

# Индексирование текста для поиска с учетом орфографических ошибок


---

Искандер Акишев  
Михаил Дворкин

СПбГУ ИТМО  
Ноябрь 2006 г.

# Часть 0 – Предисловие

---


 [Почта](#) [Помощь](#) [Настроить поиск](#)

Найдётся всё  в найденном  в регионе: Петербург [расширенный поиск](#)

[Везде](#) [Новости](#) [Маркет](#) [Адреса](#) [Словари](#) [Блоги](#) [Картинки](#) [Все службы...](#)

Результат поиска: страниц — **43 620**, сайтов — не менее **504**  
Статистика слов: аффтор — 479 078, жжот — 6 170 530.  
Запросов за месяц: аффтор — 254, жжот — 3 091. [Купить эти слова.](#)

**Опечатка?** возможно, имелось в виду: [«аффтар жжот»](#)

 [Веб](#) [Картинки](#) [Группы](#) [Каталог](#) [Дополнительно »](#)

[Расширенный поиск](#)  
[Настройки](#)

Поиск в Интернете  Поиск страниц на русском

**Веб** Результаты **1 - 10** из примерно **47 600** для **аффтор жжот**. (0,11 секунд)

Возможно, Вы имели в виду: [аффтар жжот](#)

---

# Часть I - Введение

---

- Область применения
  - Постановка задачи
  - Примеры
  - Имеющиеся результаты
-

# Область применения

---

- Необходимость поиска с учетом ошибок:
    - Поиск документов в Интернете
    - Автоматическое исправление орфографических ошибок
    - Вычислительная биология:
      - Поиск образца в неточных экспериментальных данных
      - Поиск похожих участков ДНК
-

# Постановка задачи (1)

---

- Коллекция документов  $T$  суммарного размера  $n$
  - Образец  $P$  длины  $m$
  - Предполагается не более  $d$  ошибок:
    - Пропущенный символ
    - Лишний символ
    - Измененный символ
    - *Можно также сузить на метрику Хэмминга*
-

# Постановка задачи (2)

---

- Требуется найти
    - Все вхождения
    - Все начальные позиции вхождений
    - Все документы, содержащие образец
-

# Пример

---

Документы:

**ГАСТСААААСGGGTGC**

**GTGACCGACGGATGAC**

**ССТАСАААСАТGTTСG**

**ТАААССТGAGACCAAC**

Образец: **АСААС**

Разрешенное число

ошибок:  $d = 1$

---

# Пример

---

Документы:

ГАСТСААААСGGGTGC

GTGACCCGACGGATGAC

ССТААААСАТGTTСG

ТАААССТGAGАССААС

Образец: АСААС

Разрешенное число

ошибок:  $d = 1$

Различные вхождения:

1-й документ: (6, 10), (7, 10)

3-й документ: (4, 7), (4, 8), (4, 9), (6, 9)

4-й документ: (2, 5), (11, 16), (12, 16), (13, 16)

---



# Пример

---

Документы:

ГАСТС[ААААСGGGTGC  
GTGACCGACGGATGAC  
ССТ[АС[АААСАТGTTСG  
Т[АААССТGAG[АС[СААС

Образец: АСААС

Разрешенное число  
ошибок:  $d = 1$

Начальные позиции вхождений:

1-й документ: 6, 7

3-й документ: 4, 6

4-й документ: 2, 11, 12, 13

---

# Пример

---

Документы:

ГАСТС**ААА**СGGGTGC

GTGACCGACGGATGAC

ССТ**АСАА**САТGTTСG

**ТААА**ССТGAG**АС**СААС

Образец: АСААС

Разрешенное число  
ошибок:  $d = 1$

Документы, содержащие образец:  
1, 3, 4

---

# Имеющиеся результаты (1)

| Число ошибок | Время запроса                       | Размер структуры данных | Время индексирования           | Источник              |
|--------------|-------------------------------------|-------------------------|--------------------------------|-----------------------|
| $d = 0$      | $O(m + occ)$                        | $O(n)$                  | $O(n)$                         | Weiner 1973           |
| $d = 1$      | $O(m + \log n \log \log n + occ)$   | $O(n \log^2 n)$         | $O(n \log^2 n)$                | Amir et al. 2000      |
| $d = 1$      | $O(m \log \log n + occ)$            | $O(n \log n)$           | $O(n \log n)$                  | Buchsbaum et al. 2000 |
| $d = 1$      | $O(m + occ)$                        | $O(n \log n)$           | $O(n \log^2 n)$<br>(в среднем) | Maas Novak 2004       |
| $d = O(1)$   | $O(m + \log^d n \log \log n + occ)$ | $O(n \log^d n)$         | $O(n \log^d n)$                | Cole et al. 2004      |
| $d = O(1)$   | $O(d n^\epsilon \log n)$            | $O(n)$                  | $O(n)$                         | Myers 1994            |
| $d = O(1)$   | $O(n^\epsilon)$                     | $O(n)$                  | $O(n)$                         | Navarro et al. 2000   |

# Имеющиеся результаты (2)

| Число ошибок            | Время запроса                                     | Размер структуры данных        | Время индексирования               | Источник                     |
|-------------------------|---|--------------------------------|------------------------------------|------------------------------|
| $d = O(1)$              | $O(m \log^2 n + m^2 + \text{осс})$<br>(в среднем) | $O(n \log n)$<br>(в среднем)   | $O(n \log^2 n)$<br>(в среднем)     | Chávez et al. 2002           |
| $d = O(1)$              | $O(m \min\{n, m^{d+1}\} + \text{осс})$            | $O(\min\{n, m^{d+1}\} + n)$    | $O(\min\{n, m^{d+1}\} + n)$        | Cobbs 1995<br>(Ukkonen 1993) |
| $d = O(1)$ ,<br>Hamming | $O(\log^{d+1} n)$<br>(в среднем)                  | $O(n)$                         | $O(n)$                             | Мааß 2004                    |
| $d = O(1)$              | $O(m + \text{осс})$                               | $O(n \log^d n)$<br>(в среднем) | $O(n \log^{d+1} n)$<br>(в среднем) | Мааß Novak 2005              |
| $d = O(1)$              | $O(m + \text{осс})$<br>(в среднем)                | $O(n \log^d n)$                | $O(n \log^{d+1} n)$                | Мааß Novak 2005              |

## Часть II – Необходимые знания

---

- Расстояние Левенштейна
  - Функция `minpref`
  - Бор
  - Сжатый бор
  - ▣ /-слабый бор
  - Интервальные запросы
-

# Расстояние Левенштейна

---

- $d(u, v)$  = наименьшее количество операций редактирования, необходимое, чтобы перевести  $u$  в  $v$ .
- Вычисляется методом динамического программирования:

$$d(u[1..i], v[1..j]) = \min \left( \begin{array}{l} d(u[1..i], v[1..j-1]) + 1, \\ d(u[1..i-1], v[1..j]) + 1, \\ d(u[1..i-1], v[1..j-1]) + \delta_{u[i], v[j]} \end{array} \right)$$

---

# Расстояние Левенштейна (2)

---

□ Пример:

$$d(\text{"АВТОР"}, \text{"АФФТАР"}) = 3$$

■ АВТОР

■ АФТОР

■ АФФТОР

■ АФФТАР

□ За время  $O((|u| + |v|) * k)$  можно найти  $\min(d(u, v), k)$  [Укконен 1985]

---

# Функция $\text{minpref}_u(v)$

---

- $\text{minpref}_u(v) = \min l:$ 
    - $d(\text{pref}_l(u), \text{pref}_{l+|u|-|v|}(v)) = d(u, v)$
    - $\text{suff}_{l+1}(u) = \text{suff}_{l+|u|-|v|+1}(v)$
  - Пример:
    - $\text{minpref}_{\text{АВТОР}}(\text{АФФТАР}) = 4$ 
      - АВТО●Р
      - АФФТА●Р
      - $d(\text{АВТО}, \text{АФФТА}) = 3$
-



# Лемма о minpref

---

- Пусть  $d(u, v) = k$
  - Обозначим за  $u(i)$  строку  $u$  после  $i$  из  $k$  операций редактирования
  - Если  $\text{minpref}_{u(i)}(u) > h + 1$ , то для некоторого  $j > h$   
 $\text{pref}_j(v) = \text{pref}_j(u(i-1))$
  - Например:
    - АВТОР
    - АФТОР
    - АФФ●ТОР
    - АФФТАР

