# Доклад на тему:

## Суффиксные деревья. Алгоритм Укконена

Валерий Буканов

Санкт-Петербургский государственный университет

18 октября 2020 г.

# Оглавление

# Введение

## История

**Суффиксное дерево** было создано Питером Вайнером в 1973 году. Первое название – position tree.
$>$ 92 тыс. цитирований в Google Scholar
Дональд Кнут назвал его **"Лучший алгоритм 1973 года"**.

# Введение

## История

**Суффиксное дерево** было создано Питером Вайнером в 1973 году. Первое название – position tree.
$>$ 92 тыс. цитирований в Google Scholar
Дональд Кнут назвал его **"Лучший алгоритм 1973 года"**.
Как Вы думаете в чем схожесть Вайнера и PSY?

# Введение

## История

**Суффиксное дерево** было создано Питером Вайнером в 1973 году. Первое название – position tree.
$>$ 92 тыс. цитирований в Google Scholar
Дональд Кнут назвал его **"Лучший алгоритм 1973 года"**.
Как Вы думаете в чем схожесть Вайнера и PSY?
Они оба авторы одного хита

# Основные понятия

Пусть дан текст $\mathrm{T} = t_1 t_2 \ldots t_n$ и паттерн $\mathrm{P} = p_1 \ldots p_n$

# Основные понятия

Пусть дан текст T $= t_1 t_2 \ldots t_n$ и паттерн P $= p_1 \ldots p_n$

## Определение

**Суффиксное дерево** (suffix tree ST) – сжатый бор, построенный на всех суффиксах строки T\$.

# Основные понятия

Пусть дан текст T $= t_1 t_2 \ldots t_n$ и паттерн P $= p_1 \ldots p_n$

## Определение

**Суффиксное дерево** (suffix tree ST) – сжатый бор, построенный на всех суффиксах строки T\$.

Будем хранить на ребрах вместо подстрок их индексы в исходной строке $T[i....j]$

# Основные понятия

Пусть дан текст T $= t_1 t_2 \ldots t_n$ и паттерн P $= p_1 \ldots p_n$

## Определение

**Суффиксное дерево** (suffix tree ST) – сжатый бор, построенный на всех суффиксах строки T\$.

Будем хранить на ребрах вместо подстрок их индексы в исходной строке $T[i....j]$

**Таким образом:**

# Основные понятия

Пусть дан текст T $= t_1 t_2 \ldots t_n$ и паттерн P $= p_1 \ldots p_n$

## Определение

**Суффиксное дерево** (suffix tree ST) – сжатый бор, построенный на всех суффиксах строки T\$.

Будем хранить на ребрах вместо подстрок их индексы в исходной строке $T[i....j]$

**Таким образом:**

- Ни один суффикс в ST не может полностью лежать в другом

# Основные понятия

Пусть дан текст T $= t_1 t_2 \ldots t_n$ и паттерн P $= p_1 \ldots p_n$

## Определение

**Суффиксное дерево** (suffix tree ST) – сжатый бор, построенный на всех суффиксах строки T\$.

Будем хранить на ребрах вместо подстрок их индексы в исходной строке $T[i....j]$

**Таким образом:**

- Ни один суффикс в ST не может полностью лежать в другом
- Расходуется $O(n)$ памяти

