МФТИ. ЗОШ 2025

Винер Даниил

5 января 2025 г.

План на сегодня

Организационная инфа

- Преподаватели

Организационная инфа

- Преподаватели
- Ассистенты

Организационная инфа

- Преподаватели
- Ассистенты
- Дорешки

Организационная инфа

- Преподаватели
- Ассистенты
- Дорешки
- Отсечки

Организационная инфа

- Преподаватели
- Ассистенты
- Дорешки
- Отсечки
- Git с полезной инфой

Организационная инфа

- Преподаватели
- Ассистенты
- Дорешки
- Отсечки
- Git с полезной инфой
- Чатик

Организационная инфа

- Преподаватели
- Ассистенты
- Дорешки
- Отсечки
- Git с полезной инфой
- Чатик
- Языки — Python, C++

Хэш-функция

Определение

Хэш-функция — функция, сопоставляющая объектам какого-то множества числовые значения из ограниченного промежутка

Глобальное применение

- Криптография
- Кэширование
- Хранение паролей

Применение в задачках

- Количество различных подстрок
- Поиск подстроки в строке
- Палиндромность подстроки

Полиномиальное хэширование

Пусть строка — последовательность чисел от 1 до m и $p = 1e9 + 7$, а также $k > m$

Определение

Прямой полиномиальный хэш строки — значение такого многочлена:

$$h_f = (s_0 + s_1 k + s_2 k^2 + \ldots + s_n k^n) \mod p$$

Определение

Обратный полиномиальный хэш:

$$h_b = (s_0 k^n + s_1 k^{n-1} + \ldots + s_n) \mod p$$

Сравнения подстрок

- Для начала, создадим вектор *баз*, который в дальнейшем поможет для полиномиального хэширования строки

Сравнения подстрок

- Для начала, создадим вектор *баз*, который в дальнейшем поможет для полиномиального хэширования строки
- Далее вычисляем массив префиксных хэшей, в котором p_i — хэш строки от начала до s_i

Сравнения подстрок

- Для начала, создадим вектор *баз*, который в дальнейшем поможет для полиномиального хэширования строки
- Далее вычисляем массив префиксных хэшей, в котором p_i — хэш строки от начала до s_i
- После этого функция, вычисляющая хэш подстрок, будет работать так: она принимает два индекса i и j. После этого вычисляем хэш предыдущей части строки, умноженный на соответствующую базу

Что еще можно делать с помощью хэшей

- Количество различных подстрок
- Поиск подстроки в строке
- Палиндромность подстроки

Что еще можно делать с помощью хэшей

Количество различных подстрок

Считаем хэши всех подстрок и кидаем в set, тогда $set.size()$ — ответ

Что еще можно делать с помощью хэшей

Количество различных подстрок

Считаем хэши всех подстрок и кидаем в set, тогда $set.size()$ — ответ

Поиск подстроки в строке

Используем проход *окном* по всей строке

Сложность — $O(n)$

Что еще можно делать с помощью хэшей

Количество различных подстрок

Считаем хэши всех подстрок и кидаем в set, тогда $set.size()$ — ответ

Поиск подстроки в строке

Используем проход *окном* по всей строке

Сложность — $O(n)$

Палиндромность подстроки

Сравниваем значения прямого и обратного хэша подстроки

Z-функция

Определение

Z-функция от строки s — массив z, такой что z_i равно длине максимальной подстроки, начинающейся с i-й позиции, которая равна префиксу s

Z-функция

Определение

Z-функция от строки s — массив z, такой что z_i равно длине максимальной подстроки, начинающейся с i-й позиции, которая равна префиксу s

Пример

$$\text{abracadabra} \longrightarrow z = [11, 0, 0, 1, 0, 1, 0, 4, 0, 0, 1]$$

Построение Z-массива за $O(n)$

- Инициализируем два указателя l и r для границ z-блока, устанавливая их в 0, а $z[0]$ будет равен длине строки.

Построение Z-массива за $O(n)$

- Инициализируем два указателя l и r для границ z-блока, устанавливая их в 0, а $z[0]$ будет равен длине строки.
- Идем по строке слева направо. Для каждого индекса i проверяем, лежит ли он в пределах текущего z-блока:

Построение Z-массива за $O(n)$

- Инициализируем два указателя l и r для границ z-блока, устанавливая их в 0, а $z[0]$ будет равен длине строки.
- Идем по строке слева направо. Для каждого индекса i проверяем, лежит ли он в пределах текущего z-блока:
  - Если $i > r$, то находим z_i наивно: начинаем с s_i и увеличиваем z_i, пока подстрока не перестанет совпадать с префиксом строки. После этого обновляем границы l и r.

Построение Z-массива за $O(n)$

- Инициализируем два указателя l и r для границ z-блока, устанавливая их в 0, а $z[0]$ будет равен длине строки.
- Идем по строке слева направо. Для каждого индекса i проверяем, лежит ли он в пределах текущего z-блока:
  - Если $i > r$, то находим z_i наивно: начинаем с s_i и увеличиваем z_i, пока подстрока не перестанет совпадать с префиксом строки. После этого обновляем границы l и r.
  - Если $i \le r$, то используем значение $z[i - l]$, чтобы инициализировать z_i. Если $z[i - l]$ меньше чем $r - i + 1$, то $z_i = z[i - l]$. Если $z[i - l]$ больше, то уменьшаем его до границы r и увеличиваем z_i на 1.