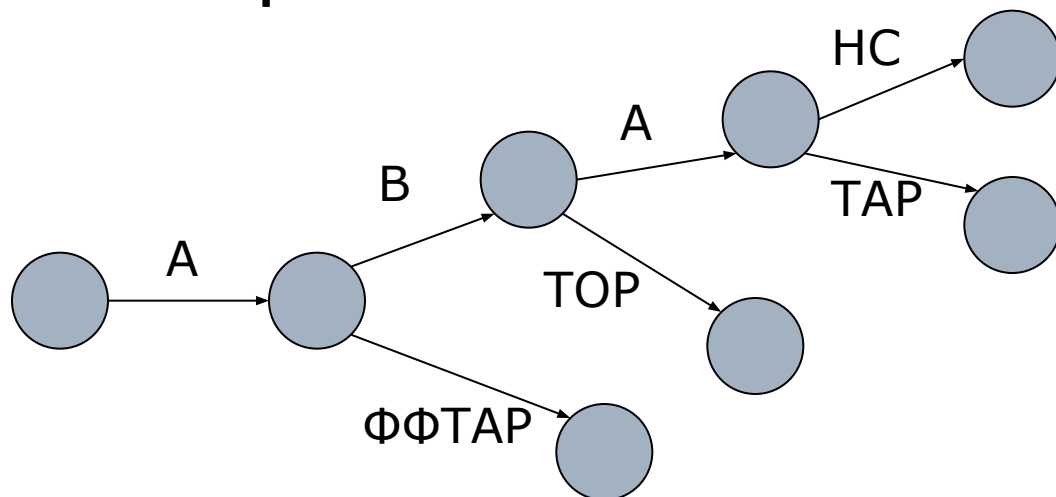
$i = 2$   
 $\text{minpref}_{u(2)}(u) = 3$   
 $h = 1$   
 $j = 2$   
 $\text{pref}_2(v) = \text{pref}_2(u(1))$
-



# Сжатый бор

---

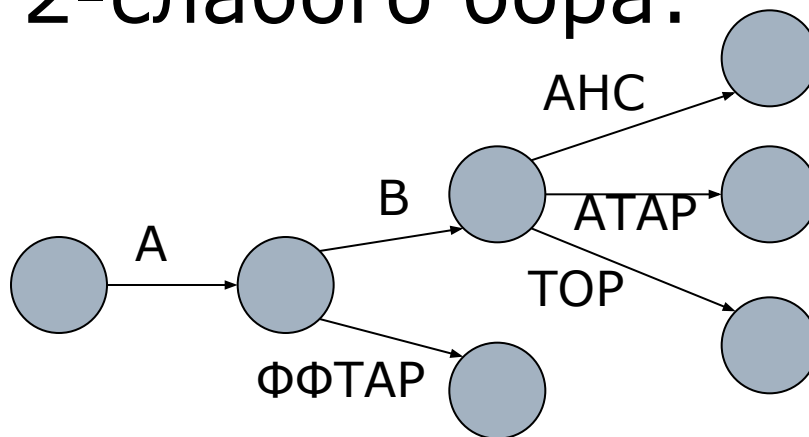
- Структура данных для хранения набора слов



# /-слабый бор

---

- Вершины глубины менее / имеют структуру сжатого бора
- После /-го уровня – никакого ветвления
- Пример 2-слабого бора:



# Интервальные запросы

---

- Дан массив  $A$  длины  $n$  с целыми числами.
  - Поступают запросы про числа в позициях с  $i$  по  $j$ .
  - Время работы  $\langle f(n), g(n) \rangle$  обозначает
    - Один запрос обрабатывается за время  $f(n)$
    - Предобработка занимает время  $g(n)$
-

# Интервальные запросы (2)

---

- RMQ – Range Minimum Query
  - Запрос  $(i, j)$  – найти индекс  $l$ , такой что  $A[l] = \min\{A[k], i \leq k \leq j\}$
  - Алгоритм Фарака-Колтона и Бендера позволяет решать RMQ за наилучшее возможное время:  
 $\langle O(1), O(n) \rangle$
-

# Интервальные запросы (3)

---

- BVRQ – Bounded Value Range Query
    - Запрос  $(i, j, k)$  – найти множество всех индексов  $l$ , таких что  $i \leq l \leq j$  и  $A[l] \leq k$
  - CRQ – Colored Range Query
    - Запрос  $(i, j)$  – найти множество всех различных значений  $A[l]$  при  $i \leq l \leq j$
-

