Алгоритм Укконена

Лилия Лурье

СПбГУ

2021

1.

Бор

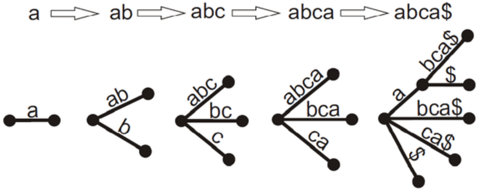
Бор — структура данных для хранения набора строк, представляющая из себя подвешенное дерево с символами на рёбрах.

Бор

Бор — структура данных для хранения набора строк, представляющая из себя подвешенное дерево с символами на рёбрах.

Суффиксное дерево

Суффиксное дерево (ST) — бор, содержащий все суффиксы некоторой строки.



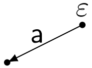
Построим суффиксное дерево для строки
$S = \text{ababca}$

ε

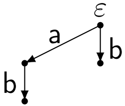
Наивный метод

Построим суффиксное дерево для строки

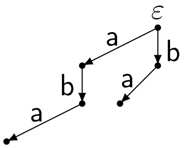
$S =$ ababca



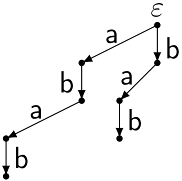
Построим суффиксное дерево для строки
$S = $ ababca



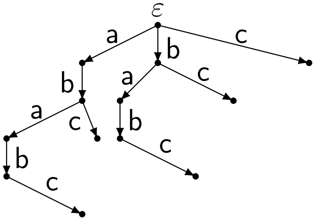
Построим суффиксное дерево для строки
$S =$ ababca



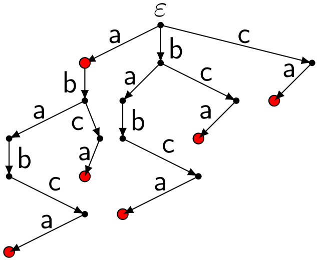
Построим суффиксное дерево для строки
$S =$ ababca



Построим суффиксное дерево для строки
$S = \mathrm{ababca}$



Построим суффиксное дерево для строки
$S = \text{ababc}\textcolor{red}{\text{a}}$



Мы строили ST, сканируя строку посимвольно
На i-ом шаге было получено ST для префикса S длины i

Мы строили ST, сканируя строку посимвольно
На i-ом шаге было получено ST для префикса S длины i

Псевдокод

for i = $1 \dots n$:
  for j = $1 \dots i$:
    treeExtend($S[j \dots i]$);

Мы строили ST, сканируя строку посимвольно
На i-ом шаге было получено ST для префикса S длины i

Псевдокод

for i $= 1 \ldots$ n:
  for j $= 1 \ldots$ i:
    treeExtend$(S[j \ldots i])$;

Ассимптотика алгоритма:

Мы строили ST, сканируя строку посимвольно
На i-ом шаге было получено ST для префикса S длины i

Псевдокод

for i = 1...n:
  for j = 1...i:
    treeExtend($S[j \dots i]$);

Ассимптотика алгоритма: $O(n^3)$

Сжатое дерево

- Наивно построенное дерево имеет размер $O(n^2)$

- Наивно построенное дерево имеет размер $O(n^2)$
- Как его уменьшить?

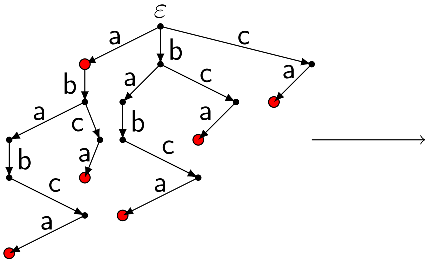
- Наивно построенное дерево имеет размер $O(n^2)$
- Как его уменьшить?

1. Избавимся от вершин с 1 потомком
   Для этого на некоторых рёбрах запишем подстроки длины, большей 1

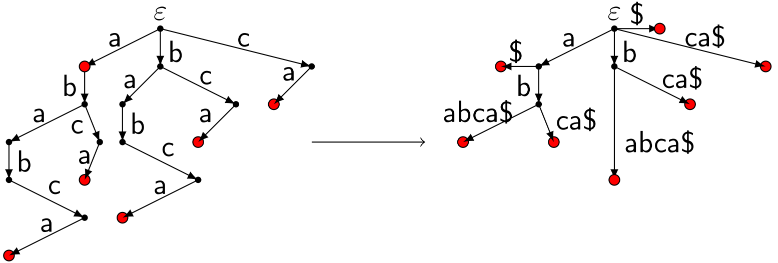
- Наивно построенное дерево имеет размер $O(n^2)$
- Как его уменьшить?