# Построение суффиксного дерева

## Наивный подход

### Online подход:

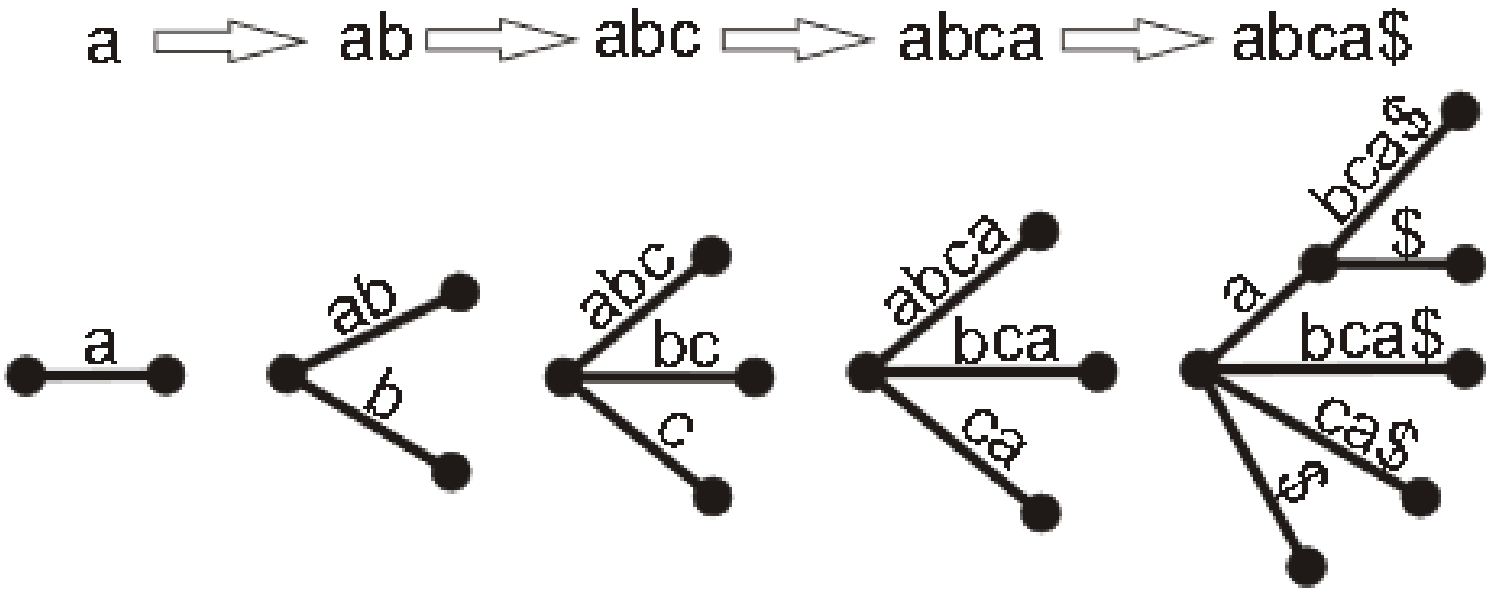
Будем строить суффиксное дерево итеративно, добавляя на каждом шаге по одному символу текста T. То есть будем строить ST для всех префиксов текста T.

# Построение суффиксного дерева

## Наивный подход

### Online подход:

Будем строить суффиксное дерево итеративно, добавляя на каждом шаге по одному символу текста T. То есть будем строить ST для всех префиксов текста T.



## Псевдокод

```
for i = 1 .. n
    for j = 1 .. i
        treeExtend(s[j..i])
```

## Псевдокод

```
for i = 1 .. n
    for j = 1 .. i
        treeExtend(s[j..i])
```

Какова его асимптотика?

## Псевдокод

```
for i = 1 .. n
    for j = 1 .. i
        treeExtend(s[j..i])
```

Какова его асимптотика? $O(n^3)$

## Псевдокод

```
for i = 1 .. n
    for j = 1 .. i
        treeExtend(s[j..i])
```

Какова его асимптотика? $O(n^3)$

А зачем брать по одному символу? Нельзя ли брать сразу весь суффикс?

## Псевдокод

```
for i = 1 .. n
    for j = 1 .. i
        treeExtend(s[j..i])
```

Какова его асимптотика? $O(n^3)$

А зачем брать по одному символу? Нельзя ли брать сразу весь суффикс?

Можно, но это уже **совсем другая история...**

# Виды продлений

# Виды продлений

- **Продление листа.** Суффикс $s[k...i-1]$ заканчивается в листе. Добавим $s_i$ в конец подстроки, которая лежит на ребре, ведущем в лист.

# Виды продлений

- **Продление листа.** Суффикс $s[k...i-1]$ заканчивается в листе. Добавим $s_i$ в конец подстроки, которая лежит на ребре, ведущем в лист.
- **Ответвление**

# Виды продлений

- **Продление листа.** Суффикс $s[k...i-1]$ заканчивается в листе. Добавим $s_i$ в конец подстроки, которая лежит на ребре, ведущем в лист.
- **Ответвление**
  - Суффикс $s[k...i-1]$ заканчивается в вершине (не листе), из которой нет пути по символу $s_i$. Создаем новый лист и на ребре пишем символ $s_i$.

# Виды продлений

- **Продление листа.** Суффикс $s[k...i-1]$ заканчивается в листе. Добавим $s_i$ в конец подстроки, которая лежит на ребре, ведущем в лист.
- **Ответвление**
  - Суффикс $s[k...i-1]$ заканчивается в вершине (не листе), из которой нет пути по символу $s_i$. Создаем новый лист и на ребре пишем символ $s_i$.
  - Суффикс $s[k...i-1]$ заканчивается на ребре с путевой меткой $s[L...R]$ в позиции p - 1 и $s_p \neq s_i$.
    Разобьем ребро новой вершиной v на $s[L...p-1]$ и $s[p...R]$.
    Подвесим к вершине v нового ребенка с дугой, помеченной символом $s_i$.

