

Классические алгоритмы  
поиска образца в строке  
Дискретный анализ 2012/13

Андрей Калинин, Татьяна Романова

12 ноября 2012 г.

## Алгоритм Кнута-Морриса-Пратта

Классический вариант алгоритма

## Алгоритм Ахо-Корасик

Задача множественного поиска  
Связи неудач  
Полный алгоритм поиска

## Поиск с джокером

# Литература

- ▶ Дэн Гасфилд, «Строки дерева и последовательности в алгоритмах: Информатика и вычислительная биология», 2003. Глава 3, «Более глубокий взгляд», стр. 46–94.
- ▶ Билл Смит, «Методы и алгоритмы вычислений на строках», 2006.

# Раздел

## Алгоритм Кнута-Морриса-Пратта Классический вариант алгоритма

Алгоритм Ахо-Корасик  
Задача множественного поиска  
Связи неудач  
Полный алгоритм поиска

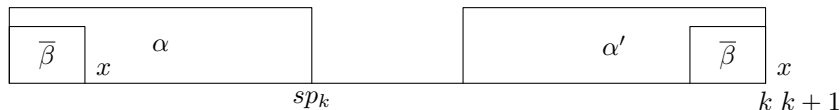
Поиск с джокером

# Отличия

- ▶ Не использует  $Z$ -блоки для расчёта значений  $sp_i$ .
- ▶ Строит последовательно значения  $sp_k$  по рассчитанным значениям от 1 до  $k - 1$ .
- ▶ Легко обобщается на случай большого количества образцов (алгоритм Ахо-Корасик)

# Обозначения

- ▶  $\alpha$  — префикс длиной  $sp_k$ ,  $\alpha'$  — суффикс.
- ▶  $x$  —  $k + 1$ -й символ.
- ▶  $\beta = \bar{\beta}x$  — префикс длины  $sp_{k+1}$ , нахождение  $\bar{\beta}$  эквивалентно вычислению  $sp_{k+1}$ .



## Связь $sp_{k+1}$ и $sp_k$

### Теорема

$$\forall k : sp_{k+1} \leq sp_k + 1;$$

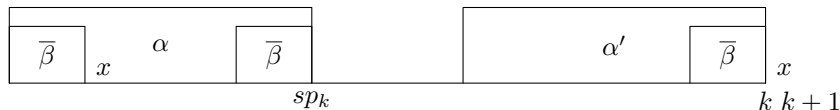
$$sp_{k+1} = sp_k + 1 \iff P(sp_k + 1) = P(k + 1)$$

### Доказательство.

Если  $sp_{k+1} > sp_k + 1$ , то  $|\beta| > |\alpha|$ , причём  $\beta$  является одновременно префиксом  $P$  и суффиксом  $P[1..k]$ , что противоречит выбору  $sp_k$ . □

## Общий случай

Если  $P(k+1) \neq P(sp_k+1)$ , то  $sp_{k+1} \leq sp_k$  и  $\bar{\beta}$  — собственный префикс и суффикс  $\alpha$ .





## Расчёт $sp_{k+1}$

```
3    $x \leftarrow P(k + 1)$   
4    $v \leftarrow sp_k$   
5   while  $P(v + 1) \neq x$  and  $v \neq 0$   
6        $v \leftarrow sp_v$   
7   if  $P(v + 1) = x$   
8        $sp_{k+1} \leftarrow v + 1$   
9   else  $sp_{k+1} \leftarrow 0$ 
```

## Расчёт $sp_{k+1}$

```
3    $x \leftarrow P(k + 1)$ 
4    $v \leftarrow sp_k$ 
5   while  $P(v + 1) \neq x$  and  $v \neq 0$ 
6        $v \leftarrow sp_v$ 
7   if  $P(v + 1) = x$ 
8        $sp_{k+1} \leftarrow v + 1$ 
9   else  $sp_{k+1} \leftarrow 0$ 
```

abxabqabxab**r**abxabqabxab**x**



## Расчёт $sp_{k+1}$

```
3    $x \leftarrow P(k + 1)$   
4    $v \leftarrow sp_k$   
5   while  $P(v + 1) \neq x$  and  $v \neq 0$   
6        $v \leftarrow sp_v$   
7   if  $P(v + 1) = x$   
8        $sp_{k+1} \leftarrow v + 1$   
9   else  $sp_{k+1} \leftarrow 0$ 
```

abxabqabxabrabxabqabxabx

## Расчёт $sp_{k+1}$

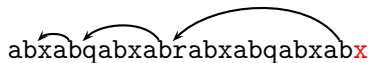
```
3    $x \leftarrow P(k + 1)$   
4    $v \leftarrow sp_k$   
5   while  $P(v + 1) \neq x$  and  $v \neq 0$   
6        $v \leftarrow sp_v$   
7   if  $P(v + 1) = x$   
8        $sp_{k+1} \leftarrow v + 1$   
9   else  $sp_{k+1} \leftarrow 0$ 
```

abxabqabxabrabxabqabxabx

## Расчёт $sp_{k+1}$

```
1  $sp_1 \leftarrow 0$   
2 for  $k \leftarrow 1$  to  $n - 1$   
3    $x \leftarrow P(k + 1)$   
4    $v \leftarrow sp_k$   
5   while  $P(v + 1) \neq x$  and  $v \neq 0$   
6      $v \leftarrow sp_v$   
7   if  $P(v + 1) = x$   
8      $sp_{k+1} \leftarrow v + 1$   
9   else  $sp_{k+1} \leftarrow 0$ 
```

abxabqabxabrabxabqabxab $x$



# Линейность

## Теорема

Алгоритм находит все значения  $sp_i(P)$  за время  $O(n)$ , где  $n$  — длина  $P$ .

