Задача 1. Речник - Анализ

Автор: Антон Димитров

Решение 1

Нека последователността от думите в базата данни на сайта отбележим в D, а множеството от думите, които идват като заявки от самия сайт отбележим с Q.

Нека подредим думите от D лексикографски и наречем тази подредена последователност S. Тогава можем лесно да намерим с двоично търсене за всяка дума W дали тя се среща в D и ако се среща на коя дума от D съответства. Също така можем да намерим за всеки един префикс на W в кой интервал от думи се среща този префикс. Нека например имаме думите: **abcd, abcet, abcf, abdrr, atsy, bacd, bace, bbcc**. Тогава ако търсим в тях за думата **W=abc** ще видим, че тя не се среща сред тях, но префиксът и **abc** е общ с думите с номера от 1 до 3. Префиксът **ab** е общ с думите с номера от 1 до 3. А префиксът **a** е общ с думите с номера от 1 до 5.

След като имаме думите от D подредени лексикографски можем да търсим всяка дума от D в това множество. Нека търсим думите в реда, в който биха били обходени от алгоритъма описан в условието. За всяка поредна дума от D искаме да намерим за всеки неин префикс колко думи в D имат този префикс. Можем първо да търсим за префикса P_1 (състоящ се от първата буква). В намерения за него интервал можем да търсим за P_2 и така последователно за всички префикси на търсената думата.

Сред думите в D се срещат всякакви префикси. Ако потърсим всеки един в S ще намерим интервал от последователни думи, които започват с този префикс. Можем за първото срещане на всеки един такъв префикс да помним колко думи до момента, които започват с него са били обработени. По този начин, като обхождаме думите от D и ги търсим в S, за всеки префикс на търсената дума можем да видим колко от прегледаните до момента думи също са имали този префикс. Нужно е също така да увеличим брояча за този префикс с едно.

По този начим имаме статистика колко думи започват с даден префикс и колко думи преди дадена дума от D започват с даден префикс.

Нека:

PrefixCount(P) е броят думи в D, които започват с даден префикс P

Before(W, P) е броят думи, които започват с P и са преди думата W в D.

Position(W) е позицията на W в D

Тогава за всяка дума W от Q трябва да потърсим дали тя се среща в D.

Ако думата се среща тогава отговорът за нея е:

$\text{Position}(W) + \text{Before}(W, P_1) + \text{Before}(W, P_2) + \ldots + \text{Before}(W, P_L)$

Ако думата не се среща тогава отговорът е:

$N + PrefixCount(P_1) + PrefixCount(P_2) + \ldots + PrefixCount(P_L)$

Където N и L са съответно броят на думите в D и дължината на думата W.

Сложността на това решение е O(L * N * logN) за предварителната обработка на D, при която се пресмятат статистиките и O (L * logN) за всяка дума от Q.

Решение 2

Задачата може да се реши и като се построи така нареченето **Trie** (http://en.wikipedia.org/wiki/Trie). Можем да поставим думите от D в дървото. От всеки връх на това дърво ще излизат най-много 26 ребра, тъй като думите във входа съдържат само малки латински букви. Всяко ребро трябва да има етикет с буквата, на която съответства. С помощта на това дърво може да се намери за всяка дума от Q дали я има в D и ако да - на коя дума съответства. За всяка дума от Q това може да стане със сложност O(L), където L е дължината на думата, за която се пита. Все пак зависи от имплементацията на дървото, сложността може и да е по-голяма.

След това думите от D могат да се обходят в реда, в който алгоритъмът от условието би ги обходил и за всяка дума да се добави единица към всеки връх от дървото, през който се премине. Нека например на думите от Q съответстват числата R_1, R_2, …, R_Q, като R_i е номерът на думата в D, на която съответства i-та дума от Q. Думите от Q трябва да се сортират по това на коя дума от D отговарят. След това думите от D започват да се обхождат и да се добавя статистика за тях в дървото. Когато се достигне до дума от D, която се среща и в Q, думата от Q трябва да се използва, за да се премине по дървото и да се намери отговорът за нея, като този отговор е сума на стойностите във върховете, през които се преминава. Тези стойности във върховете показват колко от добавените думи в дървото започват с префикса, който съответства на дадения връх.

Трябва да се избере подходящ начин за представяне на дървото, като може ребрата да са статично заделени или да се създават динамично, когато има нужда от такива. В първия случай скоростта ще е по-добра, но пък използваната памет ще е повече, докато във втория многократното динамично заделяне на памет би забавило изпълнението, но паметта ще е точно толкова колкото е нужно. Разбира се многократното заделяне на памет може да се избегне ако предварително се задели някакъв блок от обекти, от който после да се вземат обекти и да се „закачат" към дървото. Както бе споменато, скоростта за намиране на правилното ребро, по което да се продължи в дървото зависи от имплементацията. Ако ребрата са статично заделени с константна сложност може да се избере по кое да се продължи. Ако ребрата са заделени динамично може да е нужно да се обходят, за да се намери по кое трябва да се тръгне.