1. Избавимся от вершин с 1 потомком
   Для этого на некоторых рёбрах запишем подстроки длины, большей 1
2. Избавимся от терминальных вершин, не являющихся листьями
   Для этого построим ST для строки $S\$$
   Тогда суффиксам будут соответствовать только листья

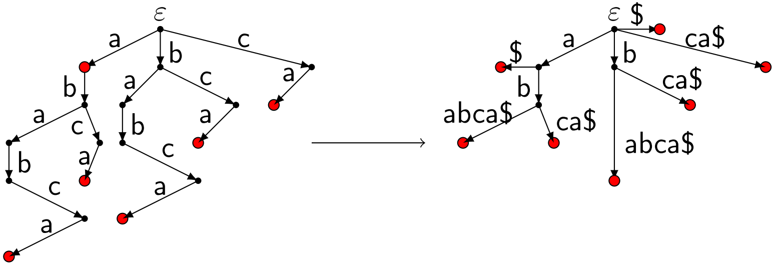
Сжатое дерево



Сжатое дерево



Сжатое дерево



- В этом дереве n листьев и $\leq n-1$ промежуточных вершин (т.к. у них ≥ 2 потомков)
- Размер этого дерева - $O(n)$

Vertex:

Vertex:

Edge[] children; $\leftarrow$ массив рёбер, ведущих в потомков

Vertex:

Edge[] children; $\leftarrow$ массив рёбер, ведущих в потомков

Edge:

Способ хранения сжатого дерева

Vertex:

Edge[] children; ← массив рёбер, ведущих в потомков

Edge:

Node aim; ← вершина - конец ребра

Vertex:

Edge[] children; $\leftarrow$ массив рёбер, ведущих в потомков

Edge:

Node aim; $\leftarrow$ вершина - конец ребра

int left, right; $\leftarrow$ индексы начала и конца подстроки, задающей ребро

Определение

Суффиксная ссылка вершины v - это вершина, в которой оканчивается длиннейший собственный суффикс подстроки, соответствующей v

Определение

Суффиксная ссылка вершины v - это вершина, в которой оканчивается длиннейший собственный суффикс подстроки, соответствующей v

- Суффиксная ссылка корня - корень

Определение

Суффиксная ссылка вершины v - это вершина, в которой оканчивается длиннейший собственный суффикс подстроки, соответствующей v

- Суффиксная ссылка корня - корень

Что мы можем сказать про суффиксную ссылку некоторой подстроки α в суффиксном дереве? (Какова длина подстроки, на которую она указывает?)

Определение

Суффиксная ссылка вершины v - это вершина, в которой оканчивается длиннейший собственный суффикс подстроки, соответствующей v

- Суффиксная ссылка корня - корень

Что мы можем сказать про суффиксную ссылку некоторой подстроки α в суффиксном дереве? (Какова длина подстроки, на которую она указывает?)
Если суффиксная ссылка α - это некоторая подстрока β, то $|\beta| = |\alpha| - 1$
Это объясняется тем, что суффиксное дерево содержит все подстроки, т.е. строка α без $\alpha[1]$ должна содержаться в ST В сжатом ST также $|\beta| = |\alpha| - 1$

Vertex:

Edge[] children; ← массив рёбер, ведущих в потомков

Node suf_link; ← суффиксная ссылка вершины

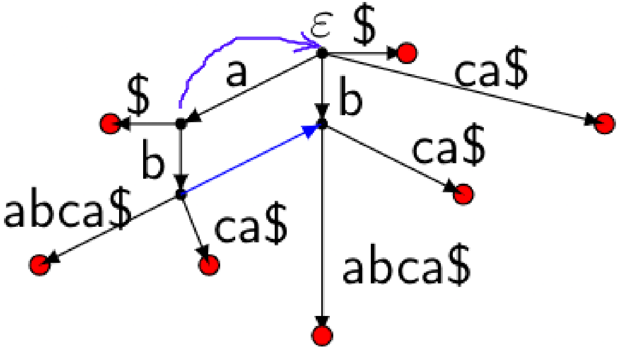
Edge:

Node aim; ← вершина - конец ребра

int left, right; ← индексы начала и конца подстроки, задающей ребро

Хранение сжатого дерева

Суффиксное дерево строки "ababca":



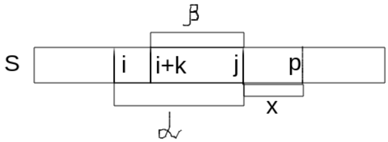
Утверждение

Пусть суффиксная ссылка строки α в ST - строка β
Тогда если из вершины, соответствующей подстроке α, есть ребро x, то и из β есть ребро x

Утверждение

Пусть суффиксная ссылка строки α в ST - строка β
Тогда если из вершины, соответствующей подстроке α, есть ребро x, то и из β есть ребро x

Доказательство:

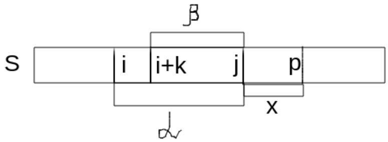


Утверждение

Пусть суффиксная ссылка строки α в ST - строка β
Тогда если из вершины, соответствующей подстроке α, есть ребро x, то и из β есть ребро x

