

Второй курс, осенний семестр 2020/21

Практика по алгоритмам #6

Строки (база, хеши)

11 октября

Собрано 16 октября 2021 г. в 13:12

Содержание

| | |
|-------------------------------------|---|
| 1. Строки (база, хеши) | 1 |
| 2. Разбор задач практики | 3 |
| 3. Домашнее задание | 7 |
| 3.1. Обязательная часть | 7 |
| 3.2. Дополнительная часть | 8 |

Строки (база, хеши)

Разрешается пользоваться только префикс-функцией, Z-функцией, хешами.

Суффиксный массив – отсортированный массив суффиксов строки.

Каждый суффикс кодируется позицией начала.

1. gcd – тоже период!

Период строки s – число x : s – префикс $s[0:x)^\infty$.

Период называется *целым*, если $x \mid |s|$.

Пусть у строки s есть периоды $a, b \leq \frac{|s|}{2}$. Докажите, что $\gcd(a, b)$ – тоже период.

2. В поисках периода (и целого, и вещественного!)

a) Найти кратчайший период строки тремя способами: π -функции, Z-функция, хеши.

b) Найти все периоды строки.

3. Подсчёт различных подстрок

a) Найти число различных подстрок строки. $\mathcal{O}(n^2)$. Два способа: Z-функция, хеши.

b) Найти подстроку данной строки, встречающуюся максимальное число раз.

c) Найти подстроку данной строки длины ровно k , встречающуюся максимальное число раз.

d) Возьмём последнее решение. Какова вероятность коллизии?

4. Суффиксный массив и сортировки

Даны строка s . Постройте её суффиксный массив.

a) $\mathcal{O}(n^2 \log n)$.

b) $\mathcal{O}(n \log^2 n)$.

c) Что оптимальнее использовать: `sort` или `stable_sort`?

5. Позиция строки в суффиксном массиве

Найти позицию строки в её суффиксном массиве. Два способа: Z-функция, хеши.

6. k -й суффикс

Найти k -й в лексикографическом порядке суффикс строки.

a) $\mathcal{O}(n \log^2 n)$.

b) $\mathcal{O}(n \log n)$.

7. Поиск с одной ошибкой

Научиться искать образец в строке, если допустимо различие в один символ между образцом и найденной подстрокой.

8. Поиск с перестановками символов и алфавита

Найти образец в строке, если допустимо:

a) в образце применять к алфавиту перестановку.

b) в образце переставлять символы.

c) в образце переставлять и алфавит, и символы.

9. Наибольшая дважды подстрока

Найти наибольшую по длине строку, которая дважды без перекрытий встречается в заданной строке. $\mathcal{O}(n \log n)$.

10. Палиндромы

- a) Найти количество подпалиндромов строки. $\mathcal{O}(n \log n)$.
- b) Найти максимальный подпалиндром строки. $\mathcal{O}(n)$.

11. Восстановление строки по P и Z функциям

За $\mathcal{O}(n)$ восстановить строку, если дана ее

- a) Z-функция.
- b) префикс-функция.

12. (*) Модуль в хешах - 1

Задача: $n \leq 10^6$ раз сравнить на равенство две подстроки s . Хватит ли `int`-ого хеша?

13. (*) Модуль в хешах - 2

Мы считаем число различных подстрок хешами за $\mathcal{O}(n^2)$, $n \leq 1000$. Хватит ли `int`-ого хеша?

14. (*) Префиксы, представимые в k - $\alpha\beta$ -виде

Для каждого префикса строки проверить, представим ли он в виде $\alpha\beta\alpha\beta \dots \alpha$, где α и β – произвольные, возможно пустые, строки, строка β повторяется ровно k раз.

15. (*) Minimal cyclic shift

Найти минимальный циклический сдвиг строки за $\mathcal{O}(n)$.

16. (*) Контрпример к $\langle P, M \rangle$ -хешу.

Даны число P и модуль M .

Постройте две различные строки длины до 10 000, имеющие равные $\langle P, M \rangle$ -хеши.

- a) $1 \leq P < M < 2^{31}$.
- b) $1 \leq P < M < 2^{63}$.

Разбор задач практики

1. gcd – тоже период!

$\text{gcd}(a, b) = xa + yb$, у x и y разные знаки. Раз $a + b < |s|$, то в любой точке можно сделать либо $+= a$, либо $-= b$, значит, можно такими прыжками отступить на gcd вперед или назад.