## Доказательство.

Время работы пропорционально числу присваиваний  $v$ :

- ▶ Один раз на каждом шаге,  $v \leftarrow sp_k$ .
- ▶ Несколько раз уменьшается внутри **while** .
- ▶ Число увеличений ограничено  $n - 1$ , следовательно и число уменьшений ограничено  $n - 1$
- ▶ Тогда общее число присваиваний  $v$  ограничено сверху  $2(n - 1) = O(n)$



## Вычисление $sp'_i$

```
1  $sp'_1 \leftarrow 0$   
2 for  $i \leftarrow 2$  to  $n$   
3      $v \leftarrow sp_i$   
4     if  $P(v + 1) \neq P(i + 1)$   
5          $sp'_i \leftarrow v$   
6     else  $sp'_i \leftarrow sp'_v$ 
```

# Раздел

Алгоритм Кнута-Морриса-Пратта  
Классический вариант алгоритма

Алгоритм Ахо-Корасик  
Задача множественного поиска  
Связи неудач  
Полный алгоритм поиска

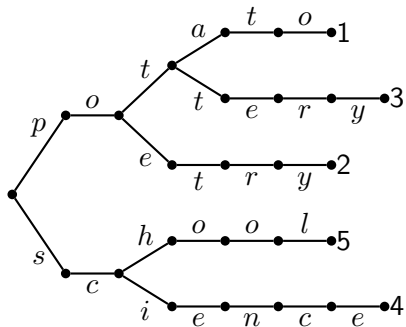
Поиск с джокером



## Формулировка

- ▶ Задано множество образцов,  $\mathbb{P} = \{P_1, P_2, \dots, P_z\}$ .
- ▶ Необходимо найти все вхождения в  $T$  любых образцов из  $\mathbb{P}$ .
- ▶ Теперь  $n = \sum_{k=1}^z |P_k|$
- ▶ Предыдущие алгоритмы могут решить за  $O(n + zm)$ .
- ▶ Возможно решение за  $O(n + m + k)$ , где  $k$  — количество вхождений в  $T$  образцов из  $\mathbb{P}$

## Дерево ключей $\mathbb{K}$



Дерево ключей  $\mathbb{K}$  для множества образцов  $\mathbb{P} = \{\textit{potato}, \textit{poetry}, \textit{pottery}, \textit{science}, \textit{school}\}$ . Дерево ключей строится за время  $O(n)$ .

## Очевидный способ

- ▶ Последовательно прикладывает корень дерева  $\mathbb{K}$  к каждой позиции в тексте и пытается пройти путь в дереве согласно символам текста.
- ▶ Время работы  $O(nm)$ .
- ▶ Можно ли сдвигать дерево более чем на одну позицию?
- ▶ Для простоты: ни один образец в  $\mathbb{P}$  не является собственной подстрокой другого образца.

# Раздел

## Алгоритм Кнута-Морриса-Пратта

Классический вариант алгоритма

## Алгоритм Ахо-Корасик

Задача множественного поиска

Связи неудач

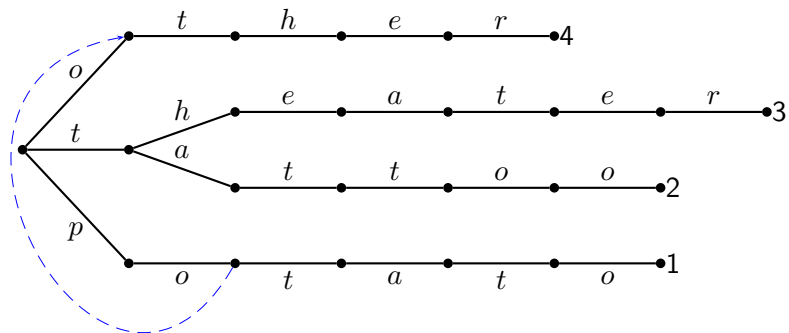
Полный алгоритм поиска

## Поиск с джокером

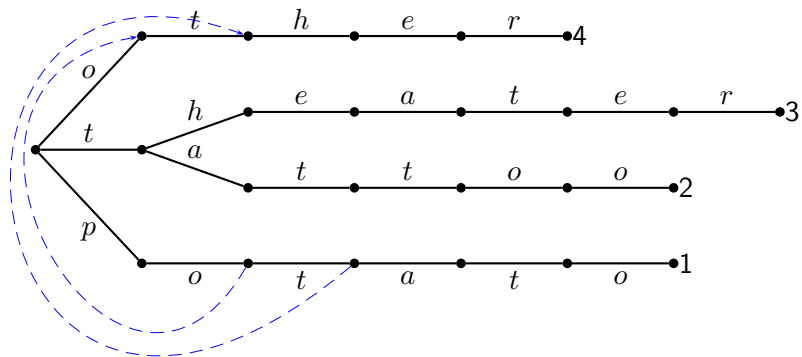
$lp(v)$

- ▶  $\mathbb{L}(v)$  — конкатенация символов на пути от корня  $\mathbb{K}$  до вершины  $v$  в порядке их появления.
- ▶ Для любой вершины  $v \in \mathbb{K}$  определим  $lp(v)$  как длину наибольшего собственного суффикса строки  $\mathbb{L}(v)$ , которая является префиксом некоторого образца из  $\mathbb{P}$ .