Доказательство:

- $\alpha = S[i..j]$, где $i \leq j \leq |S|$

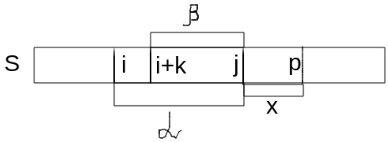


Утверждение

Пусть суффиксная ссылка строки α в ST - строка β
Тогда если из вершины, соответствующей подстроке α, есть ребро x, то и из β есть ребро x

Доказательство:

- $\alpha = S[i..j]$, где $i \leq j \leq |S|$
- В строке S есть подстрока $\alpha x = S[i..p],\ p > j,\ S[j+1..p] = x$

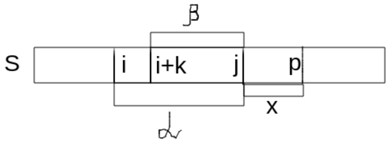


Утверждение

Пусть суффиксная ссылка строки α в ST - строка β
Тогда если из вершины, соответствующей подстроке α, есть ребро x, то и из β есть ребро x

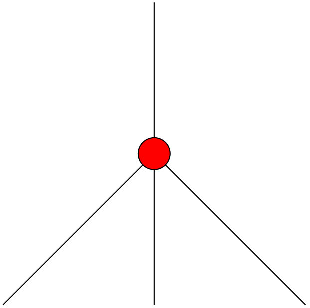
Доказательство:

- $\alpha = S[i..j]$, где $i \leq j \leq |S|$
- В строке S есть подстрока $\alpha x = S[i..p],\ p > j,\ S[j+1..p] = x$
- β - суффикс α, т.е. $\beta = S[i+k..j],\ k \leq j-i \Rightarrow$ в S есть подстрока $\beta x = S[i+k..p]$



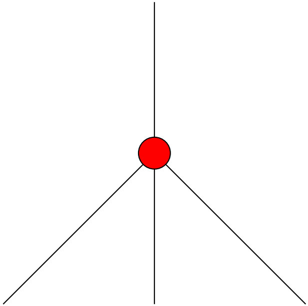
Варианты текущего состояния

Варианты текущего состояния

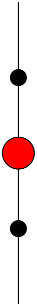


Case 1:
Храним Node

Варианты текущего состояния



Case 1:
Храним Node



Case 2:
Храним Edge и pos

Перемещение вниз

Псевдокод

Перемещение вниз

Псевдокод

step_down(x):

Перемещение вниз

Псевдокод

step_down(x):
  if Case = 1:

Перемещение вниз

Псевдокод

```
step_down(x):
  if Case = 1:
    Case ← 2;
    Edge ← Node.children(x);
    pos ← 1;
```

Перемещение вниз

Псевдокод

step_down(x):
  if Case = 1:
    Case ← 2;
    Edge ← Node.children(x);
    pos ← 1;
  else:

Перемещение вниз

Псевдокод

step_down(x):
  if Case = 1:
    Case ← 2;
    Edge ← Node.children(x);
    pos ← 1;
  else:
    pos++;

Перемещение вниз

Псевдокод

```
step_down(x):
    if Case = 1:
        Case ← 2;
        Edge ← Node.children(x);
        pos ← 1;
    else:
        pos++;
    if Case = 2 and pos = Edge.(R-L):
```

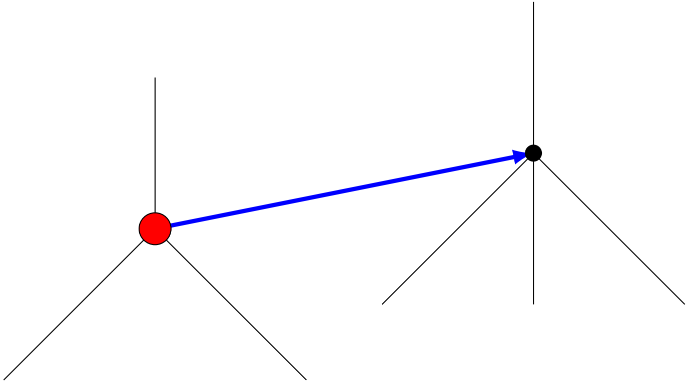
Перемещение вниз

Псевдокод

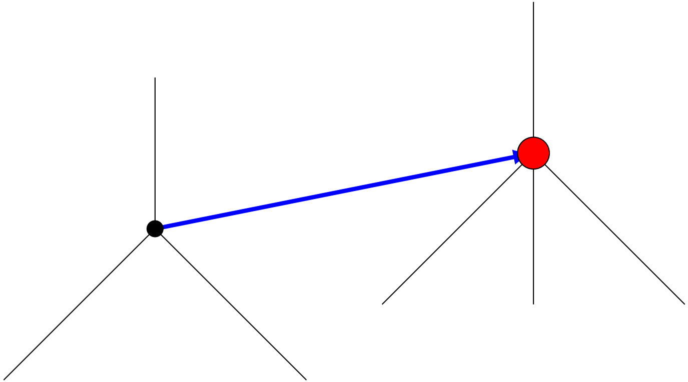
```
step_down(x):
  if Case = 1:
    Case ← 2;
    Edge ← Node.children(x);
    pos ← 1;
  else:
    pos++;
  if Case = 2 and pos = Edge.(R-L):
    Case ← 1;
    Node ← Edge.aim;
```

