Второй курс, осенний семестр 2021/22
Практика по алгоритмам #8

Суффиксный массив, бор

7 ноября

Собрано 24 ноября 2022 г. в 11:28

Содержание

Суффиксный массив, бор

1. **Задачи про суффиксный массив**
   a) Наибольшая общая подстрока двух строк за $\mathcal{O}(|s| + |t|)$.
   b) Реализуйте сжатие `LZSS` за $\mathcal{O}(n \log n)$. А за $\mathcal{O}(n)$?

2. **Бор**

   Рассмотрим строки $S = \{s_1, s_2, \ldots, s_n\}$ над алфавитом Σ.
   Бором для S называется min корневое дерево, из каждой вершины которого по каждому символу Σ исходит не более одного ребра, и $\forall i$ s_i является корректным путём от корня.

   a) Как хранить рёбра бора? Минимизируйте время/память, максимизируйте удобство.
   b) Сделайте на основе бора `unordered_set<string>`.
   c) Сделайте на основе бора `unordered_map<string, int>`.
   d) Сделайте на основе бора `set<string>`.
   e) Что сделать, чтобы все строки заканчивались только в листьях?

3. **Сортировка строк**

   Дан набор из n строк суммарной длины L над алфавитом Σ.

   a) Отсортируйте за время $\mathcal{O}(L \log |\Sigma|)$.
   b) Покажите, что $\nexists$ алгоритма сортировки строк за $\mathcal{O}(L)$.
   c) Отсортируйте за время $\mathcal{O}(L + |\Sigma|)$.
   d) `(*)` Отсортируйте за время $\mathcal{O}(L + n \log L)$.
   e) `(*)` За сколько `QuickSort` сортирует строки?

4. **Хитрый поиск**

   Дан словарь, постройте структуру, чтобы быстро отвечать на запросы

   a) `get(s)` – число строк в словаре, которые начинаются с s.
   b) `get(s)` – число строк в словаре, которые начинаются с s, заканчиваются на $\overleftarrow{s}$.
   c) `get(s,t)` – число строк в словаре, которые начинаются с s, заканчиваются на t, $|s| = |t|$.
   d) `get(s,t)` – число строк в словаре, которые начинаются с s, заканчиваются на t.

5. **Словари offline**

   Даны словарь (конечное множество слов) и длинная строка t.

   a) Для каждого слова из словаря определить, входит ли оно как подстрока в t.
   b) Для каждого слова из словаря найти число вхождений в t.

   $\mathcal{O}(|t| \cdot \max |word_i|)$.

6. **Словари online**

   Нужно обрабатывать следующие запросы

   a) Добавить слово в словарь.
   b) Сказать, сколько слов в словаре не являются ничьим префиксом.
   c) `(*)` Удалить слово из словаря.

7. **Разбиение на словарные слова**

   Дан словарь слов суммарной длины L и текст T. Слова длины $\leqslant l$.
   Представить T в виде конкатенации минимального числа словарных слов за $\mathcal{O}(L + l|T|)$.
   Слова можно использовать более одного раза.

8. **XOR** $\to \max$

   Дан массив a длины n. Найдите пару a_i, a_j: $a_i \hat{} a_j = \max$.
   $\mathcal{O}(n \log M)$. $M = \max(a_1, \ldots, a_n)$.

9. **Поиск подматрицы**

   Даны матрицы чисел A и B. Проверить, является ли B подпрямоугольником A, $\mathcal{O}(|A| + |B|)$.

10. **Почти совпадения**

    Дан словарь. В онлайне поступают слова, нужно говорить «можно ли в данном слове заменить ровно одну букву, чтобы получить словарное слово».

11. **Палиндромов мало**

    Докажите, что различных подпалиндромов не более n. *Указание:* для каждой позиции i существует не более одного нового палиндрома, заканчивающегося ровно в i.

12. **Манакер**

    Помните Z-функцию? А умеете искать максимальный подпалиндром за $\mathcal{O}(n \log n)$ хешами? Соедините эти знания и найдите для каждого i радиус максимального нечётного палиндрома с центром в i за $\mathcal{O}(n)$. А как искать максимальные радиусы чётных палиндромов?

13. (*) **XOR** $\geqslant k$

    Дан массив a длины n 32-битных целых чисел и число k. За $\mathcal{O}(n)$ посчитайте количество

    a) пар индексов таких, что побитовый `XOR` элементов по этим индексам $\geqslant k$.
    b) отрезков массива a, побитовый `XOR` всех чисел из которых $\geqslant k$.

14. (*) **Тандемный повтор – 2**

    Решите задачу про тандемный повтор за $\mathcal{O}(n \log n)$ методом разделяй и властвуй.

