Второй курс, осенний семестр 2020/21
Практика по алгоритмам #7

Суффиксный массив, хеши

18 октября

Собрано 18 октября 2021 г. в 22:37

Содержание

Суффиксный массив, хеши

1. **Тандемный повтор – 2**

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

2. **Поиск подматрицы**

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

3. **Палиндромов мало**

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

4. **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?

5. **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 $\mathcal{O}(n)$?

6. **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?

7. **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?

8. **Почти совпадения**

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

9. **(*) Строка по суффмассиву**

   a) Построить такую строку s, что её суффиксный массив совпадает с данным, за $\mathcal{O}(n)$.
   b) Минимизировать размер алфавита.

10. **(*) Regexp**

    Дан паттерн — строка длины n, состоящая из букв и знаков «?». Знаков «?» не более k. Вместо каждого знака «?» можно подставить любую букву. Предложите практически эффективный алгоритм поиска паттерна в тексте.

Разбор задач практики

1. **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 $z(\bar{l})$.

2. **Поиск подматрицы**

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

3. **Палиндромов мало**

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

4. **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 $s_1\#s_2\#\ldots\#s_n$.

5. **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 $\mathcal{O}(n)$.
   Минимум `LCP` можно насчитывать, запоминая минимум между соседями на стеке. А можно написать Фараха-Колтона-Бендера.

6. **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.

7. **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 $l \cdot \log$)

8. **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).

9. **(*) Строка по суффмассиву**
   a) `s[sa[i]] = i`.
   b) Суфмассив – p, $q \circ p = \mathrm{id}$, s – строка.
   Пусть это $p_k = i$ и $p_{k+1} = j$. Когда можно взять $s_i = s_j$? Ровно тогда, когда $q_{i+1} < q_{j+1}$.
   Алгоритм: $s_{p_0} = 0, s_{p_{i+1}} = s_{p_i} + (q_{p_i+1} > q_{p_{i+1}+1})$.

10. **(*) Regexp**

    Это задача на «вместе подумать» и «обсудить». Непонятно, как решать, да. Нужен алгоритм хороший на реальных данных (текстах или ДНК-паттернах), а не «в худшем случае».

Домашнее задание

3.1. Обязательная часть

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

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

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

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

3. **(2.5) Общая подстрока k строк**

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

   Решите задачу суффиксным массивом.

4. **(2.5) Суффдерево можно строить без Укконена!**

   Предложите алгоритм, который по данному вам суффиксному массиву и массиву LCP строит за $\mathcal{O}(n)$ сжатое суффиксное дерево. Размер алфавита $\mathcal{O}(1)$.

5. **(1) Разные подстроки**

   Найти строку над алфавитом $\{0, 1\}$, в которой $\Omega(n^2)$ различных подстрок.

6. **(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.

7. **(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) 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)`.

3. **(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 $\mathcal{O}(|s|^{2-\varepsilon})$