1. Находимся в узле сжатого ST

1. Находимся в узле сжатого ST
   Просто переходим от вершины к её суффиксной ссылке



1. Находимся в узле сжатого ST
   Просто переходим от вершины к её суффиксной ссылке



Переход по суффиксной ссылке

2. Находимся в некрайней позиции на ребре

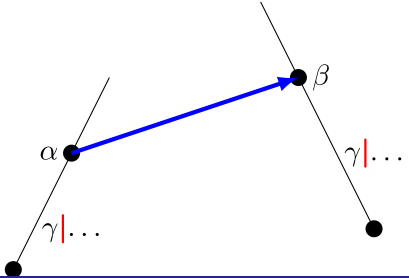
Переход по суффиксной ссылке

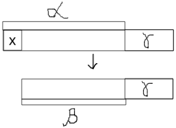
2. Находимся в некрайней позиции на ребре
   В этом случае, очевидно, нет прямой суффиксной ссылки из нашего положения

2. Находимся в некрайней позиции на ребре
   В этом случае, очевидно, нет прямой суффиксной ссылки из нашего положения
   Что делать?

2. Находимся в некрайней позиции на ребре
   В этом случае, очевидно, нет прямой суффиксной ссылки из нашего положения
   Что делать?

1.Вернуться к предыдущей вершине
2.Пройти по суффиксной ссылке
3.Пройти недостающую часть подстроки по ребру





$$x\beta = \alpha$$

Определение

Глубина вершины v (d(v)) - число рёбер на пути от корня до вершины v

Определение

Глубина вершины v (d(v)) - число рёбер на пути от корня до вершины v

Лемма

При переходе по суффиксной ссылке глубина уменьшается не более чем на 1.

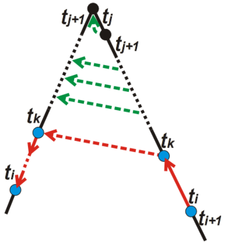
Определение

Глубина вершины v (d(v)) - число рёбер на пути от корня до вершины v

Лемма

При переходе по суффиксной ссылке глубина уменьшается не более чем на 1.

Доказательство:



Определение

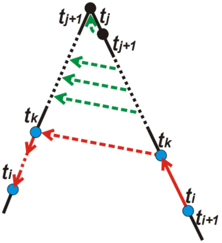
Глубина вершины v (d(v)) - число рёбер на пути от корня до вершины v

Лемма

При переходе по суффиксной ссылке глубина уменьшается не более чем на 1.

Доказательство:

- $B - S[j..i] \qquad A - S[j+1..i]$



Определение

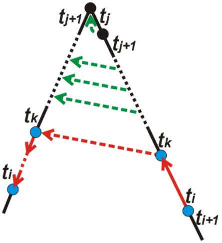
Глубина вершины v (d(v)) - число рёбер на пути от корня до вершины v

Лемма

При переходе по суффиксной ссылке глубина уменьшается не более чем на 1.

Доказательство:

- $B - S[j..i] \qquad A - S[j+1..i]$
- На пути A не более чем на одну вершину меньше, чем на B.



Определение

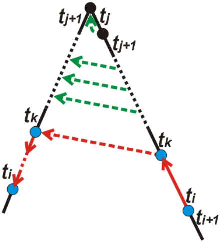
Глубина вершины v (d(v)) - число рёбер на пути от корня до вершины v

Лемма

При переходе по суффиксной ссылке глубина уменьшается не более чем на 1.

Доказательство:

- $B - S[j..i] \qquad A - S[j+1..i]$
- На пути A не более чем на одну вершину меньше, чем на B.
- Вершине v на пути B соответствует вершина u на пути A.



Определение

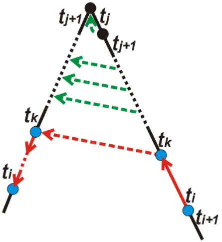
Глубина вершины v (d(v)) - число рёбер на пути от корня до вершины v

Лемма

При переходе по суффиксной ссылке глубина уменьшается не более чем на 1.

Доказательство:

- $B - S[j..i] \qquad A - S[j+1..i]$
- На пути A не более чем на одну вершину меньше, чем на B.
- Вершине v на пути B соответствует вершина u на пути A.
- Разница в одну вершину возникает, если суффиксная ссылка из первой вершины B ведёт в корень.



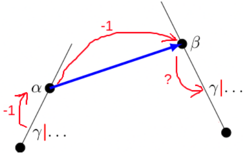
Лемма о числе переходов внутри фазы

Число переходов по рёбрам внутри фазы номер i равно $O(i)$.