# Виды продлений

- **Продление листа.** Суффикс $s[k...i-1]$ заканчивается в листе. Добавим $s_i$ в конец подстроки, которая лежит на ребре, ведущем в лист.
- **Ответвление**
  - Суффикс $s[k...i-1]$ заканчивается в вершине (не листе), из которой нет пути по символу $s_i$. Создаем новый лист и на ребре пишем символ $s_i$.
  - Суффикс $s[k...i-1]$ заканчивается на ребре с путевой меткой $s[L...R]$ в позиции p - 1 и $s_p \neq s_i$.
    Разобьем ребро новой вершиной v на $s[L...p-1]$ и $s[p...R]$.
    Подвесим к вершине v нового ребенка с дугой, помеченной символом $s_i$.
- **Ничего не делаем** Для суффикса $s[k...i-1]$ уже есть путь по символу $s_i$.

# Первые улучшения

## Определения

**Неявное суффиксное дерево** (implicit suffix tree IST) – ST строки T без \$.

# Первые улучшения

## Определения

**Неявное суффиксное дерево** (implicit suffix tree IST) – ST строки T без \$.

**i-ая фаза алгоритма** – продолжение всех суффиксов символом $t_i$.

# Первые улучшения

## Определения

**Неявное суффиксное дерево** (implicit suffix tree IST) – ST строки T без \$.

**i-ая фаза алгоритма** – продолжение всех суффиксов символом $t_i$.

Погрузимся в теорию...

# Необходимые теоремы

Количество листьев в сжатом суффиксном дереве $O(n)$

# Необходимые теоремы

Количество листьев в сжатом суффиксном дереве $O(n)$

## Теорема

***Количество внутренних вершин в сжатом суффиксном дереве меньше количества листьев***

# Необходимые теоремы

Количество листьев в сжатом суффиксном дереве $O(n)$

## Теорема

***Количество внутренних вершин в сжатом суффиксном дереве меньше количества листьев***

## Доказательство

# Необходимые теоремы

Количество листьев в сжатом суффиксном дереве $O(n)$

## Теорема

***Количество внутренних вершин в сжатом суффиксном дереве меньше количества листьев***

## Доказательство

- *индукция по n (количество вершин в дереве)*

# Необходимые теоремы

Количество листьев в сжатом суффиксном дереве $O(n)$

## Теорема

***Количество внутренних вершин в сжатом суффиксном дереве меньше количества листьев***

## Доказательство

- *индукция по n (количество вершин в дереве)*
- *База n = 2: очевидно*

# Необходимые теоремы

Количество листьев в сжатом суффиксном дереве $O(n)$

## Теорема

***Количество внутренних вершин в сжатом суффиксном дереве меньше количества листьев***

## Доказательство

- *индукция по n (количество вершин в дереве)*
- *База $n = 2$: очевидно*
- *Рассмотрим дерево на $n + 1$ вершине и найдем вершину, которая имеет не менее двух детей-листьев.*

# Необходимые теоремы

Количество листьев в сжатом суффиксном дереве $O(n)$

## Теорема

***Количество внутренних вершин в сжатом суффиксном дереве меньше количества листьев***

## Доказательство

- *индукция по n (количество вершин в дереве)*
- *База $n = 2$: очевидно*
- *Рассмотрим дерево на $n + 1$ вершине и найдем вершину, которая имеет не менее двух детей-листьев.*
- *Если у этой вершины $> 2$ детей, отрежем одного и применим и.п.*

# Необходимые теоремы

Количество листьев в сжатом суффиксном дереве $O(n)$

## Теорема

***Количество внутренних вершин в сжатом суффиксном дереве меньше количества листьев***

## Доказательство

- *индукция по n (количество вершин в дереве)*
- *База $n = 2$: очевидно*
- *Рассмотрим дерево на $n + 1$ вершине и найдем вершину, которая имеет не менее двух детей-листьев.*
- *Если у этой вершины $> 2$ детей, отрежем одного и применим и.п.*
- *Если у нее ровно 2 ребенка, отрежем обоих и также применим и.п.*

# Первые улучшения

# Первые улучшения

Как же нам ускорять построение дерева?

# Первые улучшения

Как же нам ускорять построение дерева?