Построение Z-массива за $O(n)$

- Инициализируем два указателя l и r для границ z-блока, устанавливая их в 0, а $z[0]$ будет равен длине строки.
- Идем по строке слева направо. Для каждого индекса i проверяем, лежит ли он в пределах текущего z-блока:
  - Если $i > r$, то находим z_i наивно: начинаем с s_i и увеличиваем z_i, пока подстрока не перестанет совпадать с префиксом строки. После этого обновляем границы l и r.
  - Если $i \leq r$, то используем значение $z[i-l]$, чтобы инициализировать z_i. Если $z[i-l]$ меньше чем $r-i+1$, то $z_i = z[i-l]$. Если $z[i-l]$ больше, то уменьшаем его до границы r и увеличиваем z_i на 1.
- В конце обновляем границы z-блока, если $i + z_i - 1$ выходит за пределы правой границы z-блока.

Префикс-функция

Определение

Префикс-функция от строки s — массив p, где p_i равно длине самого большого префикса строки $s_0 s_1 s_2 \dots s_i$, который также является и суффиксом i-того префикса (не считая весь i-й префикс)

Префикс-функция

Определение

Префикс-функция от строки s — массив p, где p_i равно длине самого большого префикса строки $s_0 s_1 s_2 \dots s_i$, который также является и суффиксом i-того префикса (не считая весь i-й префикс)

Пример

abracadabra $\longrightarrow \pi = [0, 0, 0, 1, 0, 1, 0, 1, 2, 3, 4]$

Пример

aaaaa $\longrightarrow \pi = [0, 1, 2, 3, 4]$

Построение за $O(n)$

Для каждого символа $s[i]$ от 1 до $N-1$:

- Если $s[i]$ совпадает с $s[k]$ (т.е. $s[i] == s[k]$), то увеличиваем k на 1 и присваиваем $\pi[i] = k$
- Если они не совпадают:
  1. Пока $k > 0$ и $s[i]$ не совпадает с $s[k]$: обновляем k с помощью $k = \pi[k-1]$
  2. После выхода из цикла, если $s[i]$ совпадает с $s[k]$, то снова увеличиваем k и присваиваем $\pi[i] = k$.
  3. Если нет совпадений, оставляем $\pi[i] = 0$

Какие задачи можно решить и так, и так?

- Поиск подстроки в строке
- Количество различных подстрок в строке
- Сжатие строки

Поиск подстроки в строке. Z-функция

Пусть у нас есть строка s и паттерн p

- Добавим к строке s символ $\#$. Получим

$$T = \# + s$$

- Теперь вычисляем для строки T её Z-функцию
- После этого ищем в Z-функции все значения равные длине p
- Если $z[i] = len(p)$, тогда подстрока p начинается в строке T с позиции i, а значит в строке s — с позиции $i - len(p) - 1$

Поиск подстроки в строке. Префикс-функция

Пусть дана строка t и паттерн s. Составим строку $K = s + \# + t$.
Пусть n — длина строки s, а m — строки t

1. Считаем префикс-функцию для строки K
2. Рассмотрим значения префикс-функции, кроме первых $n+1$, так как это строка s и разделитель
   - Если в какой-то позиции i оказалось, что $\pi[i] = n$, то в позиции $i - (n+1) - n + 1 = i - 2n$ строки t начинается очередное вхождение паттерна

Дана строка s длины n. Пусть k — текущее количество различных подстрок строки s, и мы добавляем в конец символ c

- Возьмём строку $t = s + c$ и инвертируем её
- Посчитаем Z-функцию для строки t и найдём максимальное значение $z_{\max}$, тогда в строке t встречается (не в начале) её префикс длины $z_{\max}$, но не большей длины

Тогда, число новых подстрок, появляющихся при дописывании символа c, равно $len - z_{\max}$, где len — длина строки **после приписывания символа** c

Пример

Допустим, у нас есть строка *abacaba* и мы хотим приписать в конец символ k. Тогда, посчитаем Z-функцию для строки *kabacaba*:

$$0, 0, 0, 0, 0, 0, 0, 0$$

Количество различных подстрок в строке. Префикс-функция

- Возьмём строку $t = s + c$ и инвертируем её. Наша задача — посчитать, сколько у строки t таких префиксов, которые не встречаются в ней более нигде
- Если мы посчитаем для строки t префикс-функцию и найдём её максимальное значение $\pi_{\max}$, то, очевидно, в строке t встречается (не в начале) её префикс длины $\pi_{\max}$, но не большей длины

Тогда, число новых подстрок, появляющихся при дописывании символа c, равно

$$s.size() + 1 - \pi_{\max}$$

Сжатие строки. Z-функция

Дана строка s длины n. Требуется найти самое короткое её «сжатое» представление, т.е. найти такую строку t наименьшей длины, что s можно представить в виде конкатенации одной или нескольких копий t

Для решения посчитаем Z-функцию строки s, и найдём первую позицию i такую, что $i + z[i] = n$, и при этом n делится на i. Тогда строку s можно сжать до строки длины i

Сжатие строки. Префикс-функция

Проблема в нахождении длины искомой строки t. Зная длину, ответом на задачу будет, например, префикс строки s этой длины

- Посчитаем по строке s префикс-функцию
- Рассмотрим её последнее значение, т.е. $\pi[n-1]$, и введём обозначение $k = n - \pi[n-1]$
- Если n делится на k, то это k и будет длиной ответа, иначе эффективного сжатия не существует, и ответ равен n