Лемма о числе переходов внутри фазы

Число переходов по рёбрам внутри фазы номер i равно $O(i)$.

Доказательство:

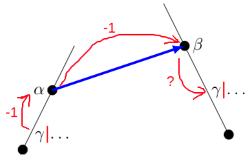


Лемма о числе переходов внутри фазы

Число переходов по рёбрам внутри фазы номер i равно $O(i)$.

Доказательство:

- Оценим количество переходов.

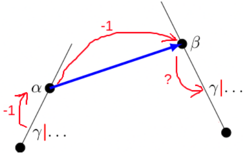


Лемма о числе переходов внутри фазы

Число переходов по рёбрам внутри фазы номер i равно $O(i)$.

Доказательство:

- Оценим количество переходов.
- Переход до ближайшей внутренней вершины уменьшает высоту на 1.

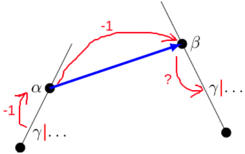


Лемма о числе переходов внутри фазы

Число переходов по рёбрам внутри фазы номер i равно $O(i)$.

Доказательство:

- Оценим количество переходов.
- Переход до ближайшей внутренней вершины уменьшает высоту на 1.
- Переход по суффиксной ссылке уменьшает высоту не более чем на 1

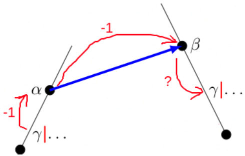


Лемма о числе переходов внутри фазы

Число переходов по рёбрам внутри фазы номер i равно $O(i)$.

Доказательство:

- Оценим количество переходов.
- Переход до ближайшей внутренней вершины уменьшает высоту на 1.
- Переход по суффиксной ссылке уменьшает высоту не более чем на 1
- Высота при движении вниз не может увеличиваться больше глубины дерева.

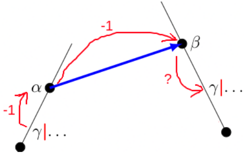


Лемма о числе переходов внутри фазы

Число переходов по рёбрам внутри фазы номер i равно $O(i)$.

Доказательство:

- Оценим количество переходов.
- Переход до ближайшей внутренней вершины уменьшает высоту на 1.
- Переход по суффиксной ссылке уменьшает высоту не более чем на 1
- Высота при движении вниз не может увеличиваться больше глубины дерева.
- Суммарно высота не может увеличиться больше чем на 2i.

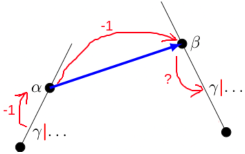


Лемма о числе переходов внутри фазы

Число переходов по рёбрам внутри фазы номер i равно $O(i)$.

Доказательство:

- Оценим количество переходов.
- Переход до ближайшей внутренней вершины уменьшает высоту на 1.
- Переход по суффиксной ссылке уменьшает высоту не более чем на 1
- Высота при движении вниз не может увеличиваться больше глубины дерева.
- Суммарно высота не может увеличиться больше чем на 2i.
- Число переходов по рёбрам за одну фазу в сумме составляет $O(i)$.



Лемма

Амортизированное время работы перехода по суффиксной ссылке - $O(1)$.

Лемма

Амортизированное время работы перехода по суффиксной ссылке - $O(1)$.

Доказательство:

Лемма

Амортизированное время работы перехода по суффиксной ссылке - $O(1)$.

Доказательство:

- Введём функцию потенциала вершины: $\Phi(v) = n - d(v)$

Лемма

Амортизированное время работы перехода по суффиксной ссылке - $O(1)$.

Доказательство:

- Введём функцию потенциала вершины: $\Phi(v) = n - d(v)$
- $\widetilde{T} = (1 + i) + (2 - i) = O(1)$

Лемма

Амортизированное время работы перехода по суффиксной ссылке - $O(1)$.

Доказательство:

- Введём функцию потенциала вершины: $\Phi(v) = n - d(v)$
- $\widetilde{T} = (1 + i) + (2 - i) = O(1)$
- Значит, суммарно все переходы по суффиксным ссылкам занимают O(n)

Шаг алгоритма

Сканируем строку посимвольно.
Пусть сейчас нами был отсканирован символ x.

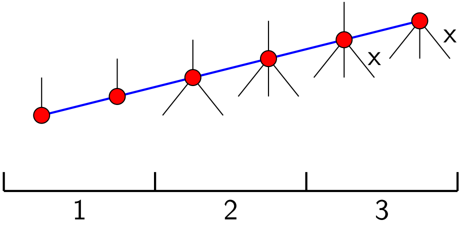
Шаг алгоритма

Сканируем строку посимвольно.
Пусть сейчас нами был отсканирован символ x.
На каждом шаге у нас есть вершины, в которых заканчиваются промежуточные суффиксы.

