SPb HSE, ПАДИИ, 1 курс, осень 2024/25
Практика по алгоритмам #20

Строки

20 февраля

Собрано 20 февраля 2025 г. в 11:25

Содержание

Строки

0. Чёрный ящик «хеши»: умеем предподсчёт за $\mathcal{O}(n)$ и $\forall$ подстроки $[l, r]$ за $\mathcal{O}(1)$ возвращать её хеш. Если хеши строк совпадают, считаем строки равными. Какие «точку» и «модуль» взять для полиномиальных хешей? Оцените вероятность коллизии: $H(s) = H(t)$.
1. Найдите min период строки с помощью хешей за $\mathcal{O}(n)$.
2. Найдите период каждого префикса строки с помощью префикс-функции.
3. Для каждого префикса строки найдите количество его префиксов, равных его суффиксу.
4. Найдите такую подстроку текста t, которая равна данной строке s с точностью до перестановки алфавита. $\mathcal{O}(|s| + |t|)$.
5. Число различных подстрок за $\mathcal{O}(n^2)$. `(*)` $\mathcal{O}(n)$ памяти.
6. Наибольшая общая подстрока двух строк за $\mathcal{O}(n \log n)$.
7. Поиск по словарю. Есть длинная строка t и словарь коротких слов (длина $\leqslant 30$).
   Нужно для каждого слова определить, сколько раз оно встречается в t.
8. Научитесь сравнивать любые две подстроки на больше-меньше за $\mathcal{O}(\log n)$.
9. Построить суффиксный массив за $\mathcal{O}(n \log^2 n)$ времени, $\mathcal{O}(n)$ памяти.
10. С помощью суффиксного массива от текста t научитесь
    в online находить вхождение s в текст за $\tilde{\mathcal{O}}(|s|)$.
11. `(*)` Найдите самый длинный палиндром строки за $\mathcal{O}(n \log n)$. За $\mathcal{O}(n)$.
12. `(*)` Жил был геном – циклическая ACGT-строка длины 10^6.
    Вам дают его 10^6 случайных подстрок длины 100 каждая. Нужно восстановить геном.

Разбор задач практики

0. **Чёрный ящик**

   См. конспект.

1. **Период с помощью хешей.**

   Чтобы проверить, период ли префикс длины t, сравним $s[0{:}n-t)$ и $s[t{:}n)$.

2. **Период с помощью префикс-функции.**

   $n - \pi[n]$.

3. **Количество префиксов, равных суффиксу.**

   $f[i] = f[\pi[i]] + 1$

4. **Перестановка алфавита.**

   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]`.

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

5. **Число различных подстрок за $\mathcal{O}(n^2)$.**

   Добавим все хеши в хеш-таблицу. $\mathcal{O}(n)$ памяти: для каждой длины k считаем ответ отдельно.

6. **Наибольшая общая подстрока двух строк за $\mathcal{O}(n \log n)$.**

   Бинпоиск по ответу. Внутри хеши всех подстрок длины k первой строки кладём в хеш-таблицу, а хеши подстрок длины k второй строки там ищем.

7. **Поиск по словарю.**

   Возьмём все $30 \cdot |text|$ хешей подстрок текста и положим в хеш-таблицу.

8. **Научиться сравнивать любые две подстроки на больше-меньше.**

   Бинпоиск по длину «совпавшей части».

9. **Построить суффиксный массив за $\mathcal{O}(n \log^2 n)$ времени, $\mathcal{O}(n)$ памяти.**

   Суффикс задаётся позицией начала $\Rightarrow$ суфмассив = перестановка чисел.
   Стандартному `sort` передаём компаратор из предыдущей задачи.

10. **Научиться в online находить вхождение s за $\tilde{\mathcal{O}}(|s|)$.**

    Бинпоиск по суффиксному массиву. Внутри за линию сравниваем. $\mathcal{O}(|s| \log |text|)$.

11. **(*) Самый длинный палиндром строки за $\mathcal{O}(n \log n)$.**

    Палиндром задаётся позицией центра. Радиус можно искать бинпоиском. Нужны хеши и прямой, и перевёрнутой строки.

12. **(*) Жил был геном**

    Жадность: соединять две строки с самым длинным зацеплением. Чтобы находить самое длинное зацепление, будем класть хеши подстрок длины k в хеш-таблицу. По убыванию k.

Домашнее задание

1. **(2) Общая подстрока k строк**

   Предложите алгоритм поиска наибольшей общей подстроки k строк длины n.
   Оцените и время работы, и используемую память.

2. **(2) Поиск с перестановкой**

   Предложите алгоритм поиска подстроки «с точностью до перестановки» s в t, при этом строки α и β считаются равными с точностью до перестановки, если $\exists$ перестановка $\pi\colon \pi(\alpha) = \beta$. Произвольный алфавит. $\mathcal{O}(n)$.

3.1. Дополнительная часть

1. **(2) Поиск с k ошибками**

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