# Построение прямолинейных программ с помощью декартовых деревьев

Евгений Курпилянский

20 июня 2012 года

С ростом размера входных данных для классических задач меняются алгоритмы, способные их эффективно решать.

Существует несколько подходов, например:

- Алгоритмы эффективного ввода-вывода – алгоритмы, минимизирующие чтение данных с жесткого диска.
- Сокращение размера входа за счет предварительной обработки.

С ростом размера входных данных для классических задач меняются алгоритмы, способные их эффективно решать.

Существует несколько подходов, например:

- Алгоритмы эффективного ввода-вывода – алгоритмы, минимизирующие чтение данных с жесткого диска.
- Сокращение размера входа за счет предварительной обработки.

С ростом размера входных данных для классических задач меняются алгоритмы, способные их эффективно решать.

Существует несколько подходов, например:

- Алгоритмы эффективного ввода-вывода – алгоритмы, минимизирующие чтение данных с жесткого диска.
- Сокращение размера входа за счет предварительной обработки.

## Способы сжатия

Существует различные способы сжатия данных, например:

- прямолинейные программы (ПП);
- антисловари;
- коллаж-системы и др.

Если сжатое представление хорошо структурировано, то существуют алгоритмы, способные решать классические задачи без распаковки данных.

# Способы сжатия

Существует различные способы сжатия данных, например:

- прямолинейные программы (ПП);
- антисловари;
- коллаж-системы и др.

Если сжатое представление хорошо структурировано, то существуют алгоритмы, способные решать классические задачи без распаковки данных.

# Определение прямолинейной программы

## Определение

**Прямолинейная программа** (ПП) строки $S$ – это контекстно-свободная грамматика в нормальной форме Хомского, выводящая в точности одно слово $S$.

## Пример

Рассмотрим ПП $X$, выводящую строку «*abaababaabaab*».

$$X_1 = b$$
$$X_2 = a$$
$$X_3 = X_2 \cdot X_1$$
$$X_4 = X_3 \cdot X_2$$
$$X_5 = X_4 \cdot X_3$$
$$X_6 = X_5 \cdot X_4$$
$$X_7 = X_6 \cdot X_5$$

# Определение прямолинейной программы

## Определение

**Прямолинейная программа** (ПП) строки $S$ – это контекстно-свободная грамматика в нормальной форме Хомского, выводящая в точности одно слово $S$.

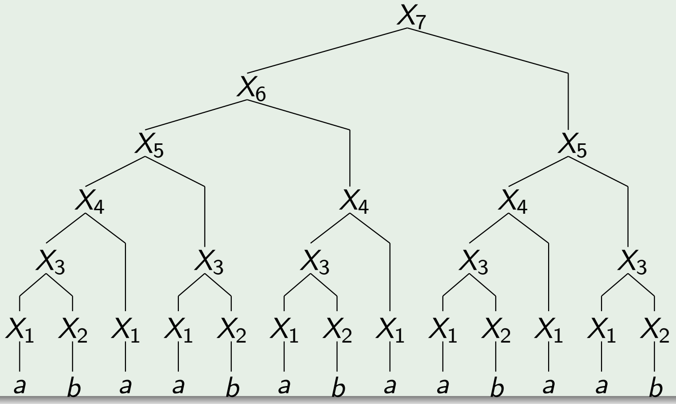
## Пример

Рассмотрим ПП $X$, выводящую строку «*abaababaabaab*».

$$
\begin{aligned}
X_1 &= b \\
X_2 &= a \\
X_3 &= X_2 \cdot X_1 \\
X_4 &= X_3 \cdot X_2 \\
X_5 &= X_4 \cdot X_3 \\
X_6 &= X_5 \cdot X_4 \\
X_7 &= X_6 \cdot X_5
\end{aligned}
$$

## Пример

Графическое изображение ПП:



# Как строить ПП?

**Утверждение.**

Задача построения минимальной ПП, выводящей заданную строку $S$ – NP-трудная.

