# Первый курс, весенний семестр 2015/16
# Практика по алгоритмам #11

## Prefix-function, Z, hash

## 25 апреля

Собрано 10 мая 2016 г. в 18:16

---

## Содержание


**1. Задачи на тему Prefix-function, Z, hash** 1

**2. Разбор задач практики** 3

**3. Домашнее задание** 6

3.1. Обязательная часть . . . . . 6

3.2. Дополнительная часть . . . . . 7

**4. Разбор домашнего задания** 8

4.1. Обязательная часть . . . . . 8

4.2. Дополнительная часть . . . . . 10

# Задачи на тему Prefix-function, Z, hash

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

1. **gcd – тоже период!**
   Пусть у строки $s$ есть периоды $a, b \leqslant \frac{|s|}{2}$. Докажите, что $\gcd(a, b)$ – тоже период.
2. **В поисках периода (и целого, и вещественного!)**
   a) Найти кратчайший период строки тремя способами: 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. **(*) Модуль в хешах - 1**
    Задача: $n \leqslant 10^6$ раз сравнить на равенство две подстроки $s$. Хватит ли `int`-ого хеша?

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

14. **(*) Префиксы, представимые в $k$-$\alpha\beta$-виде**
    Для каждого префикса строки проверить, представим ли он в виде $\alpha\beta\alpha\beta\ldots\alpha$, где $\alpha$ и $\beta$ – произвольные, возможно пустые, строки, строка $\beta$ повторяется ровно $k$ раз.

15. **(*) Minimal cyclic shift**
    Найти минимальный циклический сдвиг строки за $\mathcal{O}(n)$.

# Разбор задач практики

1. **gcd – тоже период!**
$\gcd(a, b) = xa + yb$, 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 $n - p[n], n - p[p[n]], n - p[p[p[n]]], \ldots$ $p[n], p[p[n]], \ldots$ – все суффиксы равные префиксам.
Z: для каждого $t$ проверить, что $z[t] \geqslant n - t$.
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 $[1..m[i]]$.
```
vector<int> m(n, 0);
for (int i = 0; i < n; i++) {
  auto z = zFunction(s + i);
  answer += n - i - m[i];
  for (int j = i + 1; j < n; j++)
    m[j] = max(m[j], z[j - i]);
}
```
   В обоих решениях память $\mathcal{O}(n)$, время $\mathcal{O}(n^2)$.
   b) Строка, встречающаяся максимальное количество раз, имеет длину 1, как подстрока самой частой. Просто ищем самый частый символ.

4. **Позиция строки в суффиксном массиве**
Просто сравнить строку с каждым суффиксом. Хешами за $\mathcal{O}(\log n)$ на каждый. Z: $z[i] = lcp(0, i)$, сравнить первые несовпадающие символы: $s[z[i]]$, $s[i+z[i]]$, $\mathcal{O}(1)$ на каждый суффикс.

5. **$k$-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).

6. **Суффиксный массив и стандартные сортировки**
`sort` сработает в разы медленнее `stable_sort`. Внутри `stable_sort` живёт `MergeSort`, он 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` + `InsertionSort`, общее время работы меньше `QuickSort`, но число сравнений больше.

7. **Поиск с одной ошибкой**
Ищем $s$ в $t$. Переберем начало вхождения. Проверка за $\mathcal{O}(1)$: берем max совпадающий префикс $t[i:]$ и $s$ (например, с помощью Z), после него одна ошибка, проверяем равенство суффиксов (хеши или $z(\overline{s}\#\overline{t})$).

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]`.

```
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;
}
```

   **Решение #2, хешами**. Движемся окном. Для каждого символа алфавита будем поддерживать "хеш множества позиций, где этот символ встречается". Теперь от множества хешей $A$ нужно насчитать хеш. $H(A) = \sum_{x \in A} Q^x \bmod M$.

   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. **Восстановление строки по P и Z функциям**
   И префикс-функция, и Z дают нам набор из $\mathcal{O}(n)$ отрезков, равных некоторому префиксу. Их можно отсортировать за $\mathcal{O}(n)$ (подсчетом или пользуясь их свойствами) по левому концу. Дальше двигаться по строке и, помня первый отрезок, в котором мы находимся, проставлять нужный символ. Встречая не покрытую позицию, делаем $s[i] =$ `cc++`, если размер алфавита не ограничен.
   Если хотим алфавит поменьше, можно ставить минимальный символ, не портящий префикс или Z. Для этого можно поддерживать `set` символов, из которого выкинуты те, которые идут после отрезков, кончившихся в предыдущей позиции.

10. **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.