# Часть III – Алгоритм Маасса-Новака

---

- Подход Маасса-Новака
  - Случай  $d = 1$
  - Общий случай
  - Оценка времени поиска
  - Оценка времени индексирования
  - Использование интервальных запросов
-



# Подход Маасса-Новака

---

- Старый подход №1:
    - Выберем строку  $s$  из  $T$ . За время  $O(|P|d)$  можно сравнить ее с  $P$ .
  - Старый подход №2:
    - Построим словарь всех слов, отличающихся от слов из  $T$  не более чем на  $d$ . Тогда поиск  $P$  будет занимать  $O(|P|)$ .
-

# Подход Маасса-Новака

---

- Чем плохи старые подходы?
  - Старый подход №1:
    - Перебор всех строк из  $T$  - **ВРЕМЯ**
  - Старый подход №2:
    - Индекс всех слов со всеми возможными ошибками – **ПАМЯТЬ**
-

# Подход Маасса-Новака

---

- Иногда: обнаружим один подходящий вариант и проверим его за  $O(|P|)$ .
  - Иногда: будем искать  $P$  в предпосчитанном дереве строк, мало отличающихся от строк из  $T$ .
-

# Случай $d = 1$

---

Пусть  $S$  – сжатый бор, содержащий все подстроки  $T$ ,  $h_0$  – высота  $S$ .

- Если  $P$  встречается в  $T$  как подстрока (без ошибок), то  $P$  найдется в  $S$ .
  - Если  $P$  встречается в  $T$  с одной ошибкой, возможны два важных случая:
    - Ошибка после  $h_0$ -го символа, т.е.  $\text{minpref}_s(P) > h_0$
    - Ошибка до  $h_0$ -го символа (включительно), т.е.  $\text{minpref}_s(P) \leq h_0$где  $s$  – подстрока  $T$ , похожая на  $P$ .
-

# Случай $d = 1$

---

- $\text{minpref}_S(P) > h_0$ 
    - Ищем  $P$  в боре  $S$
    - Доходим до листа
    - Сверяем метку на этом листе с оставшимся суффиксом  $P$  ("старый подход №1")
-

# Случай $d = 1$

---

- $\text{minpref}_s(P) \leq h_0$ 
    - Предподсчитаем все строки, отличающиеся от строк из  $T$  ровно одной ошибкой, и эта ошибка в позиции, не большей  $h_0$ .
    - В каждой позиции бывает  $2|\Sigma|$  разных ошибок
    - Для каждой строки из  $S$  порождается  $O(h_0)$  новых строк
    - Эти строки положим в сжатый бор  $S'$  высоты  $h_1$
  - В боре  $S'$  найдем все вхождения строки  $P$ 
    - Их может быть несколько
    - Здесь пригодятся интервальные запросы
-

# Общий случай

---

- Пусть  $P$  совпадает некоторой строкой  $s$  из  $S$  с  $d$  ошибками
  - Если  $\text{pref}_{h_0}(P) = \text{pref}_{h_0}(s)$ , дойдем в  $S$  до листа, далее "старый подход №1", на этот раз  $O(|P|d)$  времени.
  - Иначе: в боре  $S'$  есть строка  $r$ , являющаяся строкой  $P$  с исправленной первой ошибкой.
-

# Общий случай (2)

---

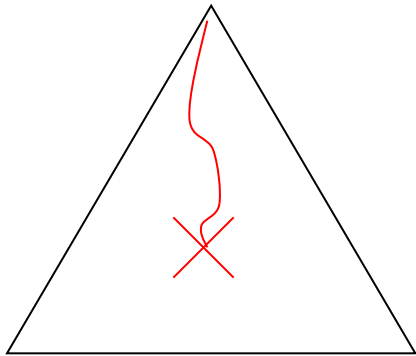
- Снова два случая:
  - $\text{minpref}_s(r) > h_1$ 
    - Дойдем в боре  $S'$  до соответствующего листа, далее "старый подход №1"
  - $\text{minpref}_s(r) \leq h_1$ 
    - Предподсчитаем все строки, отличающиеся от строк из  $S'$  одной ошибкой в первых  $h_1$  символах.
    - При этом бор разрастется в  $O(h_1)$  раз.
    - В боре  $S''$  находим строчку  $P$  и так далее.
-