Давайте использовать вспомогательные данные

# Первые улучшения

Как же нам ускорять построение дерева?
Давайте использовать вспомогательные данные

## Определение

Пусть $x\alpha$ обозначает произвольную строку, где x – ее первый символ, а $\alpha$ – оставшаяся подстрока (возможно пустая). Если для внутренней вершины v с путевой меткой $x\alpha$ существует другая вершина sl(v) с путевой меткой $\alpha$, то ссылка из v в sl(v) называется **суффиксной ссылкой** (suffix link).

# Первые улучшения

## Лемма

$\forall$ внутренней вершины v $\exists$ суффиксная ссылка ведущая в какую-то внутреннюю вершину u.

# Первые улучшения

## Лемма

$\forall$ внутренней вершины v $\exists$ суффиксная ссылка ведущая в какую-то внутреннюю вершину u.

## Доказательство

*Рассмотрим внутреннюю вершину v с путевой меткой* $s[j \ldots i]$.

# Первые улучшения

## Лемма

$\forall$ внутренней вершины v $\exists$ суффиксная ссылка ведущая в какую-то внутреннюю вершину u.

## Доказательство

*Рассмотрим внутреннюю вершину v с путевой меткой $s[j \ldots i]$. Так как эта вершина внутренняя, её путевая метка ветвится справа в исходной строке.*

# Первые улучшения

## Лемма

$\forall$ внутренней вершины v $\exists$ суффиксная ссылка ведущая в какую-то внутреннюю вершину u.

## Доказательство

*Рассмотрим внутреннюю вершину v с путевой меткой $s[j...i]$. Так как эта вершина внутренняя, её путевая метка ветвится справа в исходной строке. Тогда подстрока $s[j+1...i]$ тоже ветвится справа в исходной строке, и ей соответствует некоторая внутренняя вершина u. По определению суффиксная ссылка вершины v ведёт в u.*

# Первые улучшения

**Задача:** научится эффективно переходить между суффиксами в фазах алгоритма

# Первые улучшения

**Задача:** научится эффективно переходить между суффиксами в фазах алгоритма

## Использование суф. ссылок

Пусть был продлен суффикс $s[j...i-1]$ до суффикса $s[j...i]$.
Теперь надо продлить суффикс $s[j+1...i-1]$.
Как быстро найти этот суффикс?

# Первые улучшения

**Задача:** научится эффективно переходить между суффиксами в фазах алгоритма

## Использование суф. ссылок

Пусть был продлен суффикс $s[j...i-1]$ до суффикса $s[j...i]$.
Теперь надо продлить суффикс $s[j+1...i-1]$.
Как быстро найти этот суффикс? Пройдем вверх по дереву до ближайшей внутренней вершины v, перейдем по ее суффиксной ссылке в некоторую вершину u и спустимся по ребрам до нужного суффикса.

# Первые улучшения

**Задача:** научится эффективно переходить между суффиксами в фазах алгоритма

## Использование суф. ссылок

Пусть был продлен суффикс $s[j...i-1]$ до суффикса $s[j...i]$.
Теперь надо продлить суффикс $s[j+1...i-1]$.
Как быстро найти этот суффикс? Пройдем вверх по дереву до ближайшей внутренней вершины v, перейдем по ее суффиксной ссылке в некоторую вершину u и спустимся по ребрам до нужного суффикса.

Слишком

# Первые улучшения

**Задача:** научится эффективно переходить между суффиксами в фазах алгоритма

## Использование суф. ссылок

Пусть был продлен суффикс $s[j...i-1]$ до суффикса $s[j...i]$.
Теперь надо продлить суффикс $s[j+1...i-1]$.
Как быстро найти этот суффикс? Пройдем вверх по дереву до ближайшей внутренней вершины v, перейдем по ее суффиксной ссылке в некоторую вершину u и спустимся по ребрам до нужного суффикса.

Слишком много

# Первые улучшения

**Задача:** научится эффективно переходить между суффиксами в фазах алгоритма

## Использование суф. ссылок

Пусть был продлен суффикс $s[j...i-1]$ до суффикса $s[j...i]$.
Теперь надо продлить суффикс $s[j+1...i-1]$.
Как быстро найти этот суффикс? Пройдем вверх по дереву до ближайшей внутренней вершины v, перейдем по ее суффиксной ссылке в некоторую вершину u и спустимся по ребрам до нужного суффикса.

Слишком много буков.

