Поиск подстроки

Обозначения:

$T[0:m]$ - текст $T[i:j]$ - подстрока

$P[0:n]$ - образец

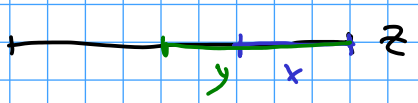
x, y - строки

$x \sqsubset y$ - x - префикс y

$x \sqsupset y$ - x - суффикс y

Утв:

$x \sqsupset z,\ y \sqsupset z,\ |x| \le |y| \Rightarrow x \sqsupset y$



Задача о поиске подстроки

Вход: $T[0:m]$, $P[0:n]$

Найти i (наименьшее):

$T[i, i+n] = P$

Наивный алгоритм

```
for i = 0 to m - n                    O((m-n)·n) = O(m·n)
    for j = 0 to n                    O(n)
        if P[j] ≠ T[i+j]
            continue
        return i
```

Алгоритм Карпа - Рабина

$T[0:m]$, $P[0:n]$

Для всех i от 0 до $m-n$

вычислим $hash(T[i, i+n]) = h_i$

// $m-n$ хешей

```
for i = 0 to m - n:
    if h_i = hash(P):
        if P = T[i, i+n]        // наивно O(n)
            return i
```

Полиномиальное хеширование

$$S[0:n]$$

$$hash(S) = q^n S[0] + q^{n-1} S[1] + \ldots + q^1 S[n-1] + S[n] \mod N$$

$$h_i = hash(T[i, i+n])$$

$$h_{i+1} = (h_i - q^n T[i]) \cdot q + T[i+n+1] \mod N$$

⇒ мы можем вычислить все h_i

за $c_1 n + (m-n) \cdot c_2 = O(m)$

Оценим. В предположении гипотезы равномерного хеширования вероятность коллизии $\frac{1}{N} \leq \frac{1}{m}$

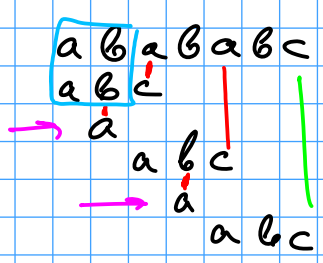
$$O(m + m + m/N \cdot n) = O(m+n)$$

($m/N \leq 1$)

Комбинаторный подход

T = abababc

P = abc



Z - функция

S = abababc $z(S)[i] = j \iff$

$S[i, i+j] \sqsubset S$, j - max

$z(S) = 0040200$

Поиск подстроки:

$T[0,m]$, $P[0,n]$

$z(P \# T)$

$z(abc \# ababababc)$

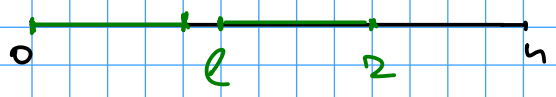
$000020203 00$

Вычисление:

Наивно за n^2

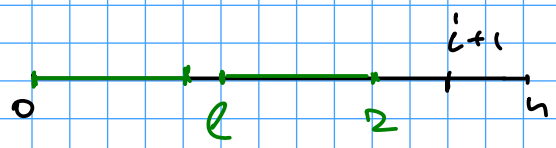
Худший случай: aaaaaaaaab

Вычисление за $O(n)$:

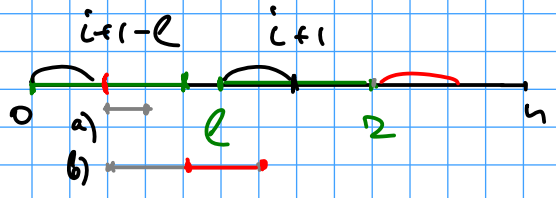


$z[i+1]$ - ? Все $z[i]$ вычислены

l, r: $S[l, r]$ - правейший префикс который мы уже нашли

① 

Вычислим $z[i+1]$ наивным образом

② 

a) $z[i+1] = \min(z[i+1-l], n-i)$

б) $z[i+1] = \min(z[i+1-l], r-i)$

проверяем сравнением за r

Пример: abc#ababaabc (l 2 l 2 l 2)

0 0 0 0 2 0 2 0 3 0 0

Z-Function (S[0,n]):

Z(0,n)

l, r = 0

for i = 1 to n

if i ≤ r:

Z[i] = min (r − i + 1, Z[i − l])

while Z[i] + i ≤ n and S[Z[i]] = S[Z[i] + i]

++ Z[i]

if i + Z[i] ≥ r

l = i, r = i + Z[i]

O(n)

В итоге: поиск подстроки за O(m + n)

(находим все вхождения)

Алгоритм Кнута – Морриса – Пратта

= Префикс - функция π

T

abcaabcaabc

P = abcaabcaabd

π(i)

сдвигаем на i − π(i)

P_1 = P[0]

P_2 = P[0,1]

⋮

?

все префиксы P

P	a b c a a b c a a b d
π	0 0 0 1 2 3 4 5 0

$\pi(i) = \max_{k<i} k : P_k \sqsupset P_i$

```
KMP(T, P)
π ← PrefixFunction(P)
k = 0  // совпало символов
for i = 0 to m:
    while k > 0 and P[k+1] ≠ T[i]
        k = π(k)
    if P[k+1] = T[i]
        k = k+1
    if k = n
        return i - n + 1                    O(m)
```

```
PrefixFunction(P):
k = 0
π[i] = 0
for i = 1 to n
    while k > 0 && P[k+1] ≠ P[i]
        k = π(k)
    if P[k+1] = P[i]
        k = k+1
        π[i] = k+1
```