

# Инопланетные омофоны

Автор задачи и разработчик: Александр Заварин

Первым делом определим пары одинаковых звуков. Представим, что звуки - вершины графа, пары одинаковых звуков - ребра. В таком случае нам нужно определить компоненты связности. Это можно определить простым проходом dfs, изначально присваиваем всем звукам индекс 0 (без индекса), идем по звукам, если у звука индекс 0, то запускаем dfs из этой вершинки, проставляя всем посещенным вершинам при этом одинаковый индекс, который до этого не выдавался другим группам одинаковых звуков. Таким образом получим массив индексов для каждого звука, по которым легко определить, считается ли пара звуков одинаковыми. Будем называть полученные индексы - индексами звучания, чтобы не путать с номером (индексом) звука в самом наборе.

Для того чтобы набрать хоть какие то баллы, переберем каждый запрос, будем идти по подстрокам в запросе одновременно. Пусть мы прочитали какую-то часть у этих подстрок. Переберем каждый звук, среди них простым проходом с каждой позиции в подстроках проверим вхождение этого звука. При проверке будем поддерживать номер звука, максимального по длине. Проверим, одинаковые ли индексы звучания у наших звуков - если нет, значит эти подстроки не омофоны и ответ «NO», иначе сместим указатель в каждой на длину соответствующего звука и повторим. Если в какой то момент мы прочитали одну подстроку полностью, то если подстроки являются омофонами, то и во второй мы тоже должны были дойти до конца, проверяем это и выводим ответ на запрос. Это решение проходит 1 подгруппу, асимптотика:  $O(q \cdot |t| \cdot S)$ .

Рассмотрим, как решить подгруппы, для которых  $b_i = d_i = |t|$ , это значит, что все подстроки в запросах являются суффиксами текста.

Для 2 подгруппы поймем, что мы повторяем некоторые действия: для каждой позиции в тексте номер звука, который бы прочитали инопланетяне (начиная с этой позиции), всегда определен однозначно, в прошлом решении мы могли его считать несколько раз в разных запросах. Тогда для каждой позиции предподсчитаем номер звука, который входит с этой позиции как подстрока в текст и максимален по длине. Теперь для каждого запроса просто будем "прыгать" по уже насчитанным звукам, проверяя их на одинаковость по звучанию и перемещая указатель на длину звука для каждой подстроки. Такое решение имеет асимптотику:  $O(|t| \cdot S + q \cdot |t|)$ .

Заметьте, что такое решение не пройдет 1 подгруппу, так как мы пользуемся свойством  $b_i = d_i = |t|$ . В общем случае предподсчет максимального звука для каждой позиции (будем называть это **максимальным прыжком** для каждой позиции) не всегда соответствует макс. прыжку в подстроке в этой же позиции (так как подстрока может заканчиваться раньше и следовательно некоторые звуки просто в нее по длине не поместятся, хотя в тексте они могли бы поместиться).

Продолжим улучшать наше решение. Перейдем к 3 подгруппе.

Так как все подстроки в запросах являются суффиксами, то на самом деле в запросах всего максимум  $|t|$  различных подстрок. Осталось придумать, как проверять очень быстро их на одинаковость по звучанию, очень хочется это делать аж за  $O(1)$  после некоторого предподсчета. На помощь нам приходит хеширование. Давайте для каждого суффикса узнаем число звуков, которые мы выговариваем при его прочтении и полиномиальный хеш из индексов звучания для каждого звука, тогда потом для того чтобы два суффикса читались одинаково, достаточно проверить, что у них одинаковые число звуков и хеш индексов звучания. Эти две вещи и **будем называть хешем далее**. Теперь как это предподсчитать, в прошлой подгруппе мы научились предподсчитывать для каждой позиции в тексте максимальный "прыжок"; тогда пойдем в тексте справа налево и последовательно для каждого суффикса насчитаем нужную нам информацию. Для суффикса длины 1 просто запишем, что у него 1 звук и хеш состоит из одного индекса звучания латинской буквы. Далее двигаемся влево, для каждой позиции мы знаем макс прыжок, а так как все суффиксы справа мы уже насчитали, то и оставшуюся часть суффикса мы тоже знаем, берем информацию у суффикса на расстоянии макс прыжка (макс звука) справа и обновляем хеш, добавляя туда индекс звучания нашего макс звука, а число звуков увеличиваем на 1. Таким образом мы получим нужную информацию про каждый суффикс и будем обрабатывать каждый запрос за  $O(1)$ . Итоговая асимптотика:  $O(|t| \cdot S + q)$ .