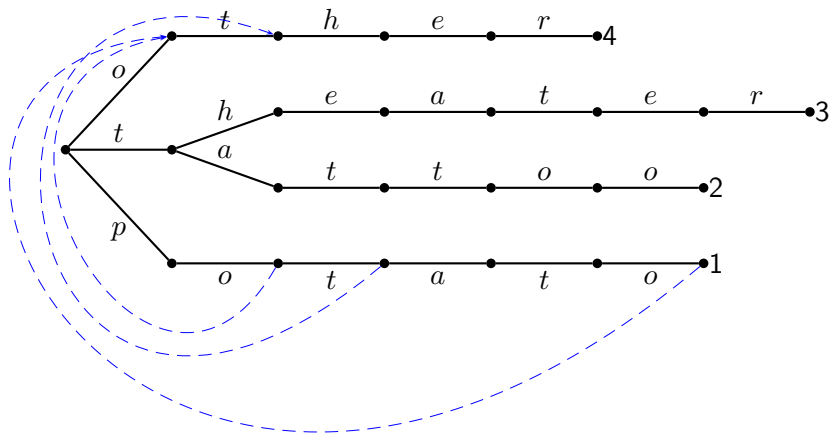
# Связи неудач



# Связи неудач

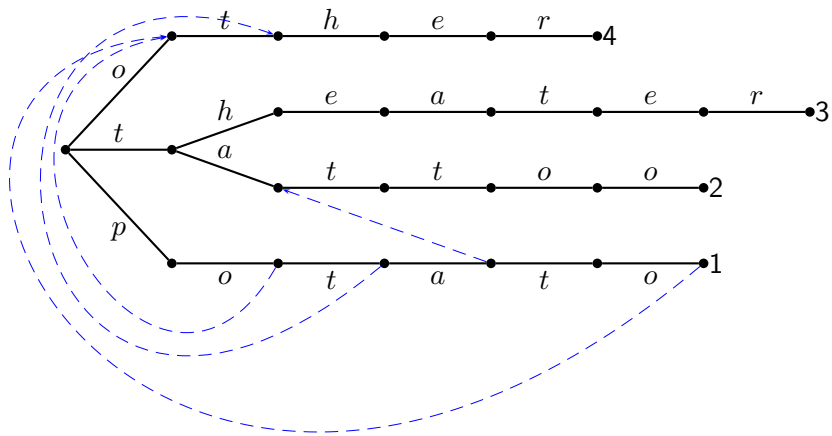


# Связи неудач

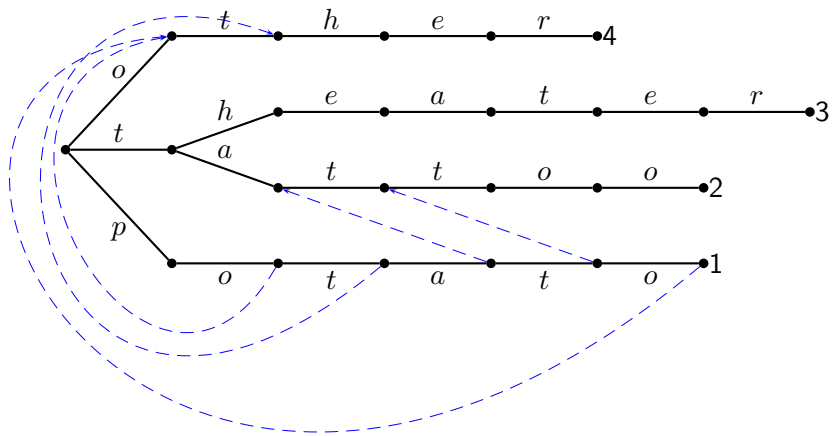




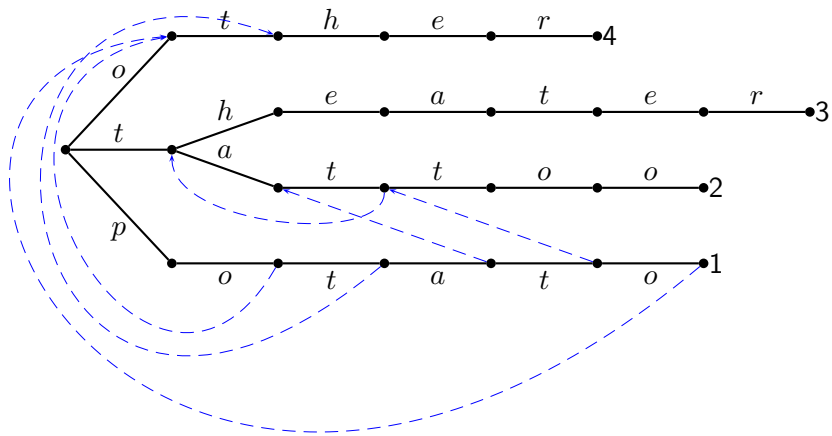
# Связи неудач



# Связи неудач

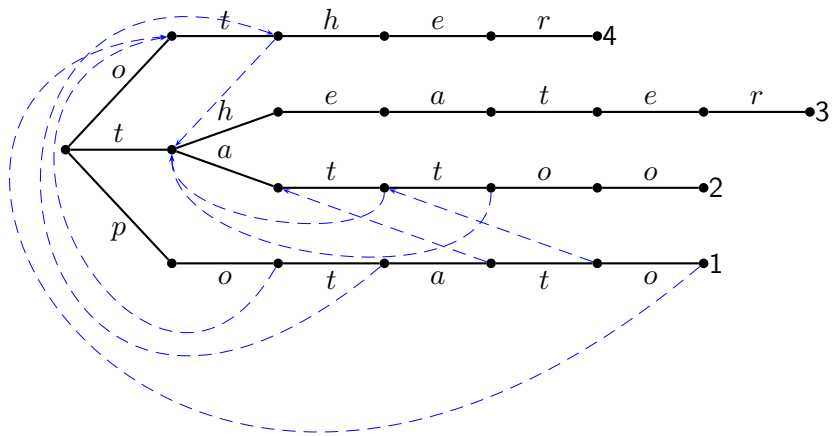


# Связи неудач

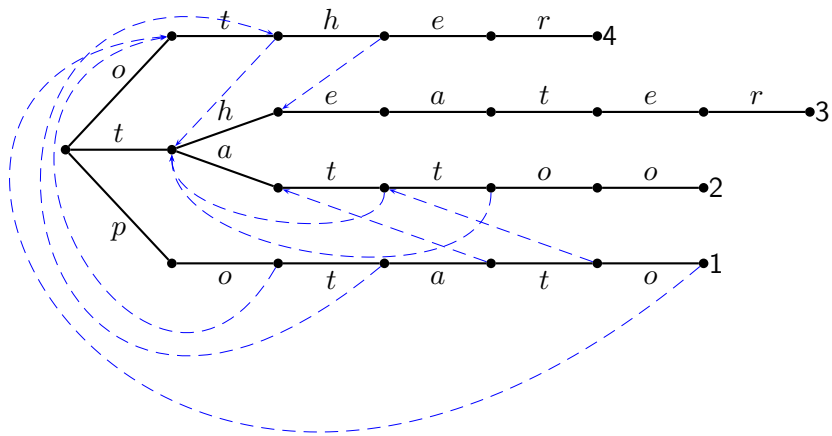




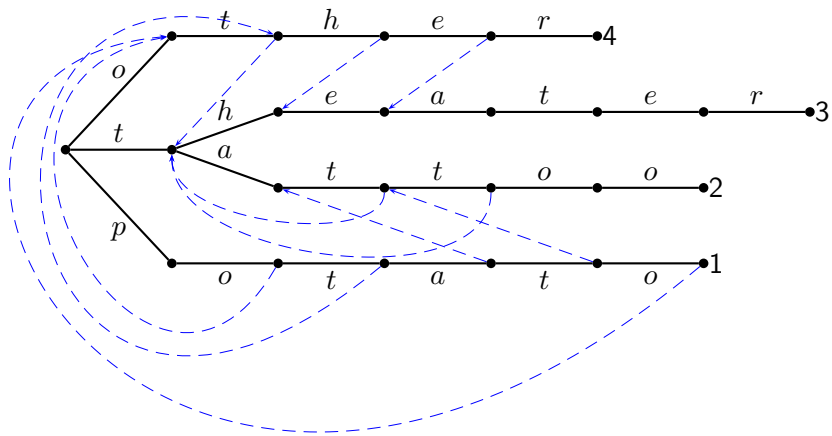
# Связи неудач



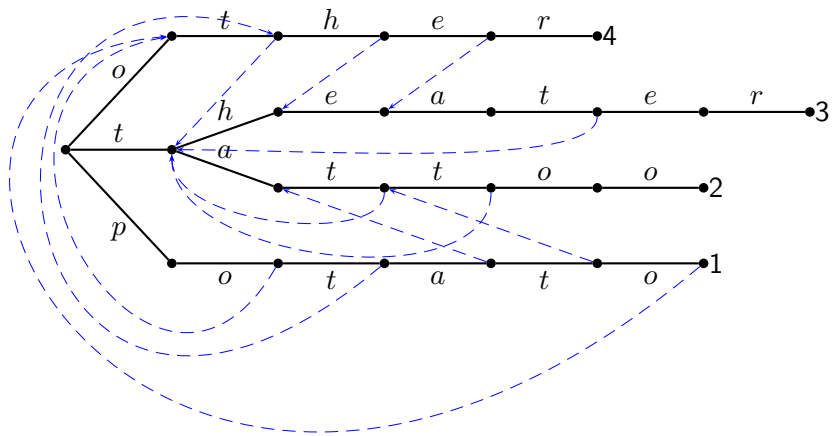
# Связи неудач



# Связи неудач

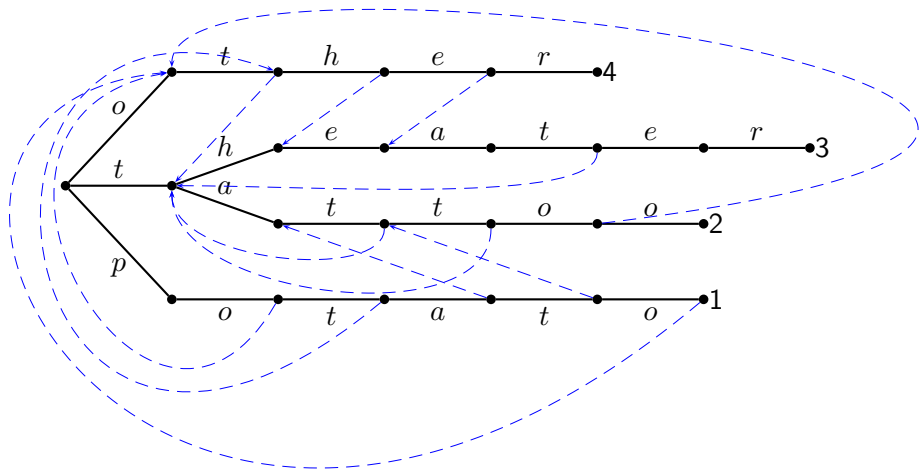


# Связи неудач

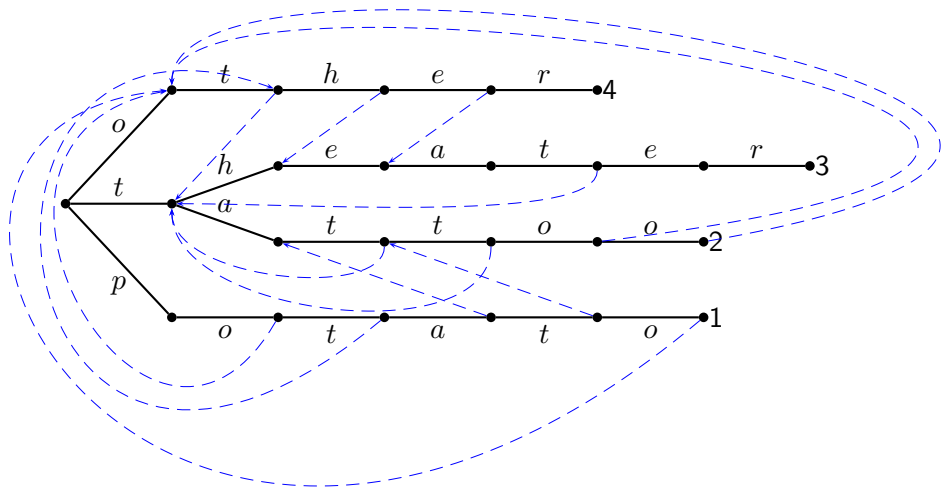