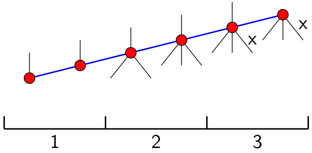
Сканируем строку посимвольно.
Пусть сейчас нами был отсканирован символ x.
На каждом шаге у нас есть вершины, в которых заканчиваются промежуточные суффиксы.
Разделим их на типы :

Шаг алгоритма

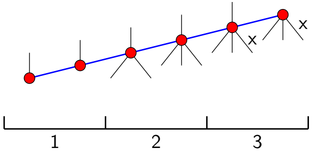
Сканируем строку посимвольно.
Пусть сейчас нами был отсканирован символ x.
На каждом шаге у нас есть вершины, в которых заканчиваются промежуточные суффиксы.
Разделим их на типы :



Шаг алгоритма



1. **Листья**

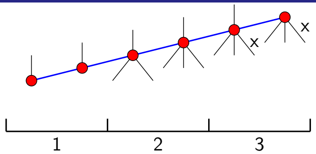


1. **Листья**

Продление листа

Добавляем x в конец подстроки, которой помечено ребро, ведущее в этот лист

Шаг алгоритма



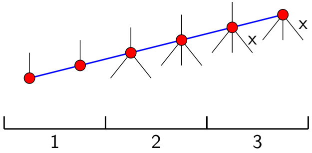
1. **Листья**

Продление листа

Добавляем x в конец подстроки, которой помечено ребро, ведущее в этот лист

Утверждение:
Листья остаются листами

Шаг алгоритма

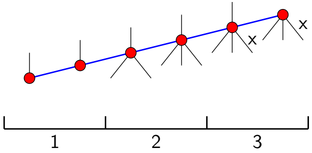


1. **Листья**

Продление листа

Добавляем x в конец подстроки, которой помечено ребро, ведущее в этот лист

Утверждение:
Листья остаются листами
Доказательство:



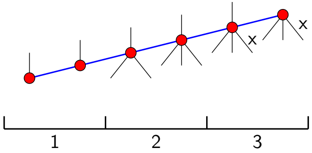
1. **Листья**

Продление листа

Добавляем x в конец подстроки, которой помечено ребро, ведущее в этот лист

Утверждение:
Листья остаются листами
Доказательство:

- У алгоритма нет механизма продолжения листового ребра дальше текущего листа.



1. **Листья**

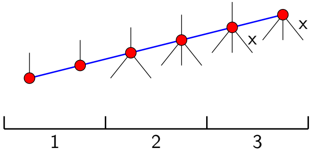
Продление листа

Добавляем x в конец подстроки, которой помечено ребро, ведущее в этот лист

Утверждение:
Листья остаются листами
Доказательство:

- У алгоритма нет механизма продолжения листового ребра дальше текущего листа.
- Если есть лист с суффиксом i, продление листа будет применяться для продолжения i на всех последующих фазах.

Шаг алгоритма



1. **Листья**

Продление листа

Добавляем x в конец подстроки, которой помечено ребро, ведущее в этот лист

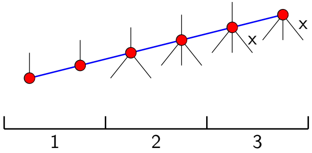
Утверждение:
Листья остаются листами
Доказательство:

- У алгоритма нет механизма продолжения листового ребра дальше текущего листа.
- Если есть лист с суффиксом i, продление листа будет применяться для продолжения i на всех последующих фазах.

Метку ребра в лист можно задать как S[i..t]

t - ссылка на переменную, хранящую конец текущей подстроки.

Шаг алгоритма



1. **Листья**

Продление листа

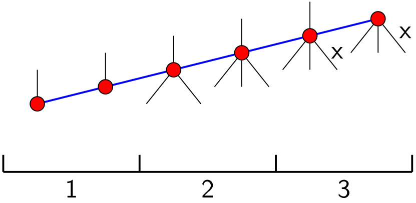
Добавляем x в конец подстроки, которой помечено ребро, ведущее в этот лист

Утверждение:
Листья остаются листами
Доказательство:

- У алгоритма нет механизма продолжения листового ребра дальше текущего листа.
- Если есть лист с суффиксом i, продление листа будет применяться для продолжения i на всех последующих фазах.

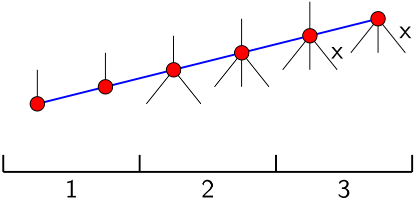
Метку ребра в лист можно задать как S[i..t]
t - ссылка на переменную, хранящую конец текущей подстроки.
Тогда на листья, после их появления, можно не смотреть.