15. (*) **РОИ-2004. Задача-ровесница.**

    Даны n словарных слов и m слов текста. Суммарная длина всех $n+m$ слов равна L. Скажем, что слова похожи, если можно из каждого удалить не более одной буквы, чтобы они стали равны. Найдите для каждого слова текста:

    a) какое-нибудь похожее слово словаря;
    b) кол-во похожих слов в словаре.

Разбор задач практики

1. **Задачи про суффиксный массив**
   a) **Общая подстрока.** Строим суфмас $s\#t\#$ и считаем LCP. Для каждого суффикса s ищем ближайший слева и справа суффикс t, два указателя. Смотрим их LCP – минимум в очереди или Фарах-Колтон-Бендер.
   А можно смотреть только позиции, где рядом суффиксы из s и из t, все равно их LCP не меньше, чем у не соседних.
   b) **LZSS.** Пусть мы уже выписали i символов. Нужно быстро найти $j < i$: $\texttt{LCP}(i, j) = \max$. И жадно из i перейти в $i + \texttt{LCP}(i, j)$. Раньше мы перебирали все j за $\mathcal{O}(i)$. Теперь мы можем посмотреть на ближайший слева/справа в суффмассиве.
   Способ за $\mathcal{O}(n \log n)$. Будем держать позиции в суфмассиве всех $j < i$ в `set<int>` (нужны `insert`, `lower_bound`).
   Способ за $\mathcal{O}(n)$. Каждой позиции суфмассива соответствует начало суффикса $sa[i]$. Нужно найти ближайший справа и слева меньший элемент массива sa.
   Умеем это делать стеком за $\mathcal{O}(n)$ (см. 1-й семестр).
   Минимум `LCP` можно насчитывать, запоминая минимум между соседями на стеке. А можно написать Фараха-Колтона-Бендера.

2. **Бор**
   a) **Хранение.** `v.next[c]` дает ребро из вершины `v` по символу `c`.
   `next` может быть массивом, `map`, `unordered_map`.
   Если массив, то $\mathcal{O}(L|\Sigma|)$ памяти, иначе $\mathcal{O}(L)$, L – суммарная длина строк.
   b) `unordered_set<string>`. При добавлении строки создаем все нужные вершины и ребра, если их еще нет. В вершинах, где кончается строка, ставим пометку, что они конечные.
   c) `unordered_map<string, int>`. Дополнительное поле в конечных вершинах.
   d) `set<string>`. `v.next` – массив или `map`. Тогда можно перебирать строки в сортированном порядке и делать `lower_bound`.
   e) Чтобы строки заканчивались только в листьях, добавим в конец каждой символ, не встречающийся в строках.

3. **Сортировка строк**
   a) $\mathcal{O}(L \log |\Sigma|)$. Строим бор, в вершине не массив, а `map`, иначе нужно выделить $\mathcal{O}(L|\Sigma|)$ памяти. `dfs` по бору, из вершины идем в детей в порядке возрастания символа.
   b) Если большой алфавит и строки длины 1, то это не проще сортировки L чисел.
   c) $\mathcal{O}(L+|\Sigma|)$. Отсортируем поразрядно пары $\langle j, s_i[j]\rangle$, т.е. «позиция в строке, символ». Также про каждую пару помним, из какой она строки.
   Теперь можно класть в бор сразу упорядоченные ребра. Берем пару, смотрим, в какую вершину бора пришли по соответствующей строке. Добавляем в нее соответствующее ребро, либо оно там уже есть, тогда это ровно последнее добавленное.
   d) `map` $\to$ `SplayMap` (теперь спуск за $l + \log$ вместо $l \cdot \log$)

4. **Хитрый поиск**
   a) `get(s)` – бор из $\texttt{LCP}(w, \overleftarrow{w})$. 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 бор. От каждой позиции текста «откладываем бор» — спускаемся по бору и отмечаем найденные слова.

6. **Словари online**

При добавлении помечаем все пройденные вершины. Ответ это ровно число непомеченных вершин. Чтобы удалять, вместо `=1`, делаем `+1` и `-1` (храним, сколько раз вершина помечена).

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 `relax(f[i + dep], f[i] + 1)`.
Максимальная глубина бора $l \Rightarrow$ время $\mathcal{O}(L + l|T|)$.

8. **XOR** $\to \max$

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 предпосчитанными логарифмами степеней двоек.
По сортированному массиву и LCP умеем строить сжатый бор за $\mathcal{O}(n)$.

9. **Поиск подматрицы**

Хеш угла (r, c) матрицы $\sum_{ij} a_{ij} P^{r-i} Q^{c-j}$. Хеш подпрямоугольника – знакопеременная сумма.

