repeat $k := k - 1$
            until $P_k \sqsupset P_q a$
        $\delta(q, a) := k$
return δ

Zložitosť tejto funkcie je $O(m^3|\Sigma|)$. Dá sa zlepšiť na $O(m|\Sigma|)$.
Zložitosť samotného výpočtu je $\Theta(n)$.

4. KMP (Knuth, Morris, Pratt)

Prefixová funkcia (Prefix function) π pre P, $|P| = m$:
$\pi : \{1, 2, \ldots, m\} \to \{0, 1, \ldots, m-1\}$ $\pi(q) = \max\{k : k < q, P_k \sqsupset P_q\}$.

KMP-MATCHER (T, P)
$n := |T|$
$m := |P|$
$\pi :=$ COMPUTE-PREFIX-FUNCTION (P)
$q := 0$
for $i := 1$ to n do
    while $q > 0$ and $P[q+1] \neq T[i]$ do
        $q := \pi(q)$
    if $P[q+1] = T[i]$ then
        $q := q + 1$
    if $q = m$ then
        print "Podvýraz sa vyskytol s posunom", $i - m$
        $q := \pi(q)$

COMPUTE-PREFIX-FUNCTION (P)
$m := |P|$
$\pi(1) := 0$
$k := 0$
for $q := 2$ to m do
    while $k > 0$ and $P[k+1] \neq P[q]$ do
        $k := \pi(k)$
    if $P[k+1] = P[q]$ then
        $k := k + 1$
    $\pi(q) := k$
return π

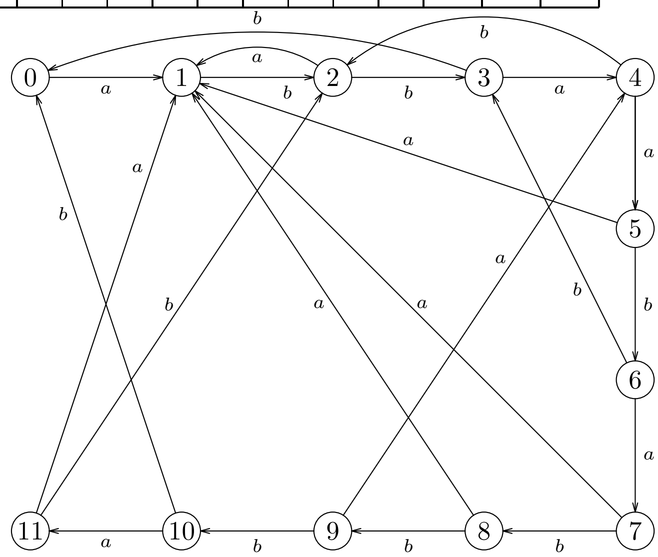
Typy úloh:
1. Vypočítajte π pre P a Σ
2. Zostavte automat na vyhľadávanie podreťazca P a znázornite ho.

Riešené príklady

- Zostavte automat na vyhľadávanie reťazca $P = abbaababbba$ a znázornite ho.
  $|P| = 11$
  Stavy: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11$, $q_0 = 0$ a $A = \{11\}$

  - $\delta(0, a) = \sigma(P_0 a) = \sigma(\mathbf{a}) = 1$
  - $\delta(0, b) = \sigma(P_0 b) = \sigma(b) = 0$
  - $\delta(1, a) = \sigma(P_1 a) = \sigma(a\mathbf{a}) = 1$
  - $\delta(1, b) = \sigma(P_1 b) = \sigma(\mathbf{ab}) = 2$
  - $\delta(2, a) = \sigma(P_2 a) = \sigma(ab\mathbf{a}) = 1$
  - $\delta(2, b) = \sigma(P_2 b) = \sigma(\mathbf{abb}) = 3$
  - $\delta(3, a) = \sigma(P_3 a) = \sigma(\mathbf{abba}) = 4$
  - $\delta(3, b) = \sigma(P_3 b) = \sigma(abbb) = 0$
  - $\delta(4, a) = \sigma(P_4 a) = \sigma(\mathbf{abbaa}) = 5$
  - $\delta(4, b) = \sigma(P_4 b) = \sigma(abb\mathbf{ab}) = 2$
  - $\delta(5, a) = \sigma(P_5 a) = \sigma(abbaa\mathbf{a}) = 1$
  - $\delta(5, b) = \sigma(P_5 b) = \sigma(\mathbf{abbaab}) = 6$
  - $\delta(6, a) = \sigma(P_6 a) = \sigma(\mathbf{abbaaba}) = 7$
  - $\delta(6, b) = \sigma(P_6 b) = \sigma(abba\mathbf{abb}) = 3$
  - $\delta(7, a) = \sigma(P_7 a) = \sigma(abbaaba\mathbf{a}) = 1$
  - $\delta(7, b) = \sigma(P_7 b) = \sigma(\mathbf{abbaabab}) = 8$
  - $\delta(8, a) = \sigma(P_8 a) = \sigma(abbaabab\mathbf{a}) = 1$
  - $\delta(8, b) = \sigma(P_8 b) = \sigma(\mathbf{abbaababb}) = 9$
  - $\delta(9, a) = \sigma(P_9 a) = \sigma(abbaab\mathbf{abba}) = 4$
  - $\delta(9, b) = \sigma(P_9 b) = \sigma(\mathbf{abbaababbb}) = 10$
  - $\delta(10, a) = \sigma(P_{10} a) = \sigma(\mathbf{abbaababbba}) = 11$
  - $\delta(10, b) = \sigma(P_{10} b) = \sigma(abbaababbbb) = 0$
  - $\delta(11, a) = \sigma(P_{11} a) = \sigma(abbaababbba\mathbf{a}) = 1$
  - $\delta(11, b) = \sigma(P_{11} b) = \sigma(abbaababbb\mathbf{ab}) = 2$

	0	1	2	3	4	5	6	7	8	9	10	11
a	1	1	1	4	5	1	7	1	1	4	11	1
b	0	2	3	0	2	6	3	8	9	10	0	2



- Určte prefixovú funkciu (z KMP algoritmu) pre reťazec P z predošlej úlohy.

	1	2	3	4	5	6	7	8	9	10	11
π	0	0	0	1	1	2	1	2	3	0	1

Napr. $\pi(9) = 3$:
$P[1..9]$ - *abbaababb*
$P[1..8]$ - *abbaabab* - nie je suffix v $P[1..9]$
$P[1..7]$ - *abbaaba* - nie je suffix v $P[1..9]$
$P[1..6]$ - *abbaab* - nie je suffix v $P[1..9]$
$P[1..5]$ - *abbaa* - nie je suffix v $P[1..9]$
$P[1..4]$ - *abba* - nie je suffix v $P[1..9]$
$P[1..3]$ - *abb* - je suffix v $P[1..9]$