# Первые улучшения

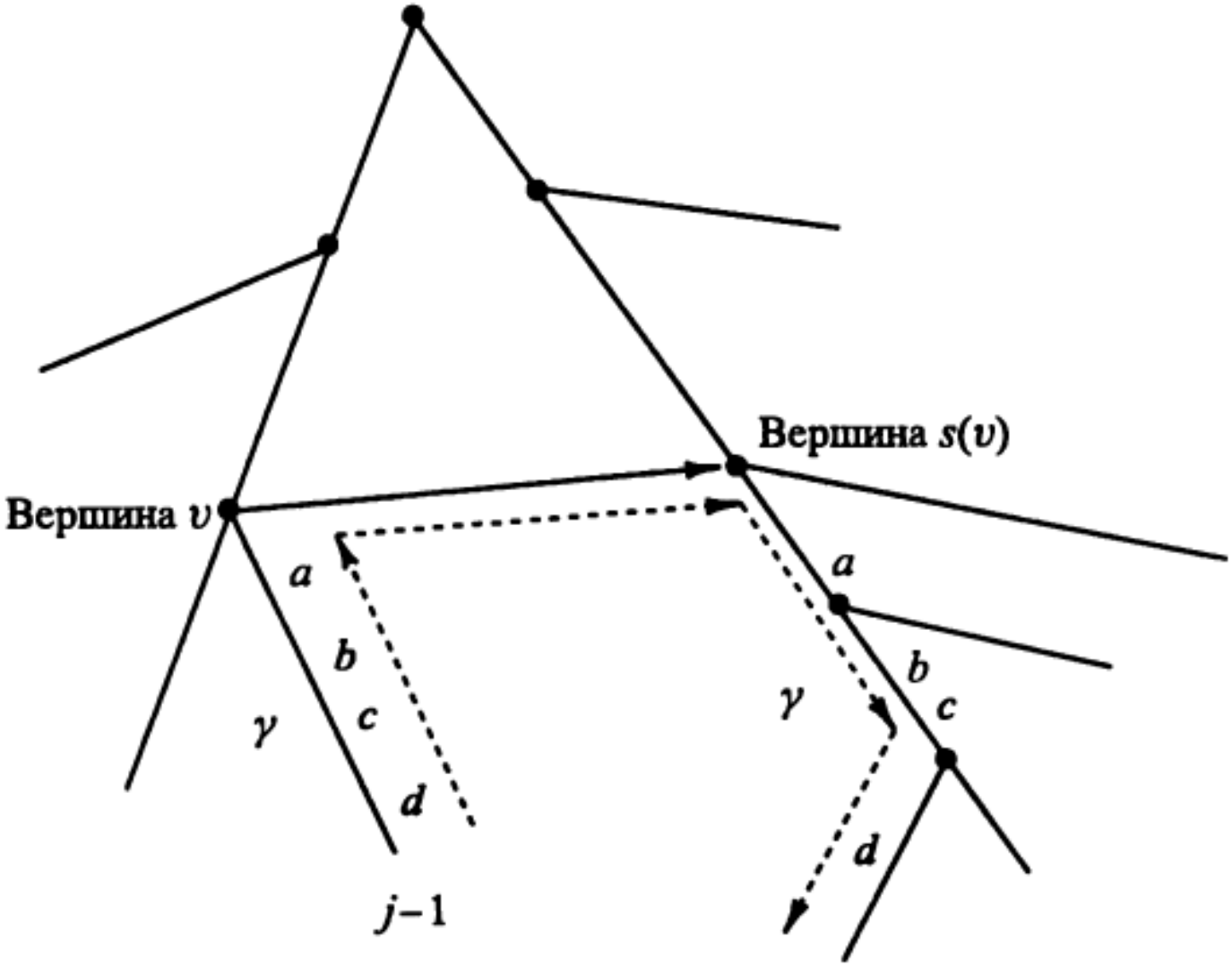
**Задача:** научится эффективно переходить между суффиксами в фазах алгоритма

## Использование суф. ссылок

Пусть был продлен суффикс $s[j...i-1]$ до суффикса $s[j...i]$.
Теперь надо продлить суффикс $s[j+1...i-1]$.
Как быстро найти этот суффикс? Пройдем вверх по дереву до ближайшей внутренней вершины v, перейдем по ее суффиксной ссылке в некоторую вершину u и спустимся по ребрам до нужного суффикса.