Шаг алгоритма



2. Не лист, нет перехода по x

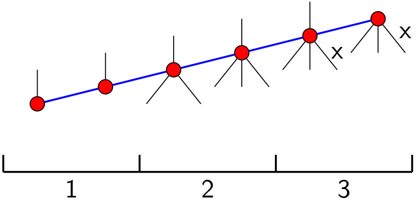
Шаг алгоритма



2. Не лист, нет перехода по x

Ответвление

Добавим к узлу новое ребро (x) и вершину на его конце

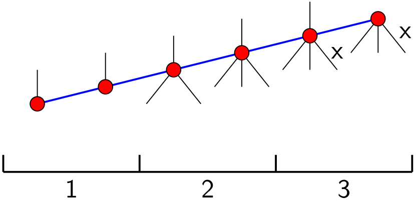


2. Не лист, нет перехода по x

Ответвление

Добавим к узлу новое ребро (x) и вершину на его конце

Утверждение:
Суммарное время добавлений равно O(n)



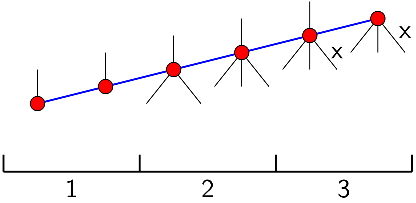
2. Не лист, нет перехода по x

Ответвление

Добавим к узлу новое ребро (x) и вершину на его конце

Утверждение:
Суммарное время добавлений равно O(n)
Доказательство:

Шаг алгоритма



2. Не лист, нет перехода по x

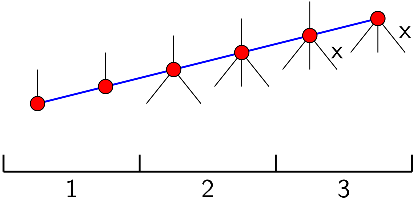
Ответвление

Добавим к узлу новое ребро (x) и вершину на его конце

Утверждение:
Суммарное время добавлений равно O(n)
Доказательство:

- Общий размер дерева - O(n)

Шаг алгоритма



2. Не лист, нет перехода по x

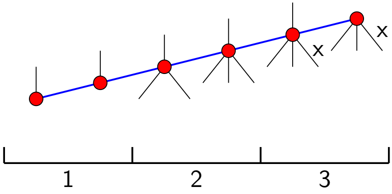
Ответвление

Добавим к узлу новое ребро (x) и вершину на его конце

Утверждение:
Суммарное время добавлений равно O(n)
Доказательство:

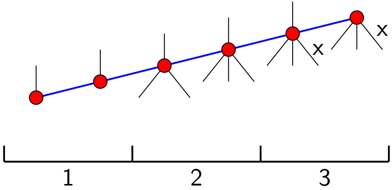
- Общий размер дерева - O(n)
- Поэтому мы не можем добавлять рёбра больше O(n) раз

Шаг алгоритма



3. Вершины с ребром x

Шаг алгоритма

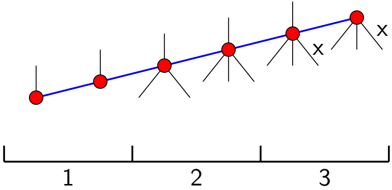


3. Вершины с ребром x

Ничего не делаем

Так как из вершины уже есть ребро, ведущее по символу x, нам не нужно ничего добавлять

Шаг алгоритма



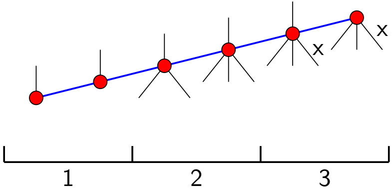
3. Вершины с ребром x

Ничего не делаем

Так как из вершины уже есть ребро, ведущее по символу x, нам не нужно ничего добавлять

Замечание:
Если мы применили это правило к какой-то вершине, то оно же будет применяться и к последующим вершинам

Шаг алгоритма



3. Вершины с ребром x

Ничего не делаем

Так как из вершины уже есть ребро, ведущее по символу x, нам не нужно ничего добавлять

Замечание:
Если мы применили это правило к какой-то вершине, то оно же будет применяться и к последующим вершинам
Значит, шаг алгоритма можно завершать после первой же встреченной вершины с ребром x.

- Храним в рёбрах, ведущих в листья, переменную t и неявно продлеваем их за O(1)

- Храним в рёбрах, ведущих в листья, переменную t и неявно продлеваем их за O(1)
- Алгоритм явно работает с суффиксами типа 2 в диапазоне $[j'..k]$

- Храним в рёбрах, ведущих в листья, переменную t и неявно продлеваем их за O(1)
- Алгоритм явно работает с суффиксами типа 2 в диапазоне $[j'..k]$
- После применения действия типа 3 на суффиксе S[k..i] фаза алгоритма завершается

Итоговый шаг алгоритма

- Храним в рёбрах, ведущих в листья, переменную t и неявно продлеваем их за O(1)
- Алгоритм явно работает с суффиксами типа 2 в диапазоне $[j'..k]$
- После применения действия типа 3 на суффиксе S[k..i] фаза алгоритма завершается
- Следующая фаща начинается с j' = k-1

Итоговый шаг алгоритма

Текущую вершину обозначим cur_pos.

