Первый курс, весенний семестр

Практика по алгоритмам #9

Строки: префикс-функция, Z, хеши

Contents

1 Новые задачи

1. **gcd – тоже период!**
   Пусть у строки s есть периоды $a, b \le \frac{|s|}{2}$. Докажите, что $\gcd(a, b)$ – тоже период.
2. **В поисках периода.**
   a) Найти кратчайший период строки тремя способами: КМП, 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, встречающую максимальное число раз.
4. **Позиция строки в суффиксном массиве.**
   Найти позицию строки в ее суффиксном массиве. Два способа: Z-функция, хеши.
5. **k-й суффикс.**
   Найти k-й в лексикографическом порядке суффикс строки.
   a) $\mathcal{O}(n \log^2 n)$.
   b) $\mathcal{O}(n \log n)$.
6. **Суффиксный массив и стандартные сортировки.**
   Построение суффиксного массива хешами за $\mathcal{O}(n \log^2 n)$:
   что оптимальнее использовать `sort` или `stable_sort`?
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. **Восстановление строки по P и Z функциям.**
   За $\mathcal{O}(n)$ восстановить строку, если дана ее
   a) Z-функция.
   b) префикс-функция.
10. **Наибольшая дважды подстрока.**
    Найти наибольшую по длине строку, которая дважды без перекрытий встречается в заданной строке. $\mathcal{O}(n \log n)$.
11. **Палиндромы.**
    a) Найти количество подпалиндромов строки. $\mathcal{O}(n \log n)$.
    b) Найти максимальный подпалиндром строки. $\mathcal{O}(n)$.
12. **(*) Префиксы представимые в $\alpha\beta$-виде.**
    Для каждого префикса строки проверить, представим ли он в виде $\alpha\beta\alpha\beta\ldots\alpha$, где α и β – произвольные, возможно пустые, строки, строка β повторяется ровно k раз.
13. **(*) Minimal cyclic shift.**
    Найти минимальный циклический сдвиг строки за $\mathcal{O}(n)$.

2 Разбор задач

1. **gcd – тоже период!**
Пусть $g = \gcd(a, b)$, докажем, что $\forall i\colon s[i] = s[i \bmod g]$. Пусть $a \geq b$. Мы изначально стоим в точке i, далее переходим в равные символы по алгоритму `if (i >= b) i -= b; else i += a;` Заметим, что после `i += a;` $i < a + b \leq |s|$, выхода за пределы строки не произойдёт.

2. **В поисках периода.**
Замечание первое: если существует целый период a, то минимальный период $m \mid a$, поэтому тоже целый. Доказательство: $\gcd(m, a)$ – тоже период, поэтому $m = \gcd(m, a) \mid a$.
Решение КПМ: периоды строки это $n{-}p[n], n{-}p[p[n]], n{-}p[p[p[n]]], \ldots$
Решение Z: перебираем период t, проверяем за $\mathcal{O}(1)$, что $z[t] \geq n{-}t$
Решение хешами: перебираем период t, проверяем за $\mathcal{O}(1)$, что $s[t, n) = s[0, n{-}t)$

3. **Подсчёт различных подстрок.** Строка, встречающаяся максимальное количество раз обязательно длины 1. Поэтому достаточно для каждого символа посчитать, сколько раз он встречается. Количество подстрок хешами считается так: перебираем длину, для данной длины строим хеш-таблицу хешей всех подстрок такой длины. Коллизии не проверяем. $\mathcal{O}(n)$ памяти, $\mathcal{O}(n^2)$ времени. Теперь решаем Z-функцией так же с $\mathcal{O}(n)$ памяти и $\mathcal{O}(n^2)$ времени:
```
for (int i = 0; i < n; i++) {
  auto z = zFunction(s + i);
  answer += n - i - used[i];
  for (int j = i + 1; j < n; j++)
    used[j] = max(used[j], z[j - i]);
}
```

4. **Позиция строки в суффиксном массиве.**
Решение хешами за $\mathcal{O}(n \log n)$: сравниваем строку со всеми её суффиксами на больше/меньше за $\mathcal{O}(\log n)$. Решение z-функцией: у нас есть $z[i] = lcp(0, i)$, сравнить строку с i-м суффиксом $s[z[i]] < s[i{+}z[i]]$.

5. **k-й суффикс.**
У нас есть компаратор, работает он за $\mathcal{O}(\log n)$. Можно его передать функции `sort`, а можно функции `nth_element`.

6. **Суффиксный массив и стандартные сортировки.**
При построении суффиксного массива хешами за $\mathcal{O}(n \log^2 n)$ `sort` работает в разы медленнее `stable_sort`. Причина в числе сравнений. Внутри `stable_sort` живёт `MergeSort`, который делает меньше сравнений, чем `QuickSort` внутри `sort`.

7. **Поиск с одной ошибкой.**
Пусть мы ищем s в t, $|s| = n$. Возьмём строки $s\#t$ и $\overline{s}\#\overline{t}$, посчитаем от них z-функцию (массивы z и $\overline{z}$). Чтобы проверить, что $t[i{:}i{+}n) \approx s$ (“равно” с одной ошибкой), достаточно проверить, что $z[i{+}|s|] + \overline{z}[n - i] \geq n{-}1$.