10. **Почти совпадения**

Сложим в хеш-таблицу все «словарные слова с одним пропуском». Пример `idea` $\to$ `*dea`, `i*ea`, `id*a`, `ide*`. При запросе, ищем «данное слово с одним пропуском в хеш-таблице». Для слова w за $\mathcal{O}(|w|)$ получаются хеши всех его версий с пропусками $\Rightarrow$ решили за $\mathcal{O}$(размера входных данных).

11. **Палиндромов мало**

Пусть в i заканчиваются два. Отразим более маленький. Вот, он уже был.

12. **Манакер**

Ищем $R[i]$ – «радиус» палиндрома с центром i (округленную вверх половину длины).
Пусть правее всех найденных кончается палиндром с центром i. Назовем его «текущий палиндром», он аналогичен текущим границам в вычислении Z-функции.
Ищем $R[j]$, изначально полагаем $R[j] = \min(R[i - (j - i)], i + R[i] - j)$.
Дальше в лоб расширяем $R[j]$. Расширится, только если перевалили за границу текущего палиндрома. Тогда текущим станет палиндром с центром j.
Время линейно потому, что каждое успешное сравнение двигает правую границу текущего палиндрома.
Четные можно считать отдельно аналогично, а можно найти только нечетные в строке $s_1\#s_2\#\ldots\#s_n$.

13. (*) **XOR $\geqslant k$**

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 поддерева по 1.

b) `XOR` подотрезка $[i, j]$ – это pref_i^pref_{j+1}, где pref_i – `XOR` на префиксе. Решаем предыдущий пункт для pref.

14. (*) **Тандемный повтор – 2**

Разобьем строку на две половинки, запустимся рекурсивно от обоих частей.

Пусть большая часть повтора лежит в левой половине строки (обратный случай аналогичен). Тогда ответ разбивается на 4 части $s_1s_2s_3s_4$, где $s_1 = s_3$, $s_2 = s_4$, s_4 начинается с первого символа правой половины.

Переберем i – начало s_2. Находим максимальный общий префикс $l[i :]$ и r, максимальный общий суффикс $l[0 : i)$ и l. Если они перекрываются, тандемный повтор найден. Чтобы их искать, используем $z(r\#l)$ и $z(\bar{l})$.

15. (*) **РОИ-2004. Задача-ровесница.**

Для каждой фиксированной длины l словарного слова строим два бора всех словарных слов длины l — слова в прямом порядке, слова в обратном порядке. Онлайн отвечаем для слова текста — несколько запросов, как в (1)-й задаче практики. Проверить «есть ли точка в прямоугольнике». Кстати, посещавшие доплекции умеют это КД-деревом за $\mathcal{O}(\log n)$.

Домашнее задание

3.1. Обязательная часть

1. **(2) LowerBound**

   Напишите код на C++ структуры данных с интрефейсом

   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:

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

2. **(2) Разбить строку на палиндромы**

   Дана строка s, нужно за $\mathcal{O}(|s|^2)$ представить её в виде конкатенации минимального числа палиндромов.

3. **(2) Дополним строку до палиндрома!**

   Дана строка s, нужно за $\mathcal{O}(|s|)$ найти такую минимальную строку t, что st – палиндром.

4. **(2.5) Общая подстрока k строк**

   Предложите алгоритм поиска max общей подстроки k строк за их суммарную длину.

   Решите задачу суффиксным массивом.

5. **(2) T9**

   Дан набор n строк s_i. Для каждой s_i найдите min по длине префикс, который не является префиксом других строк. Время $\mathcal{O}(\sum |s_i|)$.

   (+1), если у вас получится $\mathcal{O}(n)$ доппамяти.

6. **(2) Суффиксный массив**

   Пусть у вас есть чёрный ящик `SA_with_LCP(s)`, работающий за $\mathcal{O}(|s|)$.
   Например, Каркайнен-Сандерс и Касаи.

   Задача: за $\mathcal{O}(|s|)$ найти самую длинную p, которая встречается в s минимум 3 раза.

3.2. Дополнительная часть

1. **(3) Сравнение множеств**

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

   a) Добавить элемент в множество
   b) Удалить элемент из множества
   c) Проверить два множества на равенство.

2. **(2) XOR-3**

   Дан массив a длины n. За время $\mathcal{O}(n)$ найдите отрезок массива, побитовый `XOR` всех чисел из которого максимален.

3. **(2) Дерево палиндромов**

   Мы уже знаем, что различных подстрок-палиндромов не более n. Дана строка длины n. Посчитайте для каждого i максимальный палиндром-суффикс `s[0:i)`.

4. **(4) Разбить строку на 3 палиндрома**

   Дана строка s, нужно за $\mathcal{O}(|s|)$ представить её в виде конкатенации 3 палиндромов.

   Оцениваться будут только решения за $\mathcal{O}(|s|^{2-\varepsilon})$