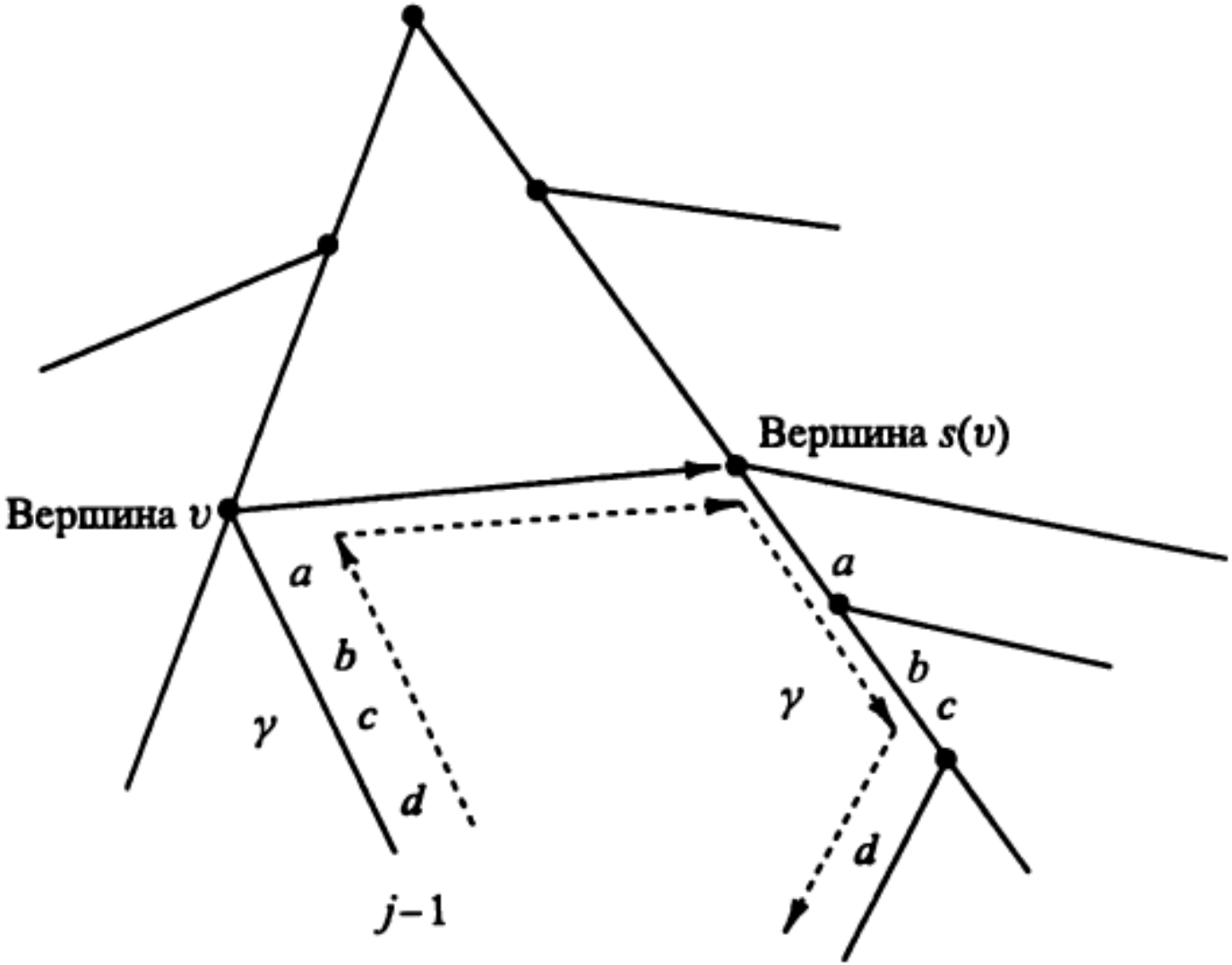
Слишком много буков. Обратимся к картинке

# Первые улучшения




А почему это быстро?

# Первые улучшения




А почему это быстро? Давайте узнаем

# Обоснования улучшений

**Глубина вершины** – количество ребер на пути от корня до этой вершины.

## Лемма 1

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

## Лемма 2

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

## Доказательство

# Обоснования улучшений

**Глубина вершины** – количество ребер на пути от корня до этой вершины.

## Лемма 1

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

## Лемма 2

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

## Доказательство

- *Проход вверх по ребру уменьшает глубину на 1*

# Обоснования улучшений

**Глубина вершины** – количество ребер на пути от корня до этой вершины.

## Лемма 1

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

## Лемма 2

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

## Доказательство

- *Проход вверх по ребру уменьшает глубину на 1*
- *Проход по суффиксной ссылке уменьшает глубину не более чем на 1*