# Связи неудач



# Связи неудач



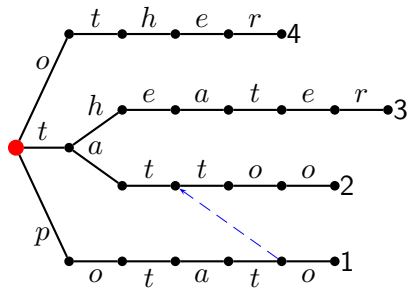
## Связь неудачи

- ▶ Допустим,  $\alpha$  — суффикс строки  $\mathbb{L}(v)$  длины  $lp(v)$ . Тогда существует единственная вершина в дереве ключей, помеченная строкой  $\alpha$ . (очевидно)
- ▶ Для вершины  $v \in \mathbb{K}$  пусть  $n_v$  — единственная вершина в  $\mathbb{K}$ , помеченная суффиксом  $\mathbb{L}(v)$  длины  $lp(v)$ . Если  $lp(v) = 0$ , то  $n_v$  — корень  $\mathbb{K}$ .
- ▶ Упорядоченная пара  $\langle v, n_v \rangle$  — связь неудачи.

## Использование связей неудач при поиске

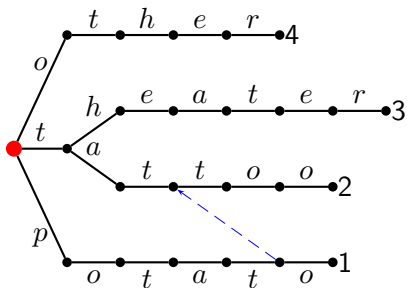
```
1  $l \leftarrow 1, c \leftarrow 1, w \leftarrow \text{root}[\mathbb{K}]$ 
2 repeat
3     while есть дуга  $\langle w, w' \rangle$ , помеченная символом  $T(c)$ 
4         if  $w'$  занумерована образцом  $i$ 
5              $P_i$  встретилась в  $T$  в позиции  $l$ 
6              $w \leftarrow w', c \leftarrow c + 1$ 
7         if  $w = \text{root}[\mathbb{K}]$ 
8              $c \leftarrow c + 1$ 
9         else  $w \leftarrow n_w, l \leftarrow c - lp(w)$ 
10 until  $c > m$ 
```

