

2 курс ПМИ, осень 2024/25  
Практика по алгоритмам #8  
Бор, Ахо-Корасик, суфдерево  
12 ноября

Собрано 11 ноября 2024 г. в 15:29

---

## Содержание

1. Бор, Ахо-Корасик, суфдерево	1
2. Разбор задач практики	3
3. Домашнее задание	6
3.1. Обязательная часть . . . . .	6
3.2. Дополнительная часть . . . . .	7

# Бор, Ахо-Корасик, суфдерево

## 1. Сравнение множеств (задача из дз)

Придумайте хеш-функцию, которая позволяет за  $\mathcal{O}(1)$  делать 3 операции с множествами целых положительных 32-битных чисел.

- Добавить элемент в множество
- Удалить элемент из множества
- Проверить два множества на равенство.
- (\*) += ко всем элементам множества.

## 2. ST $\rightarrow$ SA

По суффиксному дереву постройте суффиксный массив с LCP.

## 3. SA $\rightarrow$ ST

По суффиксному массиву с LCP постройте суффиксное дерево.

## 4. (\*) Хитрый поиск

Дан словарь, постройте структуру, чтобы быстро отвечать на запросы

- $\text{get}(s)$  – число строк в словаре, которые начинаются с  $s$ , заканчиваются на  $\overleftarrow{s}$ .
- $\text{get}(s, t)$  – число строк в словаре, которые начинаются с  $s$ , заканчиваются на  $t$ ,  $|s| = |t|$ .
- $\text{get}(s, t)$  – число строк в словаре, которые начинаются с  $s$ , заканчиваются на  $t$ .

## 5. Словари offline

Даны словарь (конечное множество слов) и текст.

- Для каждого слова из словаря определить, входит ли оно как подстрока в текст.
- Для каждого слова из словаря найти число вхождений в текст.

## 6. Словари online

Даны словарь (конечное множество слов) и текст. Обновлять ответ online при добавлении символа в конец текста.

- Пересчитать суммарное число вхождений слов из словаря в текст за  $\mathcal{O}(1)$ .
- Пересчитать множество всех вхождений слов из словаря в текст за  $\mathcal{O}(1 + |\Delta A|)$ , где  $\Delta A$  – приращение ответа после добавления очередного символа.

## 7. Разбиение на словарные слова

Дан словарь слов суммарной длины  $L$  и текст  $T$ .

- Слова длины  $\leq l$ . Представить  $T$  в виде конкатенации минимального числа словарных слов за  $\mathcal{O}(L + l|T|)$ . Слова можно использовать более одного раза.
- (\*) За  $\mathcal{O}(L + \sqrt{L}|T|)$ .
- Представить  $T$  в виде конкатенации минимального количества **подстрок** словарных слов за  $\mathcal{O}(\text{poly}(L) + |T|)$ .

## 8. Задачи про суффиксное дерево

- Найти количество различных подстрок.
- Найти самый длинный рефрен – подстроку  $s$ :  $\text{count}(s) \cdot |s| \rightarrow \max$ .

- c) Самая длинная подстрока, входящая в  $s$  дважды, причём вхождения не пересекаются.
- d) Общая подстрока двух строк за  $\mathcal{O}(|s| + |t|)$ .
- e) Общая подстрока  $k \leq 64$  строк за суммарную длину всех строк.
- f) (\*) Общая подстрока  $\forall k$  строк за суммарную длину всех строк.

### 9. Амортизация в Укконене

Приведите пример, когда Укконен

- a) при добавлении одного символа потратит  $\Omega(n)$  времени,
- b) (\*) при вычислении одной суффикссылки спустится вниз  $\Omega(\sqrt{n})$  раз.

### 10. Бажный Укконен

При подсчёте суффиксных ссылок в алгоритме Укконена маленький Петя делает спуск не по рёбрам (пройти всё ребро за шаг), а по символам (пройти один символ за шаг). Приведите пример строки, на которой полученный алгоритм будет работать  $\omega(n)$ .