# Обоснования улучшений

**Глубина вершины** – количество ребер на пути от корня до этой вершины.

## Лемма 1

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

## Лемма 2

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

## Доказательство

- *Проход вверх по ребру уменьшает глубину на 1*
- *Проход по суффиксной ссылке уменьшает глубину не более чем на 1*
- *Поэтому за фазу i совершается не более 2i переходов вверх*

# Обоснования улучшений

**Глубина вершины** – количество ребер на пути от корня до этой вершины.

## Лемма 1

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

## Лемма 2

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

## Доказательство

- *Проход вверх по ребру уменьшает глубину на 1*
- *Проход по суффиксной ссылке уменьшает глубину не более чем на 1*
- *Поэтому за фазу i совершается не более 2i переходов вверх*
- *Заметим также, что в одной фазе начальная глубина меньше конечной, так как длины суффиксов уменьшаются*

# Обоснования улучшений

**Глубина вершины** – количество ребер на пути от корня до этой вершины.

## Лемма 1

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

## Лемма 2

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

## Доказательство

- *Проход вверх по ребру уменьшает глубину на 1*
- *Проход по суффиксной ссылке уменьшает глубину не более чем на 1*
- *Поэтому за фазу i совершается не более 2i переходов вверх*
- *Заметим также, что в одной фазе начальная глубина меньше конечной, так как длины суффиксов уменьшаются*
- *Следовательно вниз мы пошли тоже не более 2i раз*

# Обоснования улучшений

**Глубина вершины** – количество ребер на пути от корня до этой вершины.

## Лемма 1

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

## Лемма 2

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

## Доказательство

- *Проход вверх по ребру уменьшает глубину на 1*
- *Проход по суффиксной ссылке уменьшает глубину не более чем на 1*
- *Поэтому за фазу i совершается не более 2i переходов вверх*
- *Заметим также, что в одной фазе начальная глубина меньше конечной, так как длины суффиксов уменьшаются*
- *Следовательно вниз мы пошли тоже не более 2i раз*

Поскольку всего n фаз и на каждой мы делаем $O(n)$ переходов, мы получили асимптотику $O(n^2)$

# Финальная версия

## Ключевые леммы

# Финальная версия

## Ключевые леммы

### Лемма (Стал листом — листом и останешься):

Если в какой-то момент был создан лист с меткой i, он останется листом во всех последовательных деревьях, созданных алгоритмом.

# Финальная версия

## Ключевые леммы

### Лемма (Стал листом — листом и останешься):

Если в какой-то момент был создан лист с меткой i, он останется листом во всех последовательных деревьях, созданных алгоритмом.

### Доказательство

- *Внимательно прочитав правила продления, мы увидим, что у нас нет правила продолжения листового ребра дальше текущего листа.*

# Финальная версия

## Ключевые леммы

### Лемма (Стал листом — листом и останешься):

Если в какой-то момент был создан лист с меткой i, он останется листом во всех последовательных деревьях, созданных алгоритмом.

### Доказательство

- *Внимательно прочитав правила продления, мы увидим, что у нас нет правила продолжения листового ребра дальше текущего листа.*
- *Если есть лист с суффиксом i, то правило 1 будет применяться для продолжения i на всех остальных фазах.*

# Финальная версия

## Ключевые леммы

### Лемма (правило 3 заканчивает дело):

Если правило продления 3 применяется в продолжении суффикса, начинающегося в позиции $j$, это же правило будет применяться ко всем остальным суффиксам $(j + 1...i)$ до конца фазы.

# Финальная версия

## Ключевые леммы

### Лемма (правило 3 заканчивает дело):

Если правило продления 3 применяется в продолжении суффикса, начинающегося в позиции $j$, это же правило будет применяться ко всем остальным суффиксам $(j + 1...i)$ до конца фазы.

### Доказательство

# Финальная версия

## Ключевые леммы

### Лемма (правило 3 заканчивает дело):

Если правило продления 3 применяется в продолжении суффикса, начинающегося в позиции $j$, это же правило будет применяться ко всем остальным суффиксам $(j+1...i)$ до конца фазы.

### Доказательство

- *При продолжении по правилу 3 путь помеченный суффиксом* $s[j...i-1]$ *должен продолжаться символом* $s_i$.

# Финальная версия

## Ключевые леммы

### Лемма (правило 3 заканчивает дело):

Если правило продления 3 применяется в продолжении суффикса, начинающегося в позиции $j$, это же правило будет применяться ко всем остальным суффиксам $(j+1...i)$ до конца фазы.