Текущую вершину обозначим cur_pos.

Псевдокод

for symb in S:

Текущую вершину обозначим cur_pos.

Псевдокод

for symb in S:

    if exists i: S[cur_pos.children(i).left] = symb: //тип 3

Текущую вершину обозначим cur_pos.

Псевдокод

for symb in S:

  if exists i: S[cur_pos.children(i).left] = symb: //тип 3
   cur_pos ← step_*down*(symb);

Текущую вершину обозначим cur_pos.

Псевдокод

for symb in S:

if exists i: S[cur_pos.children(i).left] = symb: //тип 3
cur_pos ← step_*down*(symb);

else:

Текущую вершину обозначим cur_pos.

Псевдокод

for symb in S:

    if exists i: S[cur_pos.children(i).left] = symb: //тип 3
        cur_pos ← step_*down*(symb);

    else:
        Создаём переход из cur_pos по symb;//тип 2

Текущую вершину обозначим cur_pos.

Псевдокод

for symb in S:

if exists i: S[cur_pos.children(i).left] = symb: //тип 3
cur_pos ← step_*down*(symb);

else:
Создаём переход из cur_pos по symb;//тип 2
Дописываем в ребро все символы; //тип 1

Текущую вершину обозначим cur_pos.

Псевдокод

for symb in S:

    if exists i: S[cur_pos.children(i).left] = symb: //тип 3
        cur_pos ← step_*down*(symb);

    else:
        Создаём переход из cur_pos по symb;//тип 2
        Дописываем в ребро все символы; //тип 1
        cur_pos ← "переход по *suf_link*";
        prev.suf_link ← cur_pos;

Текущую вершину обозначим cur_pos.

Псевдокод

```
for symb in S:
    while true:
        if exists i: S[cur_pos.children(i).left] = symb: //тип 3
            cur_pos ← step_down(symb);
            break;
        else:
            Создаём переход из cur_pos по symb;//тип 2
            Дописываем в ребро все символы; //тип 1
            cur_pos ← "переход по suf_link";
            prev.suf_link ← cur_pos;
```

Оценка времени работы алгоритма

- Создается не более $O(n)$ вершин

Оценка времени работы алгоритма

- Создается не более O(n) вершин
- Все листы увеличиваются на текущий символ за O(1)

- Создается не более $O(n)$ вершин
- Все листы увеличиваются на текущий символ за $O(1)$
- Текущая фаза алгоритма длится до встречи вершины типа 3. Сначала продлеваются все листовые суффиксы. Затем создаются новые внутренние вершины по типу 2

- Создается не более O(n) вершин
- Все листы увеличиваются на текущий символ за O(1)
- Текущая фаза алгоритма длится до встречи вершины типа 3. Сначала продлеваются все листовые суффиксы. Затем создаются новые внутренние вершины по типу 2
- Так как вершин не может быть создано больше, чем их есть, то амортизационно на каждой фазе будет создано O(1) вершин

- Создается не более O(n) вершин
- Все листы увеличиваются на текущий символ за O(1)
- Текущая фаза алгоритма длится до встречи вершины типа 3. Сначала продлеваются все листовые суффиксы. Затем создаются новые внутренние вершины по типу 2
- Так как вершин не может быть создано больше, чем их есть, то амортизационно на каждой фазе будет создано O(1) вершин
- На каждой фазе мы начинаем добавление суффикса не с корня, а с индекса j', который в прошлой фвзе был первым типа 3. Поэтому, по лемме о числе переходов внутри фазы, суммарное число переходов за все n фаз равно O(n)

Плюсы

- Online-подход
- Простота

Минусы

Плюсы

- Online-подход
- Простота

Минусы

- Memory bottleneck problem

Плюсы

- Online-подход
- Простота

Минусы

- Memory bottleneck problem
- Для несложных задач эффективнее использовать другие алгоритмы

Плюсы

- Online-подход
- Простота

Минусы

- Memory bottleneck problem
- Для несложных задач эффективнее использовать другие алгоритмы
- В худшем случае одна фаза может выполняться $O(n)$ времени

Плюсы

- Online-подход
- Простота

Минусы

- Memory bottleneck problem
- Для несложных задач эффективнее использовать другие алгоритмы
- В худшем случае одна фаза может выполняться $O(n)$ времени
- Существуют алгоритмы, превосходящие алгоритм Укконена

Плюсы

- Online-подход
- Простота

Минусы

- Memory bottleneck problem
- Для несложных задач эффективнее использовать другие алгоритмы
- В худшем случае одна фаза может выполняться O(n) времени
- Существуют алгоритмы, превосходящие алгоритм Укконена
- Предполагается полная загрузка дерева в оперативную

Источники информации

- *Лекции Computer Science Center*
- Викиконспекты
- *habr.com*
- *Лекции курса "Алгоритмы для Интернета"*
- *e-maxx.ru*