# Оценка времени поиска

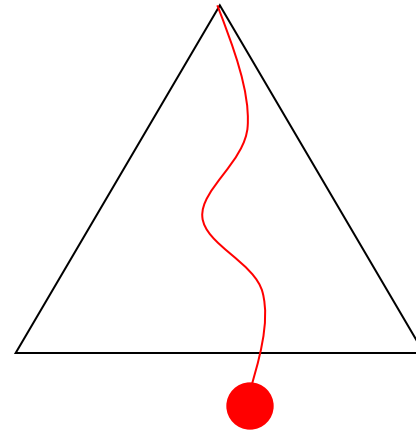
---

- В боре не оказалось строки  $P$



- $O(m)$

- Пройдя бор, мы дошли до листа



- $O(m + dm)$
-

# Оценка времени поиска (2)

---

- При обходе бора кончилась строка  $P$



- Итого:  
 $O(m + \text{осс})$

*$d$  и  $|\Sigma|$  считаются константами*

- $O(m + \text{осс})$
-

# Оценка времени индексирования

---

- Суммарный размер вспомогательных боров:  
 $O(h_0 h_1 \dots h_{d-1} |S|)$
  - Время построения индекса:  
 $O(h_0 h_1 \dots h_d |S|)$
  - $h_i = O(\log n)$ 
    - В среднем
    - С высокой вероятностью
    - *Доказано Маассом и Новаком в модели постоянного эргодического источника*
-

# Использование интервальных запросов

---

- Необходимо за время  $O(осс)$  находить
    - все первые позиции вхождений
    - все документы, содержащие образец
  - Обойдем все листья боров в лексикографическом порядке
  - Для каждого вхождения в массив  $A$  запишем первую позицию вхождения/номер документа
  - Для внутренних узлов бора, поддеревьям соответствуют интервалы в массиве  $A$
  - В массиве  $A$  необходимо за время  $O(осс)$  обрабатывать запросы CRQ
-

# Использование интервальных запросов (2)

---

- CRQ сводится к BVRQ
  - Заведем массив  $B$ :
  - $B[i] =$  предыдущая позиция в массиве  $A$  числа  $A[i]$ , либо  $-1$ , если оно ранее не встречалось
  - CRQ-запрос  $(i, j)$  сводится к BVRQ-запросу  $(i, j, i-1)$  для массива  $B$ .

$A$ 

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 1 | 1 | 2 | 1 | 3 | 3 |
|---|---|---|---|---|---|---|---|---|

$B$ 

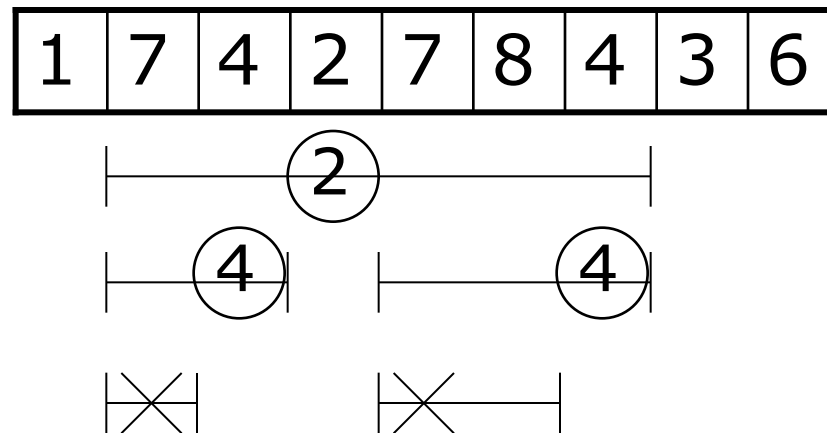
|    |   |    |   |   |   |   |    |   |
|----|---|----|---|---|---|---|----|---|
| -1 | 0 | -1 | 1 | 3 | 2 | 4 | -1 | 7 |
|----|---|----|---|---|---|---|----|---|

---

# Использование интервальных запросов (3)

---

- BVRQ решается за время  $\langle O(\text{осс}), O(n) \rangle$  сведением к RMQ:
- Запрос  $(2, 7, 6)$ :



# Часть IV - Заключение

---

- Алгоритм Маасса-Новака - первый алгоритм, обрабатывающий запрос за  $O(m+oss)$
- Важно улучшить размер и время создания индекса (в данном алгоритме огромные константы)
- Неизвестен алгоритм, с приемлемой оценкой размера индекса в худшем случае

Вопросы ?

---