### 11. Один бор хорошо, а два – лучше!

Даны два бора  $A$  и  $B$ . Найдите для каждой вершины  $u \in A$  самую глубокую вершину  $B$ , путь до которой равен суффиксу  $\text{path}: \text{root}_A \rightsquigarrow u$ .  $\mathcal{O}(|A| + |B|)$ . Размер алфавита  $\mathcal{O}(1)$ .

*Как увидеть в этой задаче обобщение Ахо-Корасик? Что есть  $A$  и  $B$  в Ахо-Корасик?*

### 12. Быстрый Ахо-Корасик

Дан бор  $A$  над алфавитом  $S$  произвольного размера, постройте суффиксные ссылки за  $\mathcal{O}(|A| \cdot \text{poly}(\log |S|))$ . На самом деле за такое время можно построить полный автомат.

*Мы уже умеем строить суффиксные ссылки несколькими способами.*

Полный автомат мы строим за  $\mathcal{O}(|A| \cdot |S|)$ . bfs + откаты, как в префикс функции, работают за  $\mathcal{O}(\sum |s_i|)$ , что может быть сильно больше  $|A|$ .

### 13. (\*) $\infty$ , избегаемость шаблонов («вирусы»)

Дан словарь слов суммарной длины  $L$ . За время  $\mathcal{O}(L)$  определите, существует ли бесконечная строка, не содержащая ни одно словарное слово как подстроку.

### 14. (\*) XOR $\rightarrow$ max

Дан массив  $a$  длины  $n$ . Найдите пару  $a_i, a_j: a_i \hat{=} a_j = \max$ .

- a)  $\mathcal{O}(n \log M)$ .  $M = \max(a_1, \dots, a_n)$ .
- b)  $\mathcal{O}(\text{sort} + n)$

### 15. (\*) XOR $\geq k$

Дан массив  $a$  длины  $n$  и число  $k$ . Длина чисел в массиве  $\mathcal{O}(1)$ . За время  $\mathcal{O}(n)$  посчитайте количество

- a) пар индексов таких, что побитовый XOR элементов по этим индексам  $\geq k$ .
- b) отрезков последовательности, побитовый XOR всех чисел из которых  $\geq k$ .

### 16. (\*) Окно и стек

Найти максимальный  $\text{reGain}$  (число вхождения  $\times$  длину  $\rightarrow$  max) с помощью суффиксного массива.

# Разбор задач практики

## 1. Сравнение множеств (задача из дз)

Мы можем поддерживать для множества  $A$  величину  $\sum_{a \in A} f(a) \bmod m$ .

Осталось придумать  $f$ : легко посчитать и сложно подделать.  $f(a) = P^a \bmod m$ .

$\forall a \ a += \Delta a \Leftrightarrow f(a) *= P^{\Delta a} \bmod m \Rightarrow$  все три операции `add`, `del`, `+=` за  $\mathcal{O}(\log a)$ .

*Вероятность ошибки:*  $P$  случайное и должно быть корнем многочлена  $\deg = \max a \Rightarrow \Pr \leq \frac{\max a}{m}$ .  $\max a \leq 10^9, m \leq 10^{18} \Rightarrow$  ОК. Но для  $A_1 = \{0\}$ ,  $A_2 = \{m-1\}$  хеши совпадут  $\forall P$ .

*Мемное решение.* Пусть  $f(x) = \text{mem}[x]$ , где `mem` — хеш-таблица, куда при первом обращении кладётся псевдорандом.  $\Pr[\text{ошибки}] = \frac{1}{m}$ , время `add`, `del`  $\mathcal{O}(1)$ , но много лишней памяти.

*Третий вариант.*  $A = \prod_{a \in A} (z-a) \bmod m$ , где  $z$  фиксированное случайное.

Множества  $A$  и  $B$  имеют коллизию  $\Leftrightarrow z$  — корень многочлена степени  $\leq \max |A|, |B| \Rightarrow$

$\Pr[\text{ошибки}] \leq \frac{|A|}{m}$ . `add` за  $\mathcal{O}(1)$ , `del` за  $\mathcal{O}(\log m)$  т.к. нужно обращать по модулю.