# Пример использования связей неудач



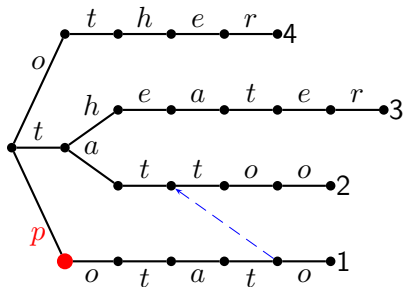
xtpotattoox

# Пример использования связей неудач



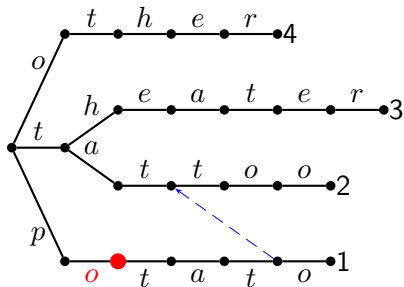
xxpotattoxx

# Пример использования связей неудач



xxp<sup>o</sup>tattooxx

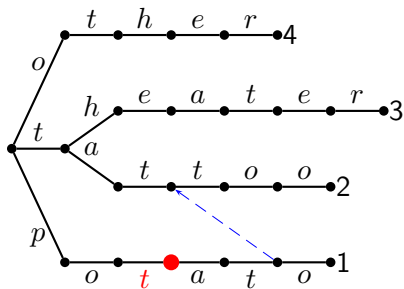
# Пример использования связей неудач



xxp<sup>o</sup>tattooxx

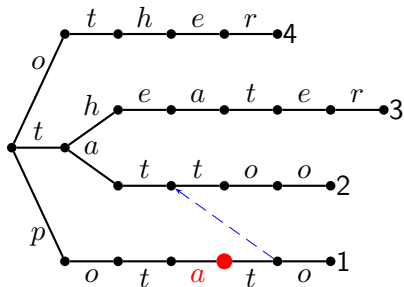


# Пример использования связей неудач



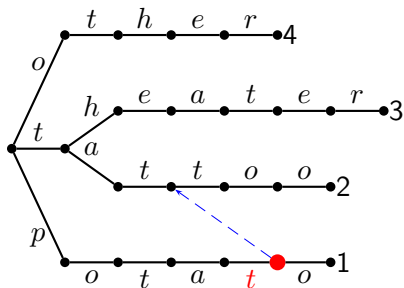
xxpotattooxx

# Пример использования связей неудач



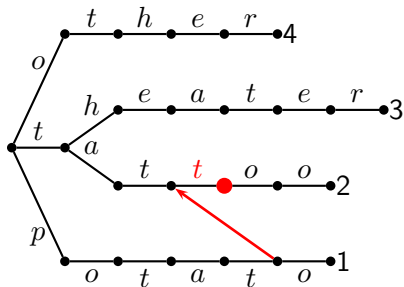
xxpotattoxx

# Пример использования связей неудач



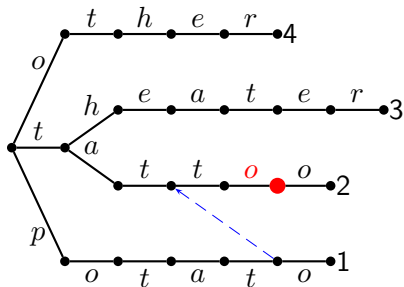
xxpotat<sup>t</sup>toohx

# Пример использования связей неудач



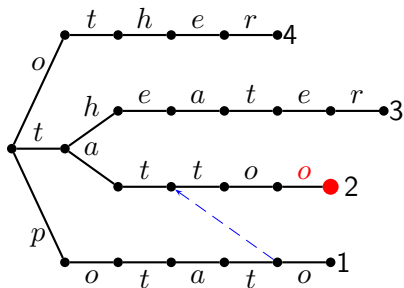
xxpotatooxx

# Пример использования связей неудач



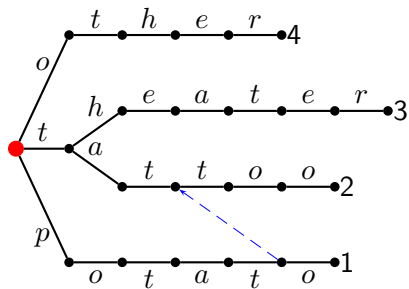
xxpotatt $\circ$ oxx

# Пример использования связей неудач



xxpotatto~~o~~xx

# Пример использования связей неудач



xxpotattooxx

## Корректность и линейность алгоритма

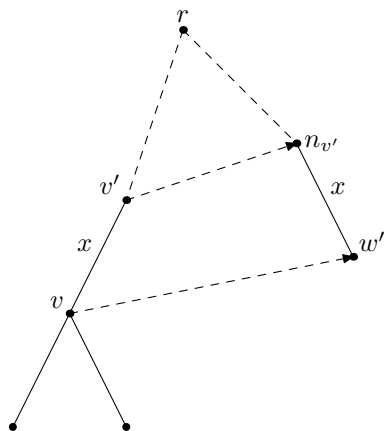
- ▶ Доказательство аналогично приведённым соображениям для алгоритма Кнута-Морриса-Пратта.
- ▶ Время поиска (с предположением об образцах, не являющихся собственными подстроками) получается порядка  $O(m)$ .



