Хеширование

Минский ШАД. Осень

17 января 2015 г.

1 Обозначения

В данной домашней работе будет много задач на строки. Введём следующие обозначения:

- $|s|$ — длина строки s
- $s[i]$ — i-й символ строки s
- $s[i \dots j]$ — подстрока строки s, которая начинается в индексе i и заканчивается в индексе j
- $\overline{s}$ — «перевёрнутая» строка s
- $\text{ord}(c)$ — произвольная инъективная функция из алфавита строки в целые числа. Тут будем считать, что символы пронумерованы по алфафиту, т.е. $\text{ord}(a) = 1, \text{ord}(b) = 2, \dots$

Например, если s = «abacaba», то:

- $|s| = 7$
- $s[3]$ = 's'
- $s[1 \dots 3]$ = «bac»
- $\overline{s[1 \dots 3]}$ = «cab»

2 Тематические задачи

1. Предложить, как решать с помощью полиномиального хеширования следующие задачи (везде вам дана строка s, причём $|s| = n$):
   (a) [½ балла] По данным парам (l_1, r_1) и (l_2, r_2) отвечать на запрос, правда ли, что равны две строки $s[l_1 \dots r_1]$ и $s[l_2 \dots r_2]$ за $\mathcal{O}(1)$. Разрешается делать препроцесс за $\mathcal{O}(n)$
   (b) [½ балла] По данным парам (l_1, r_1) и (l_2, r_2) отвечать на запрос, правда ли, что равны две строки $s[l_1 \dots r_1]$ и $\overline{s[l_2 \dots r_2]}$ за $\mathcal{O}(1)$. Разрешается делать препроцесс за $\mathcal{O}(n)$
   (c) [½ балла] По данной паре (l, r) отвечать на запрос, правда ли, что строка $s[l \dots r]$ является палиндромом за $\mathcal{O}(1)$. Разрешается делать препроцесс за $\mathcal{O}(n)$
   (d) [½ балла] Найти по данным (i, j) найти длину наибольшего общего префикса двух строк $s[i \dots |s|]$ и $s[j \dots |s|]$ за $\mathcal{O}(\log n)$. Разрешается препроцесс за $\mathcal{O}(n)$
   (e) [½ балла] Вычислить z-функцию строки за $\mathcal{O}(n \log n)$ (т.е. найти z_i для всех $i = \overline{1 \dots |s|}$). z_i — длина наидлиннейшей подстроки, которая начинается в символе с индексом i и совпадает с префиксом строки.
   (f) [½ балла] Для пары (i, j) выяснить, какой префикс лексикографически меньше: который начинается в i или который начинается в j. Время работы $\mathcal{O}(\log n)$. Препроцесс за $\mathcal{O}(n)$

(g) [½ балла] Построить суффиксный массив для строки s за время $\mathcal{O}(n \log^2 n)$. i-суффиксом (suf_i) назовём подстроку $s[i \ldots |s|]$. Суффиксный массив a_i — перестановка первых n чисел, такая, что $\mathrm{suf}_{a_i} < \mathrm{suf}_{a_{i+1}}$ для любого $i = \overline{1 \ldots |s| - 1}$. Сравнение проводится лексикографически.

Решение:

Повторим идею полиномиального хеширования. Рассмотрим строку «abacaba». Зафиксируем число p (для определённости возьмём тройку), модуль N (для определённости возьмём 1234) и каждому индесу i поставим в соотвествие число $p^i \operatorname{ord}(s[i]) \mod N$. Также посчитаем куммулятивный массив таких сумм h (тут тоже суммируем по модулю N):

s_i	a	b	a	c	a	b	a
i	0	1	2	3	4	5	6
$\operatorname{ord}(s_i)$	1	2	1	3	1	2	1
p^i	1	3	9	27	81	243	729
a_i	1	6	9	81	81	486	729
h_i	1	7	16	97	178	664	159

Для удобства записи отождествим s_i и $\operatorname{ord}(s_i)$ в дальнейшем рассуждении.

Пусть нас спросили, равны ли подстроки $s[1 \ldots 2]$ и $s[5 \ldots 6]$. Поступим так же, как обычно поступают с куммулятивным массивом, т.е. попробуем посчитать сумму на подотрезке. Что мы получим для первой подстроки: $h_2 - h_0 = ps_1 + p^2 s_2 = 15$, для второй: $h_6 - h_4 = p^5 s_5 + p^6 s_6 =$ [Вычисления по модулю] $= 1215$.

Заметим, что полученные выражения отличаются лишь в показателях степеней. Естественный выход — домножить одну величину на «разность» между степенями, а именно $(h_2 - h_0)p^{5-1} = 15 \cdot 81 = 1215 = h_6 - h_4$. Т.е. хеши действительно совпали.

Таким образом, общий метод для проверки подстрок на совпадение:

- Проверить, что $r_1 - l_1 = r_2 - l_2$, иначе строки сразу не равны
- Пусть, не теряя общности, $l_1 \leqslant l_2$. Тогда проверим на равенство $(h_{r_1} - h_{l_1 - 1})p^{l_2 - l_1}$ и $h_{r_2} - h_{l_2}$. Если хеши не совпали, то строки различны. Иначе можно с высокой долей уверенности говорить, что строки тоже совпадают.

Для пункта b кроме хешей для строки s предпросчитаем хеши для строки $\overline{s}$. Теперь надо просто находить разности из двух разных куммулятивных массивов.