11. **Палиндромы**
    В обоих пунктах нужно отдельно решить задачу для чётных и нечётных палиндромов.
    a) Для каждого центра бинпоиском ищем самый длинный палиндром. Сравниваем хешами исходной и развернутой строки.
    b) Перебираем центр, линейным поиском находим самый длинный палиндром, но начинаем поиск с текущего найденного макисмума.

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

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 равна $1 - \frac{len}{M}$, вероятность всех успехов $\leqslant (1 - \frac{len}{M})^n \leqslant e^{\frac{-n \cdot len}{M}} \approx 0$.

13. **(*) Модуль в хешах - 2**
**Нет**. Пусть мы просто все $\frac{n(n-1)}{2}$ хешей положили в `set`, всего $\approx \frac{n^4}{8}$ 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, получаем $\approx \frac{n^3}{6}$ сравнений, вероятность провала не больше $\frac{n^3}{6M} \approx \frac{1}{6}$. Правда, если в задаче, как обычно бывает, 20-50 тестов, не зайдёт.

14. **(*) Префиксы, представимые в $\alpha\beta$-виде**
Перебираем $l = |\alpha\beta|$, проверяем $z[l] \geqslant l(k-1)$. 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 $\mathcal{O}(n \log n)$: `min_element(s, s + n, hashComparator)` (предварительно раздвоим $s$).
За $\mathcal{O}(n)$: алгоритм Дюваля разложения на простые строки.

# 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 `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-функция 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)** доп.балл за $\mathcal{O}(n)$.
6. **(2) Поиск с двумя ошибками**
   Найти подстроку в тексте. При сравнении строк, если несовпадений 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 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 (равномерное распределение). Какое матожидание времени $T$, когда первый раз $s$ станет подстрокой выписанного текста?

2. **(3) Антихеш тест**
   Даны целые числа $p_1$, $m_1$, $p_2$, $m_2$. Построить две разных строки, у которых $(p_1, m_1)$ и $(p_2, m_2)$ полиномиальные хеши совпадают. Оба.

3. **(3) Тандемный повтор - 2**
   Решите задачу про тандемный повтор за $\mathcal{O}(n \log n)$ методом разделяй и властвуй.

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

# Разбор домашнего задания

## 4.1. Обязательная часть

1. **Тандемный повтор - 1**
   a) Хеши: перебираем подстроки чётной длины, сравниваем на равенство хешами первую и вторую половину подстроки.
   b) Z: пеберем позицию $i$ начала тандемного повтора, считаем $z$-функцию для строки $s[i:n)$. Если есть $z[j] \geqslant j$, то нашли тандемный повтор.
   c) LCP: перебираем $|\alpha|$ и позицию середины, проверяем, что LCP двух соответствующих суффиксов $\geqslant |\alpha|$.

   Решение через LCP наиболее короткое, но $\Theta(n^2)$ памяти. Самое быстрое по времени $Z$-решение.

2. **Общий подпалиндром**
   Бинпоиск по длине ответа $x$. Складываем в `unordered_set` хеши подстрок длины $x$ первой строки, являющиеся палиндромами (палинромность проверяем хешами прямой и развернутой строки). Затем перебираем подстроки второй строки и смотрим, есть ли их хеш в `unordered_set`.
   Важно заметить, что для чётных и нечётных длин отдельные бинпоиски (если есть палиндром длины $x$, то есть и длины $x - 2$).

3. **Ретрострока**
   Для префикса длины $i$ нам нужны суффиксы длин $p[i], p[p[i]], \ldots$ Считаем динамикой, сколько раз нужно применить к $i$ префикс-функцию, чтобы сойтись в 0: `f[i] = !i ? 0 : f[p[i]] + 1;`

4. **Z → КМП**
   Если $z[i] > 0$, то для всех $j < z[i]$ верно $p[i+j] \geqslant j+1$.
   Способ #1.

```
for (int i = 0; i < n; i++)
  for (int j = z[i] - 1; j >= 0; j--)
    if p[i + j] > 0
      break;
    else
      p[i + j] = j + 1;
```

   Если $p[i+j]$ было установлено на каком-то предыдущем шаге $i' < i$, то $i' + j' = i + j$ и $j' > j$. Итого каждое значение выставлено только один раз.
   Способ #2. Два указателя: один идет по $i$, другой заполняет ответ на отрезке $[z[i-1], z[i]]$.
   Способ #3. Проходим вперед, пишем `ans[i + z[i] - 1] = z[i]`. Затем идем обратным проходом и делаем `ans[i] = max(ans[i], ans[i + 1] - 1)`.