## Построение связей неудач

- ▶ Нужно найти для всех вершин  $v$  соответствующие им  $n_v$ .
- ▶ Предположим, что  $n_v$  вычислено для всех вершин, отстоящих на  $k$  дуг от корня.
- ▶ Вычисляем  $n_v$  для одной из вершин  $v$ , отстоящих на  $k + 1$  дуг от корня.

## Функция неудач узла $v$



- ▶  $v'$  — отец  $v$  в  $\mathbb{K}$ .
- ▶  $n_{v'}$  — известно, т.к.  $v'$  отстоит на  $k$  дуг от корня.
- ▶  $x$  — символ на дуге  $\langle v', v \rangle$ .
- ▶ Нужно проверить, есть ли дуга  $\langle n_{v'}, w' \rangle$ , помеченная символом  $x$ .
- ▶ Если есть, то  $n_v \leftarrow w'$ .
- ▶ В противном случае  $\mathbb{L}(n_v)$  — собственный суффикс  $\mathbb{L}(n_{v'})$ , за которым следует  $x$ .

## Построение связей неудач

- 1  $v'$  — отец  $v$  в  $\mathbb{K}$ ,  $x$  — символ на дуге  $\langle v', v \rangle$ .
- 2  $w \leftarrow n_{v'}$
- 3 **while** нет дуги, выходящей из  $w$ , помеченной  $x$  и  $w \neq r$
- 4      $w \leftarrow n_w$
- 5 **if** есть дуга, выходящая из  $w$  и помеченная  $x$
- 6      $n_v \leftarrow w'$
- 7 **else**  $n_v \leftarrow r$

# Линейность построения связей неудач

## Теорема

*Полное время, затрачиваемое алгоритмом построения связей неудач в применении его ко всем вершинам из  $\mathbb{K}$  равно  $O(n)$ .*

## Доказательство.

- ▶ Изменение  $lp(v)$  при выполнении алгоритма по пути одного образца  $P$  длиной  $t$ .
- ▶  $lp(v) \leq lp(v') + 1$ , следовательно  $lp(v)$  увеличивается не более чем на  $t$ .
- ▶ При уменьшении  $lp(v) \geq 0$  и внутренний цикл не может выполняться более  $t$  раз.
- ▶ Следовательно, связи неудач по пути  $P$  находятся за время  $O(t)$ , а все связи — за  $O(n)$ .



# Раздел

Алгоритм Кнута-Морриса-Пратта  
Классический вариант алгоритма

Алгоритм Ахо-Корасик  
Задача множественного поиска  
Связи неудач  
Полный алгоритм поиска

Поиск с джокером

## Снятие предположения о подстроках

Пример:  $\mathbb{P} = \{acatt, ca\}$ ,  $T = acatg$ : образец  $ca$  не будет найден.

## Снятие предположения о подстроках

Пример:  $\mathbb{P} = \{acatt, ca\}$ ,  $T = acatg$ : образец  $ca$  не будет найден.

### Теорема

1. Пусть в дереве ключей  $\mathbb{K}$  существует путь из связей неудач от вершины  $v$  к вершине, занумерованной образцом  $i$ . Тогда в  $T$  должен обнаружиться образец  $P_i$ , который оканчивается в позиции  $s$  (текущий символ), как только во время фазы поиска алгоритма Ахо-Корасик будет достигнута вершина  $v$ .

## Снятие предположения о подстроках

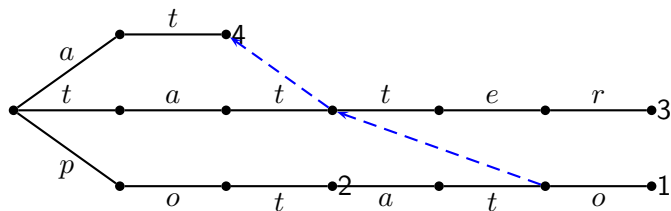
Пример:  $\mathbb{P} = \{acatt, ca\}$ ,  $T = acatg$ : образец  $ca$  не будет найден.

### Теорема

1. Пусть в дереве ключей  $\mathbb{K}$  существует путь из связей неудач от вершины  $v$  к вершине, занумерованной образцом  $i$ . Тогда в  $T$  должен обнаружиться образец  $P_i$ , который оканчивается в позиции  $s$  (текущий символ), как только во время фазы поиска алгоритма Ахо-Корасик будет достигнута вершина  $v$ .
2. И наоборот: если в ходе работы достигнута вершина  $v$ , то образец  $P_i$  появляется в  $T$ , заканчиваясь в позиции  $s$ , только если  $v$  имеет номер  $i$  или существует путь из связей неудач из  $v$  в вершину с номером  $i$ .