*Итого.* Первое — единственное решение с `+=`. Хеш-таблица даёт самое быстрое решение, но ест много памяти. Третье решение лучше первого по ошибке и времени `add`.

## 2. ST $\rightarrow$ SA

dfs по дереву, помним позицию самой высокой вершины с момента, как последний раз были в листе, эта высота и есть LCP.

## 3. SA $\rightarrow$ ST

Пусть мы уже построили дерево для первых  $i$  суффиксов SA, стоим в листе. Поднимаемся от листа до уровня  $lcp[i]$ , создаём развилку, новый лист.

## 4. Хитрый поиск

a) `get(s)` — бор из  $LCP(w, \overleftarrow{w})$ . Спускаемся по  $s$ , смотрим число конечных вершин в поддереве.

b) `get(s, t)` — бор из строк  $w_1 w_n w_2 w_{n-1} w_3 w_{n-2} \dots w_n w_1$ . Спускаемся попеременно по символам  $s$  и  $\overleftarrow{t}$ . Смотрим число конечных вершин в поддереве.

c) `get(s, t)` — бор прямых строк, отдельно бор обратных строк. Конечные вершины помечены номерами строк.

Спускаемся в первом боре по  $s$ , во втором по  $\overleftarrow{t}$ , получили два поддерева. Ответ — пересечение множеств пометок в этих поддеревьях. В каждом дереве все пометки различны, умеем решать такую задачу 2D-запросом.

## 5. Словари offline

Строим Ахо-Корасик для словаря.

a) Проходим по тексту, помечаем все посещенные вершины. В конце проходим снизу вверх по бору и делаем `visited[suf[v]] |= visted[v]`.

b) Проходим по тексту, считаем число посещений каждой вершины. В конце проходим снизу вверх по бору и делаем `count[suf[v]] += count[v]`.

## 6. Словари online

Строим Ахо-Корасик для словаря.

- Посчитаем динамику: количество терминальных вершин на суффиксном пути.
- Посчитаем супер-суффиксные ссылки: ближайшая терминальная вершина на суффиксном пути.

## 7. Разбиение на словарные слова

- Строим бор из словаря.  $f[i]$  – минимальная стоимость выписать префикс длины  $i$ .  
Динамика вперёд: спускаемся по бору суффиксом  $s[i:]$ , если очередная вершина бора конечная, то  $\text{relax}(f[i + \text{dep}], f[i] + 1)$ .  
Максимальная глубина бора  $l \Rightarrow$  время  $\mathcal{O}(L + l|T|)$ .
- Строим суфдерево для  $s_1\#s_2\#\dots\#s_n$  за  $\mathcal{O}(L)$ .  
Жадно разбиваем текст за  $\mathcal{O}(|T|)$ : пока текст не пуст, отрезаем самый длинный префикс текста, по которому можем спуститься в боре.  
*Способ #2:* Ахо-Корасиком. Он находит самую глубокую вершину бора, в которую можно прийти суффиксом текста.  
Считаем динамику, держим очередь с минимумом для окна, покрытого путем до текущей вершины бора.

## 8. Задачи про суффиксное дерево

В суффиксном дереве  $s \forall x$  подстрока  $s$  заканчивается в вершине  $u$  или посреди ребра  $v \rightarrow u$ .  
В поддереве  $u$  заканчиваются все суффиксы начинающиеся в  $x$  (все вхождения  $x$ ).

