Задача A. [Нечеткий поиск] (85 баллов)

Ограничение по времени: [0.5 секунд]
Ограничение по памяти: [64 Мб]

Даны две строки — P и T, длины не более 100 000. Строка T состоит только из строчных латинских букв. Строка P тоже состоит из строчных латинских букв, но еще может содержать от 0 до 10 символов ?, каждый из которых может заменять собой одну любую букву. Вам нужно найти все позиции i в строке T, начиная с которых возможно вхождение P в T, если каким-то образом заменить символы ? на буквы.

В первой строке входа — строка P, во второй — строка T. Длины обеих строк не превосходят 100 000, при этом они обе непустые.

В первой строке выведите число k — количество таких позиций i, что строка P может входить в строку T, начиная с позиции i. Во второй строке перечислите все возможные позиции в возрастающем порядке. Позиции нумеруются с нуля. Разделяйте две последовательные позиции одним пробелом.

тест	ответ
ab? ababcabc	3 0 2 5
??? ababcabc	6 0 1 2 3 4 5

Решение

Обозначим за q количество знаков вопроса в P. Строка P представляется в виде $P = ?^*s_1?^+s_2?^+\ldots?^+s_k?^*$, где $?^*$ означает знак вопроса, повторенный 0 или более раз, а $?^+$ означает знак вопроса, повторенный 1 или более раз. При этом, очевидно, $k \leqslant q+1$. Например, если $P = ?ab?c?def$, то $s_1 = ab$, $s_2 = c$, $s_3 = def$. Если $P = ab???c??$, то $s_1 = ab$, $s_2 = c$.

За линейное время по длине строки P можно найти это разбиение на знаки вопроса и непустые строки между ними. При этом запомним не только сами строки $s_1, s_2, \ldots, s_k$, но также для каждой строки s_i запомним отступ t_i от начала строки P до начала строки s_i.

Теперь давайте для каждой из строк s_i найдем все вхождения в строку T. Пусть эти вхождения начинаются с позиций $a_{i1}, a_{i2}, \ldots, a_{il_i}$. Тогда возможные позиции начала строки P, исходя из того, что ее подстрока s_i должна входить в T на t_i символов правее начала самой строки P,— это $a_{i1} - t_i, a_{i2} - t_i, \ldots, a_{il_i} - t_i$. Строка P может входить не во всех этих позициях в строку T, но она уж точно не может входить в T ни в каких других позициях, кроме этих.

Кроме того, если для какой-то позиции $a_{ij} - t_i$, определенной таким способом, окажется, что строка P, начиная с этой позиции, просто не помещается в строке T по длине, то эту позицию также необходимо отбросить. Как может возникнуть такая ситуация? Например, если $P = a??, T = aaa$. Тогда $s_1 = a$, и список позиций для s_1 состоит из позиций $1, 2, 3$. При этом с позиций 2 и 3 строка P просто не помещается в строке T. Этот случай никак нельзя отбросить на этапе поиска строки s_1 в T, его надо отсеивать явным образом.

Таким образом, для каждого i имеем некоторый список возможных позиций начала вхождения строки P в строку T. Какие-то из этих позиций могут быть ложными, однако никакая позиция, не входящая в список для s_i, заведомо не может быть позицией начала P в T.

Тогда давайте как-нибудь найдем все позиции, входящие в списки для каждого s_i. Любая такая позиция, очевидно, будет позицией начала P в T. Действительно, для любой такой позиции во-первых каждое s_i будет начинаться в P с t_i, а во-вторых P целиком помещается в T, а значит и для каждого знака вопроса в P найдется соответствующая ему буква в T, неважно какая, и P правильно вкладывается в T.

Списки позиций для каждого s_i мы находим за $O(|P|+|T|)$ алгоритмом КМП или Z-алгоритмом, либо быстрее, если применить алгоритм Ахо-Корасик и найти все эти списки сразу за $O(|P|+|T|+L)$, где L — суммарный размер всех этих списков. Как быстро найти пересечение этих списков? Можно воспользоваться следующим стандартным приемом. Заведем массив A размера $|T|$, каждая позиция которого соответствует возможной позиции начала P в T. Заполним массив A нулями. Далее, для

i-го списка пройдемся по всем элементам списка и добавим в соответсвующие позиции массива A по единице. После выполнения этого для всех списков найдем все позиции в массиве A, в которых написано число k (общее число строк s_i, соответственно число списков позиций). Эти позиции и будут позициями вхождения P в T, т.к. они входят во все наши k списков, а никакие другие позиции не входят.

Итого, сложность алгоритма составляет $O((|P| + |T|)q)$, где q — количество вопросов в строке P, т.к. мы выполняем КМП $O(q)$ раз, затем проходимся по $O(q)$ спискам длины $O(|T|)$, затем один раз проходим по массиву длину $O(|T|)$. Если использовать алгоритм Ахо-Корасик вместо КМП, то сложность в худшем случае получается $O(|P|+q|T|)$, т.к. $L \leqslant q|T|$. Памяти необходимо $O(|P|+|T|)$, т.к. на хранение всех строк s_i вместе со сдвигами t_i необходимо суммарно $O(|P|)$ памяти, а списки позиций для разных s_i можно не хранить одновременно, а сразу же добавлять в соответствующие позиции массива A единицы, а потом уже переходить к рассмотрению строки s_{i+1}. На изначальные строки нужно $O(|P|+|T|)$ памяти, плюс $O(|P|)$ на строки s_i, плюс $O(|P|+|T|)$ на выполнение одного КМП плюс $O(|T|)$ на массив A — всего $O(|P| + |T|)$.