2. В поисках периода

Есть период длины $t \Leftrightarrow s[t, n] = s[0, n - t] \Leftrightarrow$ суффикс, равный префиксу.

КМП: периоды это $n - p[n], n - p[p[n]], n - p[p[p[n]]], \dots$,
ибо $p[n], p[p[n]], \dots$ – все суффиксы, равные префиксам.

Z: для каждого t проверить, что $z[t] \geq n - t$.

Хеши: для каждого t проверить, что $s[t, n] = s[0, n - t]$.

3. Подсчёт различных подстрок

- a) Хеши: перебираем x , кладем в `unordered_set` хеши всех подстрок длины x , смотрим `size`.
Z: в позиции i начинается $n - i$ подстрок, вычтем встречавшиеся левее, длины $[1..m[i]]$.

```
1 vector<int> m(n, 0);
2 for (int i = 0; i < n; i++):
3     auto z = zFunction(s + i);
4     answer += n - i - m[i];
5     for (int j = i + 1; j < n; j++)
6         m[j] = max(m[j], z[j - i]);
```

В обоих решениях память $\mathcal{O}(n)$, время $\mathcal{O}(n^2)$.

- b) Любой символ самой частой подстроки – тоже самая частая подстрока :) Просто ищем самый частый символ.
c) `unordered_map<uint64_t, int> cnt; cnt[hash(i, i+k)]++;`
d) Если все подстроки различны, вероятность коллизии $\frac{n^2}{MOD}$.

4. Суффиксный массив и сортировки

`char s[n+1]; int p[n] = {0, 1, 2, 3, ...};`

- a) $\mathcal{O}(n^2 \log n) - \text{sort}(p, p + n, \text{cmp1})$. Очень маленькая константа.
b) $\mathcal{O}(n \log^2 n) - \text{sort}(p, p + n, \text{cmp2})$.

```
1 bool cmp1(int i, int j):
2     return strcmp(s + i, s + j) < 0;
3 bool cmp2(int i, int j):
4     int x = LCP(i, j); // бинпоиском с хешами
5     return s[i + x] < s[j + x];
```

`sort` работает в разы медленнее `stable_sort`.

`MergeSort` внутри `stable_sort` делает меньше сравнений, чем `QuickSort` внутри `sort`:
 $n \log_2 n - \Theta(n)$ в худшем против $2n \ln n - \Theta(n)$ в среднем.

На самом деле `sort = IntroSort = QuickSort + HeapSort + InsertionSort`, общее время работы меньше `QuickSort`, но число сравнений больше.

Когда функция сравнения тяжелая, оптимальнее `stable_sort`.

5. Позиция строки в суффиксном массиве

Просто сравнить строку с каждым суффиксом. Хеши за $\mathcal{O}(\log n)$ на каждый.

Z: $z[i] = \text{lcp}(0, i)$, сравнить первые несовпадающие символы: $s[z[i]]$, $s[i+z[i]]$, $\mathcal{O}(1)$ на суффикс.

6. k -й суффикс

У нас есть компаратор за $\mathcal{O}(\log n)$. Его нужно передать функции `sort` ($\mathcal{O}(n \log n)$ вызовов), либо функции `nth_element` ($\mathcal{O}(n)$ вызовов).

7. Поиск с одной ошибкой

Ищем s в t . Переберем начало вхождения i .

Чтобы проверить за $\mathcal{O}(1)$, берем LCP $t[i:]$ и s (Z-функция), после LCP одна ошибка.

Осталось проверить равенство суффиксов (хеши или $z(\overline{s\#t})$).