- Число подстрок.** В боре это число вершин. В сжатом боре это сумма длин ребер.
- Рефрен.**  $\text{count}(s)$  – число суффиксов, кончающихся в поддереве, если спуститься по  $s$ .  
Заметим, что достаточно смотреть на строки, кончающиеся в вершинах.
- Дважды входящая.** Ищем  $v: \min(R[v] - L[v], \text{depth}[v]) \rightarrow \max$ . Ответ может быть посреди ребра.
- Общая подстрока.** Суфдерево для  $s\#t\#$ . Самая глубокая вершина, в поддереве которой есть и суффикс  $s$ , и суффикс  $t$ . Чтобы это определить, храним  $\min$  и  $\max$  суффиксы в поддереве.
- $k \leq 64$  **строк.** Для каждой вершины считаем битмаску номеров строк, суффиксы которых есть в ее поддереве. Ищем самую глубокую, в поддереве которой есть все  $k$ .
- $\forall k$ . За  $\mathcal{O}(n)$  посчитаем для каждой вершины количество различных чисел в поддереве.  
Было в прошлом семестре через LCA и суммирование снизу вверх по дереву.

## 9. Амортизация в Укконене

- $aaa \dots ab$ . Добавление последнего символа создаст  $n$  новых листьев.
- $axbbbbbbbbbbby\ xbc\ xbbc\ xbbbc\ xbbbbc\ xbbbbb\ \dots\ axbbbbbbbbbbbz$

## 10. Бажный Укконен

$aaa \dots a$ . Добавляем  $b$ . Для создания листа суффикса  $i$  нужно спуститься из корня на  $i$ .

11. **Один бор хорошо, а два – лучше!**

Замкнем бор  $B$  до полного автомата за  $\mathcal{O}(|B|)$ . Для каждой вершины  $u \in A$  ответ пересчитываем через предка:  $\text{ans}[a[v][c]] = b[\text{ans}[v]][c]$ .

12. **Быстрый Ахо-Корасик**

Храним в каждой вершине  $v$  персистентный массив  $\text{next}[v]: \text{next}[v] = \text{next}[\text{suf}[v]]$  кроме тех рёбер, которые торчат из вершины  $v$  вниз.  $\text{suf}[v] = \text{next}[\text{suf}[p[v]]][\text{pchar}[v]]$ .

13. (\*)  $\infty$ , **избегаемость шаблонов («вирусы»)**

Строим автомат Ахо-Корасик для словаря. Запретим (выкинем из автомата) вершины, на пути из суффссылок от которых есть запрещённые слова. Проверим, есть ли в графе цикл, достижимые из стартовой вершины. Бесконечная строка  $\exists$  iff  $\exists$  такой цикл.

14. (\*) **XOR**  $\rightarrow$  **max**

a) Числа – битовые строки длины **ровно**  $\log M$ .

Строим на них бор, старшие биты ближе к корню.

Переберём  $a_i$ . Будем спускаться по бору, стараясь получить  $a_j$  с максимальным  $a_i \hat{=} a_j$ : если есть ребро по биту, не равному соответствующему биту в  $a_i$ , спускаемся по нему.

b)  $\mathcal{O}(\text{sort} + n)$ . Сначала заметим, что можно не перебирать  $a_i$ , а найти оба элемента параллельным спуском по бору: идем по разным битам, если можем.

Далее, размер сжатого бора  $\mathcal{O}(n)$ . Имея сжатый бор, можно за  $\mathcal{O}(n)$  найти ответ.

Сортируем массив и считаем LCP соседних. LCP можно посчитать битовыми операциями и предсчитанными логарифмами степеней двоек.

По сортированному массиву и LCP умеем построить сжатый бор за  $\mathcal{O}(n)$ .

15. (\*) **XOR**  $\geq k$ 

a) Строим бор на числах массива как битовых строках равной длины.

Перебираем  $a_i$ , считаем число пар с ним. Для этого спускаемся по бору туда, где XOR даст результат, как в  $k$ . Если спустились по 0, прибавляем к ответу число листьев поддеревы, куда ведёт ребро по 1.

b) XOR подотрезка  $[i, j]$  – это  $\text{pref}_i \hat{=} \text{pref}_{j+1}$ , где  $\text{pref}_i$  – XOR на префиксе. Решаем предыдущий пункт для  $\text{pref}$ .