$\Downarrow$

Для построения ПП требуется использовать приближенные алгоритмы.

# Как строить ПП?

**Утверждение.**

Задача построения минимальной ПП, выводящей заданную строку $S$ – NP-трудная.

$$\Downarrow$$

Для построения ПП требуется использовать приближенные алгоритмы.

### Определение

**Факторизация** строки $S$ – это набор строк $w_1, w_2, \ldots, w_k$ такой, что $S = w_1 \cdot w_2 \cdot \ldots \cdot w_k$.

### Определение

$LZ$-**факторизация** строки $S$ — это факторизация $S = w_1 \cdot w_2 \cdots w_k$ такая, что для любого $j \in 1..k$

- $w_j$ состоит из одной буквы, не встречающейся в $w_1 \cdot w_2 \cdots w_{j-1}$; или
- $w_j$ — наибольший префикс $w_j \cdot w_{j+1} \cdots w_k$, встречающийся в $w_1 \cdot w_2 \cdots w_{j-1}$.

### Факторизации строки «*abaababaabaab*»

- $a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot a \cdot b$;
- $a \cdot b \cdot a \cdot aba \cdot baaba \cdot ab$;

### Определение

**Факторизация** строки $S$ – это набор строк $w_1, w_2, \ldots, w_k$ такой, что $S = w_1 \cdot w_2 \cdot \ldots \cdot w_k$.

### Определение

$LZ$-**факторизация** строки $S$ — это факторизация $S = w_1 \cdot w_2 \cdots w_k$ такая, что для любого $j \in 1..k$

- $w_j$ состоит из одной буквы, не встречающейся в $w_1 \cdot w_2 \cdots w_{j-1}$; или
- $w_j$ — наибольший префикс $w_j \cdot w_{j+1} \cdots w_k$, встречающийся в $w_1 \cdot w_2 \cdots w_{j-1}$.

### Факторизации строки «*abaababaabaab*»

- $a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot a \cdot b$;
- $a \cdot b \cdot a \cdot aba \cdot baaba \cdot ab$;

### Определение

**Факторизация** строки $S$ – это набор строк $w_1, w_2, \ldots, w_k$ такой, что $S = w_1 \cdot w_2 \cdot \ldots \cdot w_k$.

### Определение

$LZ$-**факторизация** строки $S$ — это факторизация $S = w_1 \cdot w_2 \cdots w_k$ такая, что для любого $j \in 1..k$

- $w_j$ состоит из одной буквы, не встречающейся в $w_1 \cdot w_2 \cdots w_{j-1}$; или
- $w_j$ — наибольший префикс $w_j \cdot w_{j+1} \cdots w_k$, встречающийся в $w_1 \cdot w_2 \cdots w_{j-1}$.

### Факторизации строки «*abaababaabaab*»

- $a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot a \cdot b$;
- $a \cdot b \cdot a \cdot aba \cdot baaba \cdot ab$;

### Постановка задачи

Вход: Строка $T$ и ее $LZ$-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку $T$.

### Алгоритм Риттера

- Дерево вывода ПП представляется в виде $AVL$-дерева.
- Предложены алгоритмы конкатенации двух AVL-деревьев и вырезания подстроки из AVL-дерева.
- Размер полученной ПП всего в $O(\log n)$ больше размера минимальной ПП

## Постановка задачи

Вход: Строка $T$ и ее $LZ$-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку $T$.

## Алгоритм Риттера

- Дерево вывода ПП представляется в виде $AVL$-дерева.
- Предложены алгоритмы конкатенации двух AVL-деревьев и вырезания подстроки из AVL-дерева.
- Размер полученной ПП всего в $O(\log n)$ больше размера минимальной ПП

## Постановка задачи

Вход: Строка $T$ и ее $LZ$-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку $T$.