Для 4 подгруппы осталось придумать, как быстро насчитать макс прыжки для каждой позиции в тексте. Вспомним замечательный алгоритм Ахо-Корасик, развернем каждый звук и на полученных строках построим бор. Посчитаем на боре суффиксные ссылки как в Ахо-Корасике, а также терминальные суффиксные ссылки из каждой вершины в ближайшую терминальную (терминальная вершина бора - вершина, где заканчивается какой-то звук). Теперь пойдем по тексту справа налево и будем перемещаться по нашему бору, как в алгоритме поиска строк в тексте: пытаемся сделать переход по букве, если не выходит, прыгаем по суффиксной ссылке и повторяем попытку. Пусть мы прошли какую-то часть текста, параллельно перемещаясь по бору, давайте посмотрим, куда для вершинки бора ведет терминальная суффиксная ссылка. Во-первых, если мы находимся в какой-то вершине бора, это значит, что если мы будем спускаться по бору вниз, то в тексте мы будем перемещаться с текущей позиции по буквам вправо. Т.е. если мы находимся в некоторой позиции в тексте, то строка, полученная из букв при спуске по бору из текущей вершины до корня, будет входить как подстрока, начиная с текущей позиции в тексте. При переходе по суффиксной ссылке мы просто удаляем какое-то число символов в конце этой строки, поэтому при переходе по терминальной суффиксной ссылке мы попадем в вершинку бора такую, что строка из букв, начиная с этой вершинки и до корня, входит как подстрока, начиная с нашей позиции в тексте, она является звуком и при этом длина максимальна, это ровно то, что нам нужно. Предподсчет суффиксных ссылок работает за  $O(26 \cdot S)$ , проход по тексту суммарно за  $O(|t|)$ , оставшаяся часть решения такая же, как в прошлой подзадаче, итого асимптотика:  $O(26 \cdot S + |t| + q)$ . Заметьте, что в каждой позиции мы получаем звук максимальной длины из одной терминальной суффиксной ссылки, не прыгая по суффиксным ссылкам дальше, как в обычном алгоритме Ахо-Корасика для поиска всех вхождений, поэтому число вхождений подстрок никак не влияет на асимптотику.

Теперь рассмотрим 5 подгруппу. В ней во всех запросах все подстроки являются префиксами текста. Давайте вернемся к более простому решению из 3 подгруппы, за квадрат для каждой позиции в тексте найдем макс. прыжок. Теперь рассмотрим, как выглядит последовательность звуков для некоторого префикса: мы прыгаем по макс прыжкам, набирая параллельно хеш, в какой то момент макс. прыжок будет перепрыгивать границу нашего префикса, получается на самом деле все, чего нам не хватает - это того, как выговаривается какой то префикс нашего звука, который слишком длинный, если бы мы знали эту информацию, то мы бы легко смерджили два хеша (вспомните, что мы дополнительно храним число звуков, поэтому для мерджа нам надо просто один из хешей умножить на основание хеша в степени числа звуков). В таком случае мы хотим посчитать хеш для каждого префикса у наших звуков (разве что для префикса, являющегося самим звуком, считать это не надо, для них хеш - это просто индекс звучания нашего звука). Воспользуемся динамическим программированием: отсортируем все строки по длине, для звуков - букв латинского алфавита хеш префикса очевиден, далее предположим, мы предподсчитали дп для всех строк длины  $< k$ . Теперь пусть мы рассматриваем некоторый звук длины  $k$ , пусть мы находимся в начале этого звука, тогда за  $O(S)$  мы можем определить макс. звук, который мы можем выговорить при этом, не выговаривая весь звук полностью. Важно заметить, что мы никогда не должны залезать на последнюю позицию звука, так как звук выговаривать полностью мы и так умеем. Заметим, что все звуки, которые мы будем выговаривать, будут длины  $< k$ , следовательно, для них насчитано дп, а значит, мы можем просто перенести результаты префиксов звука, который мы выговариваем с первой позиции, в наше дп. Заведем глобальный хеш и запишем туда наш выговоренный звук, сместимся в нашем изначальном звуке длины  $k$  на длину выговоренного и повторим эти же действия, только при переносе результатов дп будем их мерджить с глобальным хешем. Таким образом за  $O(S^2)$  предподсчитаем наше дп для всех звуков. Теперь, чтобы посчитать хеш для каждого префикса текста, также завеем глобальный хеш (изначально равный 0) начнем с первой позиции, за  $O(S)$  найдем макс. звук, перенесем результаты его дп для каждого префикса, а его добавим в глобальный хеш, сместимся и повторим те же действия, только при переносе результатов дп будем их мерджить с глобальным хешем. Далее отвечаем на каждый запрос за  $O(1)$ , получаем асимптотику:  $O(|t| \cdot S + S^2 + q)$ .