Путь от *potat* до *at* через *tat*



## Полный алгоритм

```
1  $l \leftarrow 1, c \leftarrow 1, w \leftarrow \text{root}[\mathbb{K}]$ 
2 repeat
3     while есть дуга  $\langle w, w' \rangle$ , помеченная символом  $T(c)$ 
4         if  $w'$  занумерована образцом  $i$  или
           существует путь из связей неудач из  $w'$ 
           в вершину с номером  $i$ 
5              $P_i$  встретилась в  $T$  в позиции  $l$ 
6              $w \leftarrow w', c \leftarrow c + 1$ 
7         if  $w = \text{root}[\mathbb{K}]$ 
8              $c \leftarrow c + 1$ 
9         else  $w \leftarrow n_w, l \leftarrow c - lp(w)$ 
10 until  $c > m$ 
```

## Детали реализации

- ▶ Связь выхода в вершине  $v$  указывает на нумерованную вершину, отличную от  $v$  и достижимую из  $v$  за наименьшее число связей неудач.
- ▶ Связи выхода можно получить за время  $O(n)$  при построении связей неудач.
- ▶ Использование связей выхода можно решить задачу множественного совпадения за  $O(m + k)$ , где  $k$  — полное число вхождений.
- ▶ Полное время работы алгоритма:  $O(n)$  на подготовку и  $O(m + k)$  на поиск, т.е.  $O(n + m + k)$ .

# Раздел

Алгоритм Кнута-Морриса-Пратта  
Классический вариант алгоритма

Алгоритм Ахо-Корасик  
Задача множественного поиска  
Связи неудач  
Полный алгоритм поиска

Поиск с джокером

# Джокер

- ▶ Специальный метасимвол, «джокер»  $?$ , совпадает с любым символом.
- ▶ Образец может содержать в себе джокер, например  $ab??c?$ .
- ▶ Тогда в  $xabvccbxababcax$  образец встречается дважды.
- ▶ При неограниченном количестве джокеров в строке линейное решение неизвестно.
- ▶ Конечный автомат — нужно время на построение.
- ▶ При ограничении на число джокеров (независимо от длины  $P$ ) линейное решение основывается на алгоритме Ахо-Корасик.

## Общая идея

1.  $C$  — вектор длины  $T$ , инициализированный нулями.
2.  $\mathbb{P} = \{P_1, P_2, \dots, P_k\}$  — набор максимальных подстрок  $P$  без джокеров.  $l_1, l_2, \dots, l_k$  — начальные позиции этих подстрок в  $P$ . Для  $P = ab??c?ab??$   $\mathbb{P} = \{ab, c, ab\}$  и  $l_1 = 1$ ,  $l_2 = 5$  и  $l_3 = 7$

## Общая идея

1.  $C$  — вектор длины  $T$ , инициализированный нулями.
2.  $\mathbb{P} = \{P_1, P_2, \dots, P_k\}$  — набор максимальных подстрок  $P$  без джокеров.  $l_1, l_2, \dots, l_k$  — начальные позиции этих подстрок в  $P$ . Для  $P = ab??c?ab??$   $\mathbb{P} = \{ab, c, ab\}$  и  $l_1 = 1$ ,  $l_2 = 5$  и  $l_3 = 7$
3. Алгоритмом Ахо-Корасик найти все вхождения  $P_i$  в  $T$ .  
Для каждого вхождения  $P_i$  в  $j$ -й позиции текста увеличить счётчик  $C[j - l_i + 1]$  на единицу.

## Общая идея

1.  $C$  — вектор длины  $T$ , инициализированный нулями.
2.  $\mathbb{P} = \{P_1, P_2, \dots, P_k\}$  — набор максимальных подстрок  $P$  без джокеров.  $l_1, l_2, \dots, l_k$  — начальные позиции этих подстрок в  $P$ . Для  $P = ab??c?ab??$   $\mathbb{P} = \{ab, c, ab\}$  и  $l_1 = 1$ ,  $l_2 = 5$  и  $l_3 = 7$
3. Алгоритмом Ахо-Корасик найти все вхождения  $P_i$  в  $T$ .  
Для каждого вхождения  $P_i$  в  $j$ -й позиции текста увеличить счётчик  $C[j - l_i + 1]$  на единицу.
4. Вхождение  $P$  в  $T$ , начинающиеся в позиции  $p$ , имеется в том и только том случае, если  $C(p) = k$ .



## Общая идея

1.  $C$  — вектор длины  $T$ , инициализированный нулями.
2.  $\mathbb{P} = \{P_1, P_2, \dots, P_k\}$  — набор максимальных подстрок  $P$  без джокеров.  $l_1, l_2, \dots, l_k$  — начальные позиции этих подстрок в  $P$ . Для  $P = ab??c?ab??$   $\mathbb{P} = \{ab, c, ab\}$  и  $l_1 = 1$ ,  $l_2 = 5$  и  $l_3 = 7$
3. Алгоритмом Ахо-Корасик найти все вхождения  $P_i$  в  $T$ .  
Для каждого вхождения  $P_i$  в  $j$ -й позиции текста увеличить счётчик  $C[j - l_i + 1]$  на единицу.
4. Вхождение  $P$  в  $T$ , начинающиеся в позиции  $p$ , имеется в том и только том случае, если  $C(p) = k$ .
5. Время поиска  $O(km)$  из-за использования массива  $C$ , если  $k$  ограничено константой, не зависящей от  $|P|$ , то время поиска — линейно.