5. **Поиск с ошибкой в алфавите**
   **Решение за** $\mathcal{O}(n|\Sigma|)$. Можно просто перебрать сдвиг алфавита и любым линейным алгоритмом искать подстроку в строке.

   **Решение за** $\mathcal{O}(n)$. Как в задаче с практики про перестановку алфавита, считаем модифицированную префикс-функцию: максимальный суффикс, равный префиксу с точностью до сдвига алфавита. При сравнении символов в подсчёте префикс-функции, смотрим, что их разность равна разности предыдущих символов.

**Простое решение за $\mathcal{O}(n)$.** Строки заменим на новые, образованные разностями соседних символов. Разность берётся по модулю. Для новых строк нужно найти подстроку в строке в обычном смысле.

6. **Поиск с двумя ошибками**
   Переберем позицию $i$ начала вхождения $s$ в $t$. С помощью $z(s\#t)$ ищем максимальный общий префикс $s$ и $t[i{:}i{+}|s|)$, после него позиция ошибки. С помощью $z(\overline{t}\#\overline{s})$ ищем максимальный общий суффикс $s$ и $t[i{:}i{+}|s|)$, перед ним позиция ошибки. Сравниваем середину хешами.

7. **Поиск с $k$ ошибками**
   Для каждой позиции текста не более $k+1$ раз делаем следующее: ищем за $\mathcal{O}(\log n)$ хешами максимальный общий префикс интересных нам кусков текста и образца, после совпадения следует позиция ошибки, смотрим на следующие куски.
   Кстати, мы уже умеем искать LCP за $\mathcal{O}(1)$, используя суффмассив.

## 4.2. Дополнительная часть

1. **Обезьянка за клавиатурой**
   Пусть обезьяна принимает ставки на следующий символ: поставив $x$ денег и угадав, можно получить $|\Sigma|x$. Матожидание выигрыша обезъяны в такой игре равно нулю, то есть матожидание суммы полученных ставок равно матожиданию суммы выплаченных выигрышей.
   Пусть каждую секунду приходит человек и ставит 1 на $s_1$. В случае выигрыша он на следующем ходу ставит все, что выиграл, на $s_2$, потом все, что выиграл, на $s_3$, и так далее.
   Когда выпадет $s$, что-то выиграют те, у кого был угаданный префикс $s$. Итого в этот момент выплаченный выигрыш равен $|\Sigma|^n + |\Sigma|^{\pi[n]} + |\Sigma|^{\pi[\pi[n]]} + \ldots$ Он равен матожиданию полученных ставок. Раз изначальная ставка каждого равна 1, сумма полученных ставок равна числу раундов игры.

2. **Антихеш тест**
   Решение #1: возьмём алгоритм с лекции. Будем вместо чисел $P^i \bmod M$ возьмём пары чисел $\langle P_1^i \bmod M_1, P_2^i \bmod M_2 \rangle$. На нечётном шаге будем уменьшать разности у $\langle P_1, M_1 \rangle$, на чётном у $\langle P_2, M_2 \rangle$. По тем же причинам, что исходный алгоритм, и те, и другие сойдутся к нулю.

   Решение #2: построим алгоритмом с лекции две строки $s$ и $t$ одинаковой длины, дающие коллизию для $\langle P_1, M_1 \rangle$-хеша. Теперь скажем, что $s$ и $t$ – наш новый алфавит. Любые строки равной длины над этим алфавитом имеют одинаковый $\langle P_1, M_1 \rangle$-хеш. Запустим алгоритм с лекции ещё раз =). Длина теста – квадрат длины теста к $\langle P_1, M_1 \rangle$-хешу.

3. **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 $s_1 s_2 s_3 s_4$, 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 $z(\bar{l})$.

4. **Покрытие строки**
   $f[i]$ – ответ на задачу для префикса строки длины $i$. $\pi[i]$ – значение префикс функции. Заметим, что

$$f[i] = \begin{cases} f[\pi[i]], f[i] \leqslant \pi[i] (*) \\ i, f[i] > \pi[i] (\text{очевидно}) \end{cases}$$

   Почему $(*)$?
   Пусть $f[i] > f[\pi[i]]$, но тогда строку $s[0\!:\!f[i])$ можно покрыть строкой $s[0\!:\!f[\pi[i]])$. ?!?
   Пусть $f[i] < f[\pi[i]]$, но тогда строку $s[0\!:\!f[\pi[i]])$ можно покрыть строкой $s[0\!:\!f[i])$. ?!?
   Как считать динамику $f[i]$?
   Для каждого значения $x$ поддерживать $right[x]$: $f[right[x]] = x, right[x] = \max$. $\mathcal{O}(n)$.