8. Поиск с перестановками символов и алфавита

a) **Можно переставлять алфавит.** Считаем модифицированную префикс-функцию: самый длинный суффикс данной позиции, равный префиксу с точностью до перестановки алфавита. Такой же код, как у обычной префикс-функции, но внутри модифицированное сравнение на равенство. Заведем массив `prev[i]`, предыдущая позиция символа `s[i]`.

```
1 bool isEqual(int i, int j): // (s[0, i] == s[j - i, j]) <=> (s[i] == s[j])
2   if (prev[i] != -1) return s[j] == s[j - (i - prev[i])];
3   else return prev[j] < j - i;
```

Решение #2, хеши. Движемся по тексту окном длины $|s|$. Для каждого символа алфавита c будем поддерживать h_c «хеш множества позиций, где этот символ встречается». Строка найдена, если множества $\{h_c\}$ для строк s и t совпадают. Чтобы проверять это за $\mathcal{O}(1)$, нужно поддерживать для каждого из множеств хеш $H = \sum_c Q^{h_c} \bmod M$.

b) **В образце s можно переставлять символы.** Идём по тексту, для окна текста длины $|s|$ поддерживаем массив частот `count[char]` и количество совпадений с массивом частот s .

c) **В образце можно переставлять и алфавит, и символы.** Идём по тексту, для окна поддерживаем массив частот частот `count[count[char]]` и количество совпадений с аналогичной конструкцией для образца.

9. Наибольшая дважды подстрока

Бинарный поиск по ответу. Внутри бинпоиска для каждого хеша подстроки длины x в `unordered_map` запоминаем самое левое и самое правое вхождения подстроки.

10. Палиндромы

В обоих пунктах нужно отдельно решить задачу для чётных и нечётных палиндромов.

a) Для каждого центра бинпоиском ищем самый длинный палиндром. Сравниваем хеши исходной и развернутой строки.

b) Перебираем центр, линейным поиском находим самый длинный палиндром, но начинаем поиск с текущего найденного максимума.

```
1 answer = 0;
2 for (int i = 0; i < n; i++)
3   while (substr(i - answer, i) == reversed_substr(i, i + answer))
4     answer++;
```

11. Восстановление строки по P и Z функциям

И префикс-функция, и Z дают нам набор из $\mathcal{O}(n)$ отрезков $[l_i, r_i]: l_i > 0, s[0:r_i-l_i] = s[l_i:r_i]$.

Двигаемся слева направо по строке, если j -й символ не покрыт ни одним отрезком, делаем $s[j] = ss++$, иначе, он покрыт отрезком $[l_i, r_i)$, делаем $s[j] = s[j - 1]$.

12. (*) Модуль в хешах - 1

Да. Вероятность провала одного сравнения равна $\frac{1}{M}$, вероятность провала хотя бы одного из n сравнений не больше $\frac{n}{M} \approx 10^{-3}$ для $M = 10^9 + 7$.

Как это увидеть быстро? n сильно меньше M .

Но нет. Если нам будут пытаться мешать, вероятность одного успеха мы умеем оценивать только как $1 - \frac{len}{M}$, вероятность всех успехов $\leq (1 - \frac{len}{M})^n \leq e^{-\frac{n \cdot len}{M}} \approx 0$.

13. (*) Модуль в хешах - 2

Нет. Пусть мы просто все $\frac{n(n-1)}{2}$ хешей положили в `set`.

Получили $\approx \frac{n^4}{8}$ пар хешей, которые должны различаться.

Смотрим на $(1 - \frac{1}{M})^{n^4/8}$, она почти 0. Как это быстро увидеть? $\frac{n^4}{8}$ сильно больше M .

Но да. А если для длин решаем независимо, то для длины $n - k$ есть $\frac{k(k+1)}{2}$ пар, которые должны различаться. Итого $\sum_k \frac{k(k+1)}{2} \approx \frac{n^3}{6}$ сравнений, вероятность провала $\leq \frac{n^3}{6M} \approx \frac{1}{6}$. Правда, если в задаче, как обычно бывает, 20-50 тестов, не зайдёт.

14. (*) Префиксы, представимые в $\alpha\beta$ -виде

Перебираем $l = |\alpha\beta|$, проверяем $z[l] \geq l(k - 1)$.

Тогда все префиксы на отрезке $[lk, lk + \min(l, z[lk]))$ $\alpha\beta$ -представимы.

Пометить все точки, покрытые отрезками, можно за $\mathcal{O}(n)$.

15. (*) Minimal cyclic shift

За $\mathcal{O}(n \log n)$: `ss = s + s; n = |s|; min_element(ss, ss + n, hashComparator)`.

За $\mathcal{O}(n)$: **алгоритм Дюваля** разложения на простые строки.

16. (*) Контрпример к $\langle P, M \rangle$ -хешу.

Есть решение за \sqrt{M} , следующее из парадокса дня рождений.

В этом случае строки достаточно брать длины $\log M$.