## Алгоритм Риттера

- Дерево вывода ПП представляется в виде $AVL$-дерева.
- Предложены алгоритмы конкатенации двух AVL-деревьев и вырезания подстроки из AVL-дерева.
- Размер полученной ПП всего в $O(\log n)$ больше размера минимальной ПП

## Постановка задачи

Вход: Строка $T$ и ее $LZ$-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку $T$.

## Алгоритм Риттера

- Дерево вывода ПП представляется в виде $AVL$-дерева.
- Предложены алгоритмы конкатенации двух AVL-деревьев и вырезания подстроки из AVL-дерева.
- Размер полученной ПП всего в $O(\log n)$ больше размера минимальной ПП

# Построение ПП с помощью декартовых деревьев

## Мысль

Почему бы для построения ПП не использовать другое сбалансированное двоичное дерево?

## Декартово дерево

- Декартово дерево – двоичное дерево поиска, в каждой вершине которого хранится ее приоритет. При этом всегда выполняется свойство: приоритет в потомках меньше, чем приоритет самой вершины.
- Доказано, что если приоритеты выбираются случайно, высота декартова дерева $O(\log n)$.
- Над декартовым деревом определены две стандартные операции: конкатенация и разрезание.

# Построение ПП с помощью декартовых деревьев

## Мысль

Почему бы для построения ПП не использовать другое сбалансированное двоичное дерево?

## Декартово дерево

- Декартово дерево – двоичное дерево поиска, в каждой вершине которого хранится ее приоритет. При этом всегда выполняется свойство: приоритет в потомках меньше, чем приоритет самой вершины.
- Доказано, что если приоритеты выбираются случайно, высота декартова дерева $O(\log n)$.
- Над декартовым деревом определены две стандартные операции: конкатенация и разрезание.

# Построение ПП с помощью декартовых деревьев

## Мысль

Почему бы для построения ПП не использовать другое сбалансированное двоичное дерево?

## Декартово дерево

- Декартово дерево – двоичное дерево поиска, в каждой вершине которого хранится ее приоритет. При этом всегда выполняется свойство: приоритет в потомках меньше, чем приоритет самой вершины.
- Доказано, что если приоритеты выбираются случайно, высота декартова дерева $O(\log n)$.
- Над декартовым деревом определены две стандартные операции: конкатенация и разрезание.

# Построение ПП с помощью декартовых деревьев

## Мысль

Почему бы для построения ПП не использовать другое сбалансированное двоичное дерево?

## Декартово дерево

- Декартово дерево – двоичное дерево поиска, в каждой вершине которого хранится ее приоритет. При этом всегда выполняется свойство: приоритет в потомках меньше, чем приоритет самой вершины.
- Доказано, что если приоритеты выбираются случайно, высота декартова дерева $O(\log n)$.
- Над декартовым деревом определены две стандартные операции: конкатенация и разрезание.

# Построение ПП с помощью декартовых деревьев

## Модификации декартовых деревьев

Были исследованы следующие модификации:

- Рандомизированные двоичные деревья поиска **(C. Martinez, S. Roura, 1998)**
- Декартовы деревья по неявному ключу **(Н. Дуров, А. Лопатин, 2002)**
- Персистентные деревья

## Новая модификация

Основная структура данных алгоритма построения ПП – новая модификация декартовых деревьев, которая совмещает в себе идеи всех трех перечисленных модификаций.

# Построение ПП с помощью декартовых деревьев

## Модификации декартовых деревьев

Были исследованы следующие модификации:

- Рандомизированные двоичные деревья поиска **(C. Martinez, S. Roura, 1998)**
- Декартовы деревья по неявному ключу **(Н. Дуров, А. Лопатин, 2002)**
- Персистентные деревья

## Новая модификация

Основная структура данных алгоритма построения ПП – новая модификация декартовых деревьев, которая совмещает в себе идеи всех трех перечисленных модификаций.