8. **Поиск с перестановками символов и алфавита.**
   a) К алфавиту можно применять перестановку. Считаем префикс функцию, внутри стандартной процедуры подсчёта префикс функции используем необычное сравнение на равенство. Подробнее:
   ```
   // s[0..i) == t[j-i..j), and the question is s[i] == t[j]?
   bool isEqual(int i, int j) {
     if (s_prev[i] != -1) // s[s_prev[i]] == s[i], s_prev[i] < i
       return t[j] == t[j - (i - s_prev[i])];
     return t_prev[j] == -1 || j - t_prev[j] > i;
     // t[t_prev[j]] == t[j], t_prev[j] < j
   }
   ```
   b) В образце можно переставлять символы. Идём по тексту слева направо, для окна, в котором лежит кандидат на совпадение с образцом, поддерживаем массив частот `count[char]` и количество совпадений с массивом частот образца.
   c) В образце можно переставлять и алфавит, и символы. Идём по тексту слева направо, для окна поддерживаем массив частот частот `count[count[char]]` и количество совпадений с аналогичной конструкцией для образца.
9. **Восстановление строки по P и Z функциям.**
   При восстановлении строки проще всего предаствлять, что нам даны пары отрезков равных символов. Если i принадлежит хотя бы одному отрезку (не префиксу), то мы знаем $s[i] = s[i - shift]$. Иначе значение $s[i]$ нужно выбрать так, чтобы оно случайно не совпало ни с чем нужным. Если ограничений на алфавит нет, то $s[i] =$ `cc++`, иначе в $s[i]$ пишем минимально возможный символ, чтобы нигде префикс-фукнция (Z-функция) не возросла.
10. **Наибольшая дважды подстрока.**
    Бинарный поиск по ответу. Внутри бинпоиска для каждого хеша подстроки длины x в хеш-таблице запоминаем самое левое и самое правое вхождения подстроки.
11. **Палиндромы.** Чтобы найти количество, для каждого центра i бинпоиском найдём $R_1[i]$ и $R_0[i]$ (нечётная длина и чётная длина). Ответ равен $\sum_i (R_0[i] + R_1[i])$.
    Чтобы выбрать максимальный чётный подпалиндром, достаточно следующего:
    ```
    answer = 0;
    for (int i = 0; i < n; i++)
      while (substr(i - answer, i) == substr(i, i + answer))
        answer++;
    ```
12. **(*) Префиксы представимые в $\alpha\beta$-виде.**
    Перебираем $l = |\alpha\beta|$, проверяем, что $z[l] \ge l(k-1)$. Отмечаем, что все префиксы на отрезке $[lk, lk + \min(l, z[lk])]$ $\alpha\beta$-представимы. $\mathcal{O}(n)$.
13. **(*) Minimal cyclic shift.**
    Решение за $\mathcal{O}(n \log n)$: `min_element(s, s + n, hashComparator)`.
    Решение за $\mathcal{O}(n)$: алгоритм Дюваля разложения на простые строки.

3 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 lcp. За каждое решение вы получите по баллу
2. **(3) Общий подпалиндром.**
   Нужно за $\mathcal{O}(n \log n)$ найти максимальный общий подпалиндром.
3. **(3) Ретрострока.**
   Для каждого префикса строки найти количество его префиксов равных его суффиксу. $\mathcal{O}(n)$.
4. **(3*) LZSS.**
   Алгоритм кодирования LZSS. Дана строка s. Выписываем её слева направо. Пусть уже выписан префикс $[0, i)$. Можно или, потратив 1 доллар, записать в код строки s_i и выписать i-й символ, или, потратив 5 долларов, записать в код строки (j, len) и выписать сразу len символов. Здесь $j < i$, а $s[j : j{+}len) = s[i : i{+}len)$. Ваша задача – за $\mathcal{O}(n^2)$ выписать всю строку за минимальную стоимость. Дополнительный балл можно получить, решив задачу с $\mathcal{O}(n)$ памяти.
5. **(3) Z → КМП**
   Преобразовать Z-функция в префикс-функцию без промежуточного восстановления строки.
6. **(3) Поиск с ошибкой в алфавите.**
   Найти подстроку в тексте. При сравнении строк можно делать циклический сдвиг алфавита в одной из них. $\mathcal{O}(n\Sigma)$.
7. **(3) Поиск с двумя ошибками.**
   Найти подстроку в тексте. При сравнении строк, если несовпадений было не более двух, строки считаются равными. $\mathcal{O}(n)$.
8. **(3) Поиск с k ошибками.**
   Найти подстроку в тексте. При сравнении строк, если несовпадений было не более k, строки считаются равными. $\mathcal{O}(nk \log n)$.

3.2 Дополнительная часть

1. **(5) Обезьянка за клавиатурой.**
   За одну секунду в конец изначально пустого текста дописывается случайная буква (равномерное распределение). Какое матожидание времени T, когда первый раз s станет подстрокой выписанного текста?
2. **(6) Антихеш тест.**
   Даны целые числа p и m. Построить две разных строки, у которых (p, m) полиномиальный хеш совпадёт. $h(s_0, s_1, \ldots, s_n) = (\sum s_i p^i) \bmod m$.
   a) **(2)** $\mathcal{O}(m^{1/2})$.
   b) **(2)** $\mathcal{O}(m^{1/3})$.
   c) **(2)** $m \leq 10^{36}$.
3. **(4) Тандемный повтор 2.**
   Решите задачу про тандемный повтор за $\mathcal{O}(n \log n)$ методом разделяй и властвуй!
4. **(4) Покрытие строки.**
   Говорят, что строка α покрывает строку s, если каждый символ s покрыт хотя бы одним вхождением α. Дана s, найти минимальную по длину α. $\mathcal{O}(n \log n)$.