Есть более быстрое решение. Рассмотрим строки s и t , $|s| = |t| = n$ над алфавитом $\{a, b\}$.

Тогда $H_{P,M}(s) - H_{P,M}(t) = \sum c_i P^i \pmod{M}$, где $|c_i| \leq 1$. В другую сторону тоже работает – \forall знакопеременной суммы $\sum c_i P^i$, существуют строки s, t .

Решение:

$A_0 = \text{sorted } \{P^0, P^1, \dots, P^{n-1}\}$.

$A_{i+1} = \text{sorted } \{A_i[1] - A_i[0], A_i[3] - A_i[2], A_i[5] - A_i[4], \dots\}$

Заметим, что по построению $\forall i, j \ A_i[j] \geq 0$ и является знакопеременной суммой P^k .

Если $A_i[0] = 0$, то мы нашли $s \neq t$: $H_{P,M}(s) - H_{P,M}(t) = 0$.

Осталось понять, насколько быстро процесс сойдётся и сойдётся ли вообще.

Для этого сделаем кучу смелых предположений.

1. Элементы A_0 – случайные числа в диапазоне $[0, M)$.

Обозначим за d_i верхнюю границу диапазона, $d_0 = M$.

2. $|A_i| = \frac{n}{2^i} \Rightarrow$ элементы A_{i+1} – случайные в диапазоне $d_i / \frac{n}{2^i}$.

3. Если $n = 2^k$, то получится для A^k получим $|A_k| = 1$.

Мы хотим получить $1 = d_k = M / (\prod_{i=1..k} 2^i) \Rightarrow \log M = \frac{k(k+1)}{2} \Rightarrow k \approx \sqrt{2 \log M}$.

4. Получили $M \approx 2^{64} \Rightarrow k \approx 11 \Rightarrow$ достаточно $|s| \approx 4000$.

Домашнее задание

3.1. Обязательная часть

1. (3) Тандемный повтор – 1

Тандемным повтором называется строка вида $\alpha\alpha$. Найдите за $\mathcal{O}(n^2)$ самый длинный тандемный повтор. Нужно представить три решения, используя

- a) хеши
- b) Z-функция
- c) предподсчитанный массив $\text{lcp}[i, j]$

За каждое решение вы получите по одному баллу.

2. (2) Общий подпалиндром

Нужно за $\mathcal{O}(n \log n)$ найти максимальный общий подпалиндром двух строк.

3. (2) Ретрострока

Для каждого префикса строки найти количество его префиксов равных его суффиксу. $\mathcal{O}(n)$.

4. (3) Z \rightarrow КМП

Преобразовать Z-функцию в префикс-функцию без промежуточного восстановления строки.

5. (1) Поиск с ошибкой в алфавите

Найти подстроку в тексте.

При сравнении строк можно делать циклический сдвиг алфавита в одной из них.

Время решения $\mathcal{O}(n|\Sigma|)$. (+1) допбалл за $\mathcal{O}(n)$.

6. (2) Поиск с двумя ошибками

Найти подстроку в тексте. При сравнении строк, если несовпадений символов было не более двух, строки считаются равными. $\mathcal{O}(n)$.

7. (2) Поиск с k ошибками

Найти подстроку в тексте. При сравнении строк, если несовпадений символов было не более k , строки считаются равными. $\mathcal{O}(nk \log n)$.

3.2. Дополнительная часть

1. (4) Обезьянка за клавиатурой

За одну секунду в конец изначально пустого текста дописывается случайная буква.

Какое матожидание времени T , когда первый раз s станет подстрокой выписанного текста?

Решения за полином от $n = |s|$ получают 1 балл. Чтобы получить 3 балла, нужно $\mathcal{O}(n^2)$.

2. (3) Антихеш тест

Даны целые числа p_1, m_1, p_2, m_2 . Построить две разных строки, у которых полиномиальные хеши (p_1, m_1) и (p_2, m_2) совпадают. Оба.

3. (3) Тандемный повтор – 2

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

4. (3) Покрывание строки

Говорят, что строка α покрывает строку s , если каждый символ s покрыт хотя бы одним вхождением α . Иначе говоря, все вхождения α в s , как отрезки, покрывают всю s . Дана s , найти минимальную по длине α , покрывающую s . $\mathcal{O}(n \log n)$.