# Построение ПП с помощью декартовых деревьев

## Модификации декартовых деревьев

Были исследованы следующие модификации:

- Рандомизированные двоичные деревья поиска **(C. Martinez, S. Roura, 1998)**
- Декартовы деревья по неявному ключу **(Н. Дуров, А. Лопатин, 2002)**
- Персистентные деревья

## Новая модификация

Основная структура данных алгоритма построения ПП – новая модификация декартовых деревьев, которая совмещает в себе идеи всех трех перечисленных модификаций.

# Построение ПП с помощью декартовых деревьев

## Модификации декартовых деревьев

Были исследованы следующие модификации:

- Рандомизированные двоичные деревья поиска **(C. Martinez, S. Roura, 1998)**
- Декартовы деревья по неявному ключу **(Н. Дуров, А. Лопатин, 2002)**
- Персистентные деревья

## Новая модификация

Основная структура данных алгоритма построения ПП – новая модификация декартовых деревьев, которая совмещает в себе идеи всех трех перечисленных модификаций.

# Рандомизированные ПП

## Определение

Дерево вывода некоторой ПП будем называть **рандомизированным**, если оно удовлетворяет одному из следующих условий:

- Это дерево состоит ровно из одного узла и в нем хранится, выводимый терминал.
- Оба поддерева являются независимыми рандомизированными деревьями, и размер левого дерева может быть любым с равной вероятностью.

## Результаты

- Было доказано, что высота рандомизированного дерево вывода в среднем $O(\log n)$.
- Были модифицированы операции конкатенации и разрезания.

# Рандомизированные ПП

## Определение

Дерево вывода некоторой ПП будем называть **рандомизированным**, если оно удовлетворяет одному из следующих условий:

- Это дерево состоит ровно из одного узла и в нем хранится, выводимый терминал.
- Оба поддерева являются независимыми рандомизированными деревьями, и размер левого дерева может быть любым с равной вероятностью.

## Результаты

- Было доказано, что высота рандомизированного дерево вывода в среднем $O(\log n)$.
- Были модифицированы операции конкатенации и разрезания.

# Рандомизированные ПП

## Определение

Дерево вывода некоторой ПП будем называть **рандомизированным**, если оно удовлетворяет одному из следующих условий:

- Это дерево состоит ровно из одного узла и в нем хранится, выводимый терминал.
- Оба поддерева являются независимыми рандомизированными деревьями, и размер левого дерева может быть любым с равной вероятностью.

## Результаты

- Было доказано, что высота рандомизированного дерево вывода в среднем $O(\log n)$.
- Были модифицированы операции конкатенации и разрезания.

### Алгоритм построения рандомизированных ПП

Идея алгоритма совпадает с идеей алгоритма Риттера.

### Результат

Было доказано, что, если $n$ – длина строки $S$, а $LZ$-факторизация содержит $k$ факторов, то:

- Размер полученной ПП в $O(\log n)$ раз больше минимальной.
- Время работы алгоритма $O(k \log n)$ и требуемая память $O(k \log n)$.

### Алгоритм построения рандомизированных ПП

Идея алгоритма совпадает с идеей алгоритма Риттера.

### Результат

Было доказано, что, если $n$ – длина строки $S$, а $LZ$-факторизация содержит $k$ факторов, то:

- Размер полученной ПП в $O(\log n)$ раз больше минимальной.
- Время работы алгоритма $O(k \log n)$ и требуемая память $O(k \log n)$.

### Алгоритм построения рандомизированных ПП

Идея алгоритма совпадает с идеей алгоритма Риттера.

### Результат

Было доказано, что, если $n$ – длина строки $S$, а $LZ$-факторизация содержит $k$ факторов, то:

- Размер полученной ПП в $O(\log n)$ раз больше минимальной.
- Время работы алгоритма $O(k \log n)$ и требуемая память $O(k \log n)$.

### Алгоритм построения рандомизированных ПП

Идея алгоритма совпадает с идеей алгоритма Риттера.