16. (\*) **Окно и стек**

Взяли LCP соседних, получили задачу: найти отрезок такой, что  $\text{длина} \times \min \rightarrow \max$ . Умеем её решать стеком за  $\mathcal{O}(n)$  из первого семестра.

# Домашнее задание

## 3.1. Обязательная часть

### 1. (2) Первое и последнее вхождение

Даны словарь и текст. Найдите для каждого словарного слова первое и последнее его вхождения в текст в качестве подстроки.

### 2. (3) Навигация в боре

Даны бор  $A$  и строка  $s$ . Найдите вершину бора  $v$ , от которой строку  $s$  можно отложить вниз по бору. Размер алфавита  $\mathcal{O}(1)$ . Время  $\mathcal{O}(|A| + |s|)$ .

### 3. (3) Количество строк

Дан словарь и число  $n$ . Посчитайте количество строк длины  $n$  над алфавитом  $\{a, b\}$ , которые не содержат ни одного словарного слова, как подстроку. Время  $\mathcal{O}(nL)$ , где  $L$  – суммарная длина слов в словаре.

(+1) допбалл за  $\mathcal{O}(L)$  памяти.

### 4. (2) LowerBound

Напишите код на C++ структуры данных с интерфейсом

a) `void add(const vector<int> &s);` – добавить строку.

b) `vector<int> lowerbound(const vector<int> &s);` – найти строку лексикографически минимальную среди больше либо равных  $s$ .

Размер алфавита  $\mathcal{O}(1)$ . За основу возьмите следующий бор:

```
1 struct Vertex {
2     static const int ALPHABET = 26;
3     vector<int> next;
4     bool isEnd;
5     Vertex() : next(ALPHABET, -1), isEnd(0) { }
6 }
7 vector<Vertex> trie(1); // бор из одной вершины
8 int root = 0;
```

### 5. (3) Ключевые подстроки

Дан набор строк  $s_i$ . Для каждой  $s_i$  найдите минимальную по длине подстроку, которая не встречается в других  $s_j$ .

### 6. (3) Уникальные суффиксы

Два типа запросов в онлайн, в сумме должно отработать за линию:

a) `addLetter(c)` – дописать в конец строки символ  $c$ .

b) `isUnique(len)` – является ли суффикс длины  $len$  уникальной подстрокой.

### 7. (3) $k$ -я общая подстрока

Найти  $k$ -ю лексикографически общую подстроку  $s$  и  $t$  за  $\mathcal{O}(|s| + |t|)$ . Размер алфавита  $\mathcal{O}(1)$ .

(+1) за произвольный размер алфавита.

## 3.2. Дополнительная часть

### 1. (1) XOR-3

Дана массив натуральных чисел  $a$  длины  $n \leq 10^6$ . Найдите отрезок  $a$ , побитовый XOR всех чисел из которого максимален. Полный балл получит решение за время  $\mathcal{O}(n + \text{sort})$ .

На дополнительный балл можно придумать чистое  $\mathcal{O}(n)$ .

### 2. (3) Поиск по отрезку словаря

Дан словарь  $s_1, s_2, \dots, s_n$ . Отвечать в offline на запросы  $\text{get}(t, l, r)$  – сколько строк из  $\{s_l, \dots, s_r\}$  входят в  $t$  как подстроки. Время работы – линия от размера входа на полилог.

### 3. (3) Динамический словарь

В словаре теперь добавляются «add(s)» и удаляются «del(s)» слова.

Необходимо в online научиться отвечать на запрос  $\text{get}(t)$  вида «входит ли в текст  $t$  хоть одно словарное слово». Время работы add(s) и del(s):  $\mathcal{O}(|s| \log L)$ , время работы  $\text{get}(t)$ :  $\mathcal{O}(|t| \log L)$  ( $L$  – суммарная длина всего).

### 4. (3) Три вхождения

Дана строка  $s$ . Найти число строк  $t$  таких, что они имеют хотя бы 3 непересекающихся вхождения в строку  $s$ .