Теперь вернемся к общему случаю задачи.

Для решения 6 подгруппы мы хотим для каждой подстроки предподсчитать ее хеш, чтобы так же, как и в прошлых подгруппах, обрабатывать запросы за  $O(1)$ . Для каждой позиции в тексте завеем битовый массив размера  $|t|$ , в  $k$ -й ячейке будет лежать 1, если начиная с этой позиции есть

звук длины  $k$ , который входит как подстрока, это легко посчитать за  $O(|t| \cdot S)$ . Теперь переберем левую границу подстроки  $l$ , далее будем перебирать правую границу от  $l$  до  $|t|$ , поддерживая при этом хеш подстроки. Для подстроки длины 1 хеш очевиден, так же мы будем отмечать позиции важными, если на данный момент начиная с них мы выговариваем звук при прочтении подстроки. Пусть мы обработали уже какое то число правых границ и у нас есть хеш подстроки  $[l, r - 1]$ , а так же отмечены важные позиции, хотим перейти к подстроке  $[l, r]$ . Переберем все важные позиции от  $l$  до  $r$ , среди них выберем такую наименьшую позицию  $i$ , что из  $i$  позиции можно сказать звук длины  $r - i + 1$ , если такая есть, то обновим хеш, убрав из нее хеш всех звуков, которые мы выговариваем, начиная с  $i$  позиции (и убрать маркеры важных позиций для них, разве что кроме позиции  $i$ ) и добавим новый звук, который говорим из  $i$  позиции, если такой позиции нет, значит мы добавляем в наш хеш звук-букву из позиции  $r$  и отмечаем ее важной. Тогда получаем решение за куб:  $O(|t|^3 + |t| \cdot S + q)$ . Так же возможны другие вариации решения, доказательство этого остается читателю в качестве упражнения.

Для 7 подгруппы используем идею из 5 подгруппы для предподсчета дп у всех звуков за  $O(S^2)$  и так же для каждой позиции в тексте предподсчитаем макс. прыжок, но в этой подгруппе будем обрабатывать запросы онлайн. Для запроса нам для каждой подстроки надо узнать хеш. Для того чтобы для подстроки узнать хеш изначально поставим его равным 0, начнем с первой позиции, если макс. прыжок не перепрыгивает правую границу подстроки, добавим его в хеш и сдвинемся на длину прыжка, если полностью набрали подстроку, остановимся, мы получили нужный хеш, иначе как только найдем прыжок, который перепрыгивает границу, узнаем результат дп для нужного префикса в нем и смержим хеш из дп с нашим. Таким образом получаем решение за  $O(|t| \cdot S + S^2 + q \cdot |t|)$ .