Для пункта c воспользуемся решением предыдущего пункта (положим $l_1 = l_2$, $r_1 = r_2$)

Для пункта d воспользуемся бинарным поиском по ответу, а именно будем перебирать длину этой подстроки и сравнивать подстроки описанным методом (пусть текущее значение длины строки — k, тогда можно положить $l_1 = i$, $r_1 = i + k - 1$, $l_2 = j$, $r_2 = j + k - 1$).

В пункте e надо лишь применить решение из d, считая, что $j = 0$.

В пункте f найдём длину самой длинной подстроки, i-префикса и j-префикс (пункт d), а затем сравним первый несовпадающий символ двух префиксов (просто посмотрев в строку на соответствующие места).

В пункте g просто отсортируем массив из первых n чисел с помощью компоратора, описанного в пункте f.

2. Предложить функцию для хеширования мультимножеств. А именно, по мультимножеству A и числу m ваша функция $h(A, m)$ должна выдавать число в диапазоне $0 \ldots . 2^m - 1$, такое что (можно

считать, что у вас есть хеш-функция для любого возможного элемента мультимножества, которая вычисляется за $\mathcal{O}(1)$):

(a) [$^1\!/_2$ балла] $h(A, m) = h(B, m)$, если $A = B$

(b) [$^1\!/_2$ балла] Функция должна быть легко обновляемая (т.е. при добавлении элемента в мультимножество должно быть можно пересчитать значение $h(A \cup \{x\}, m)$ за $\overline{o}(|A|^\varepsilon)$, для любого $\varepsilon > 0$)

(c) [$^1\!/_2$ балла] Функция должна быть сюръективна (можно считать, что функция хеширования элемента сюръективна)

(d) [3 балла] Функция должна быть стойкой. С целью упрощения будем считать, что функция стойкая, если выполняется хотя бы одно из двух:

- Рассмотрим конечное множество элементов B и будем считать, что все элементы мультимножества лежат в B. Зададимся числом n и рассмотрим множество мультимножеств $S_n = \{A : |A| \leqslant n\}$. Функцию будем называть стойкой, если $\forall m$ и для любого k $(0 \leqslant k < 2^m)$, $P\{h(A, m) = k | A \in S_n\} \to \frac{1}{2^m}$, при $n \to \infty$
- Функцию будет называть стойкой, если для достаточного большого m и $|A|$ $\nexists$ такая константа k, что $\forall |A|$ $\exists C, D$, такие что $|C| \leqslant k$, $|D| \leqslant k$ и $h(A, m) = h((A \cup C) \setminus D, m)$

Решение:

Для пункта a и b подходит, например $h \equiv 0$.

Для пункта c можно использовать, например, $h(A) = \bigoplus\limits_{x \in A} H(x)$

Для пункта d нужно использовать уже хорошие функции. Предыдущая функция не подходит, потому что можно дважды добавить один и тот же элемент и получить тот же хеш. Хорошей функцией может являться, например, такая $h(A) = \prod\limits_{x \in \mathrm{uniq}(A)} H(x)^{\mathrm{cnt}(x)}$, где cnt — кратность элемента x в A, а uniq — множество всех элементов мультимножества. За доказательством этого факта можно обратиться сюда.

3 Задачи на повторение

3. [$^1\!/_2$ балла] Дан отсортированный массив различных целых чисел. Надо определить, существует ли такой индекс i, что $a_i = i$. Сложность алгоритма должна быть $\mathcal{O}(\log n)$, где n — длина массива.

Решение:

Так как числа целые и различные, то $a_i \geqslant a_{i-1} + 1$. Рассмотрим функцию $f(i) = a_i - i$. Она неубывающая. Поэтому можно найти первую точку, где она не меньше нуля за $\mathcal{O}(\log n)$ с помощью бинарного поиска.

4. [$^1\!/_2$ балла] Пусть мы имеем два положительные неубывающие функции $f(x)$ и $g(x)$, причём $f(n) = \mathcal{O}(g(n))$. Правда, что $2^{f(n)} = \mathcal{O}(2^{g(n)})$? Если это может как выполняться, так и не выполняться, привидете примеры обоих случаев. Иначе докажите утверждение.

Решение:

Иногда это выполнятся, например $f(n) = g(n)$. В частности, это правда, если $f(n) \leqslant g(n)$, при $n \to \infty$.

С другой стороны, если, к примеру, $g(n) = 2f(n)$, то $2^{f(n)}$ и $2^{g(n)}$ отличаются уже не в константу раз.

5. [½ балла] Пусть у нас есть k отсортированных последовательностей из n чисел каждая. Предлагается такой алгоритм слияния их в одну: сначала сольём две первых последовательности, затем результат с третьей, и так далее. Какова сложность полученного алгоритма? Считаем, что слияние двух массивов происходит за их суммарную длину. Какова сложность полученного алгоритма.

Решение:

$$\sum_{i=1}^{k-1} n + in = n \sum_{i=1}^{k-1} i + 1 = \mathcal{O}(nk^2)$$

4 Практические задачи

6. [1 балл] Реализуйте решение задачи про поиск $i = a_i$.

7. [1 балл] Реализуйте решение задачи про суффиксный массив через хеши.

Задание	1	2	3	4	5	6	7	Сумма
Баллы	3½	4½	½	½	½	1	1	11½