### Доказательство

- *При продолжении по правилу 3 путь помеченный суффиксом* $s[j...i-1]$ *должен продолжаться символом* $s_i$.
- *Точно так же продолжаются все остальные пути помеченные* $s[j+1...i-1], s[j+2...i-1]$ *и тд.*

# Финальная версия

## Алгоритм

- Храним в листьях переменную x и неявно продлеваем их за $O(1)$.

# Финальная версия

## Алгоритм

- Храним в листьях переменную x и неявно продлеваем их за $O(1)$.
- Алгоритм явно работает с суффиксами в диапазоне от $j^*$ до $k$, $k \leq i$.

# Финальная версия

## Алгоритм

- Храним в листьях переменную x и неявно продлеваем их за $O(1)$.
- Алгоритм явно работает с суффиксами в диапазоне от $j^*$ до $k$, $k \leq i$.
- Действительно, если суффикс $s[j...i-2]$ был продлён до суффикса $s[j...i-1]$ на прошлой фазе по правилу 1, то он и дальше будет продлеваться по правилу 1.

# Финальная версия

## Алгоритм

- Храним в листьях переменную x и неявно продлеваем их за $O(1)$.
- Алгоритм явно работает с суффиксами в диапазоне от $j^*$ до $k$, $k \leq i$.
- Действительно, если суффикс $s[j...i-2]$ был продлён до суффикса $s[j...i-1]$ на прошлой фазе по правилу 1, то он и дальше будет продлеваться по правилу 1.
- Если он был продлён по правилу 2, то была создана новая листовая вершина, значит, на текущей фазе i этот суффикс будет продлён по листу.

# Финальная версия

## Алгоритм

- Храним в листьях переменную x и неявно продлеваем их за $O(1)$.
- Алгоритм явно работает с суффиксами в диапазоне от $j^*$ до $k$, $k \leq i$.
- Действительно, если суффикс $s[j...i-2]$ был продлён до суффикса $s[j...i-1]$ на прошлой фазе по правилу 1, то он и дальше будет продлеваться по правилу 1.
- Если он был продлён по правилу 2, то была создана новая листовая вершина, значит, на текущей фазе i этот суффикс будет продлён по листу.
- Следовательно после применения правила 3 на суффиксе $s[k...i]$ текущую фазу можно завершить, а следующую начать с $j^* = k - 1$.

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

1. for c in s:

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

1. for c in s:
2. while True:

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

1. for c in s:
2. while True:
3. **if** есть переход из cur по c:

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

```
1. for c in s:
2.         while True:
3.                 if есть переход из cur по c:
4.                         cur = переход из cur по c
```

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

```
1. for c in s:
2.         while True:
3.                 if есть переход из cur по c:
4.                         cur = переход из cur по c
5.                         break
```

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

```
1. for c in s:
2.         while True:
3.                 if есть переход из cur по c:
4.                         cur = переход из cur по c
5.                         break
6.                 else:
```

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

```
1. for c in s:
2.         while True:
3.                 if есть переход из cur по c:
4.                         cur = переход из cur по c
5.                         break
6.                 else:
7.                         Сделать переход из cur по c
```

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

```
1. for c in s:
2.         while True:
3.             if есть переход из cur по c:
4.                 cur = переход из cur по c
5.                 break
6.             else:
7.                 Сделать переход из cur по c
8.                 Дописываем туда все символы //лист
```

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

```
1. for c in s:
2.         while True:
3.                 if есть переход из cur по c:
4.                         cur = переход из cur по c
5.                         break
6.                 else:
7.                         Сделать переход из cur по c
8.                         Дописываем туда все символы //лист
9.                         cur = переход из cur по суф. ссылке
```

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

```
1.  for c in s:
2.      while True:
3.          if есть переход из cur по c:
4.              cur = переход из cur по c
5.              break
6.          else:
7.              Сделать переход из cur по c
8.              Дописываем туда все символы //лист
9.              cur = переход из cur по суф. ссылке
10.             prev.sufflink = cur
```

# Линейная оценка

Пусть cur это наша текущая вершина

## Псевдокод

```
1.  for c in s:
2.      while True:
3.          if есть переход из cur по c:
4.              cur = переход из cur по c
5.              break
6.          else:
7.              Сделать переход из cur по c
8.              Дописываем туда все символы //лист
9.              cur = переход из cur по суф. ссылке
10.             prev.sufflink = cur
```

Суммарное время работы алгоритма $O(n)$

# Источники


- Дэн Гасфилд — Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология — СПб.: Невский Диалект; БХВ-Петербург, 2003. — 654 с: ил.
- **Коспект Юрия Лифшица**
- Викиконспекты
- **Видеолекции Павла Маврина**
- **Habr**