Для 8 подгруппы воспользуемся идеей с Ахо-Корасиком из 4 подгруппы, чтобы за  $O(26 \cdot S + |t|)$  предподсчитать все макс. прыжки в тексте. Так же нам надо научиться считать дп для звуков быстрее квадрата. Для этого нам нужно оптимизировать поиск макс. звука при подсчете дп в самом звуке. Заметим, что похожее мы уже делали в тексте, бор для Ахо-Корасика у нас уже есть, тогда давайте просто для каждого звука пройдемся с права налево, начиная с предпоследней позиции (это важно, так как нам нельзя залезать на последнюю букву) и для каждой позиции предподсчитаем макс. звук как для обычного текста, на это мы потратим  $O(S)$  действий. Для запросов искать хеш подстрок будем так же, как и в 7 подгруппе, из-за этого в нашей асимптотике появится  $O(q \cdot |t|)$ , итого получаем:  $O(26 \cdot S + |t| + S + q \cdot |t|) = O(26 \cdot S + q \cdot |t|)$ .

Чтобы сдать 9 и 10 подгруппы, осталось придумать, как быстрее обрабатывать наши запросы. Для каждой позиции в тексте мы знаем макс. прыжок, тогда давайте построим разреженную таблицу на этих прыжках, которая будет для каждой степени двойки  $2^k$  хранить хеш из  $2^k$  звуков, которые мы выговариваем при этих  $2^k$  прыжках и позицию, в которой мы после этих прыжков окажемся, на предподсчет мы потратим  $O(|t| \log |t|)$ . Тогда теперь в запросе мы можем для каждой подстроки за  $O(\log |t|)$  найти позицию, после которой макс прыжок перепрыгивает границу и так же хеш макс. звуков, которые мы выговаривали по пути к этой позиции. Получим потрясающую асимптотику:  $O(26 \cdot S + |t| \cdot \log |t| + q \cdot \log |t|)$ .

В 4, 8 и 9 подгруппы можно сдать оверкилл решения, в которых в асимптотике добавляется  $O(S \cdot \sqrt{S})$  или  $O(|t| \cdot \sqrt{S})$ . Решение в Ахо-Корасике не разворачивает звуки и пользуется фактом, что когда мы находимся в некоторой вершине бора, переходов до корня по терминальным суффиксным ссылкам будет не больше  $\sqrt{S}$ .

Так же есть красивое решение, которое проходит все подгруппы с ограничением  $|t| \leq 6000$ . В нем мы сохраняем все звуки в бор, далее для каждой позиции за  $O(|t|^2)$  узнаем все звуки, которые можно выговорить с этой позиции и сортируем их по длине. Мы хотим для каждой подстроки узнать ее хеш. Будем перебирать начальную позицию  $i$ , теперь запустим рекурсивную функцию *func* следующего вида: она принимает хеш  $h$ ,  $l$  - позицию, с которой мы хотим выговорить звук,  $r$  - правую границу, дальше которой мы не можем выговаривать (изначально положим хеш  $h = 0$ ,  $l = i$ ,  $r = |t|$ ). Теперь что мы будем делать в функции, возьмем минимальный звук, который мы можем выговорить с позиции  $l$  (это звук из одной буквы), добавим в хеш  $h$  этот звук и запишем для соответствующей подстроки  $[i, l]$ . Теперь если в этой же позиции  $l$  есть следующий звук большего размера  $k$ , тогда запустим нашу же функцию от следующих аргументов: *func*( $h +$  (звук из одной буквы),  $l + 1, l + k - 2$ ), она посчитает ответы для всех подстрок от  $[i, l + 1]$

до  $[i, l + k - 2]$ , для подстроки  $[i, l + k - 1]$  хеш - это хеш  $h$ , в который добавили наш звук длины  $k$ . Теперь посмотрим, есть ли с этой же позиции  $l$  еще больший звук, пусть он есть и он длины  $p$ , тогда повторим тоже самое запустим функцию  $func(h + (\text{звук длины } k), l + k, l + p - 2)$  и далее опять посмотрим, есть ли звук большего размера. Если в какой то момент звука большего размера не оказалось или он залезает дальше границы  $r$ , просто запустим функцию с ограничением  $r$ :  $func(h + (\text{звук максимальной длины } f, \text{ не залезающий за границу}), l + f, r)$ . Это работает за  $O(|t|^2)$ . Итоговая асимптотика  $O(|t|^2 + S + q)$ .