### Результат

Было доказано, что, если $n$ – длина строки $S$, а $LZ$-факторизация содержит $k$ факторов, то:

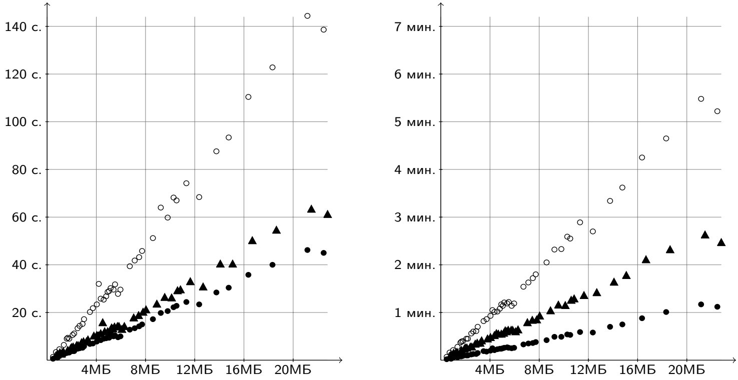
- Размер полученной ПП в $O(\log n)$ раз больше минимальной.
- Время работы алгоритма $O(k \log n)$ и требуемая память $O(k \log n)$.

# Практические результаты

Алгоритмы были протестированы на:

- строках Фибоначчи;
- случайных строках;
- ДНК, взятых с сайта `http://www.ddbj.nig.ac.jp/`.

# Скорость работы на строках ДНК

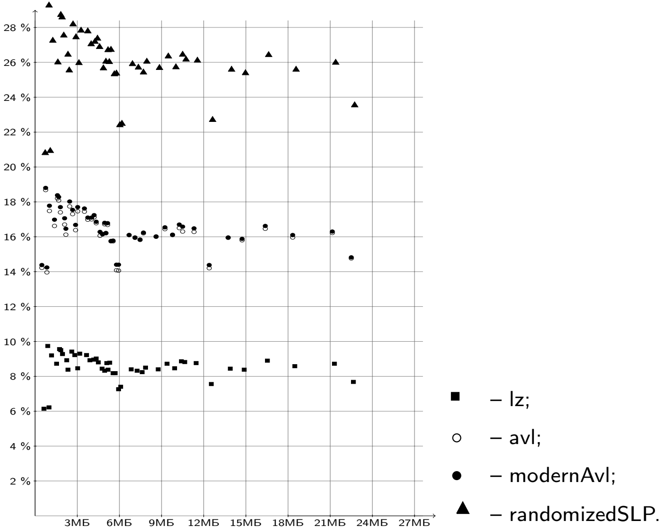



○ – алгоритм Риттера;

● – модернизированный алгоритм Риттера;

▲ – алгоритм построения рандомизированной ПП.

# Коэффициенты сжатия на строках ДНК



# Результаты и дальнейшие планы

## Результаты

- Новый алгоритм построения ПП.
- Практические результаты сравнения алгоритмов по двум параметрам: скорость работы и степень сжатия.

Исходные коды алгоритмов можно посмотреть здесь:
`http://code.google.com/p/overclocking`

## Планы

- Применить эвристику оптимизации порядка конкатенаций в алгоритме построения рандомизированных ПП

# Результаты и дальнейшие планы

## Результаты

- Новый алгоритм построения ПП.
- Практические результаты сравнения алгоритмов по двум параметрам: скорость работы и степень сжатия.

Исходные коды алгоритмов можно посмотреть здесь: `http://code.google.com/p/overclocking`

## Планы

- Применить эвристику оптимизации порядка конкатенаций в алгоритме построения рандомизированных ПП

# Результаты и дальнейшие планы

## Результаты

- Новый алгоритм построения ПП.
- Практические результаты сравнения алгоритмов по двум параметрам: скорость работы и степень сжатия.

Исходные коды алгоритмов можно посмотреть здесь:
`http://code.google.com/p/overclocking`

## Планы

- Применить эвристику оптимизации порядка конкатенаций в алгоритме построения рандомизированных ПП