S©MR ISSN 1813-3304

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ

Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru

Том 10, стр. 538–550 (2013) УДК 519.16
MSC 68W32

ДВЕ ЗАДАЧИ О ВОССТАНОВЛЕНИИ ПОВРЕЖДЕННЫХ СТРОК

М.В. РУБИНЧИК, Ю.В. ГАМЗОВА

Abstract. We consider two problems related to pattern matching in damaged strings. In both problems, the goal is to recover the original undamaged text and pattern in an optimal way.

Problem 1. *For given damaged text and damaged pattern, recover the text and the pattern in a way that maximizes the number of occurrences of the pattern in the text.*

We define *total Hamming distance* between a text of length n and a pattern of length m to be the sum of Hamming distances for all pairs (pattern, factor of length m of the text).

Problem 2. *For given damaged text and damaged pattern, recover the text and the pattern in a way that minimizes the total Hamming distance between them.*

We prove both problems to be NP-hard and provide efficient algorithms to various polynomially solvable subcases of these problems.

Keywords: damaged string, partial strings, pattern matching.

1. Введение

Задача поиска подстроки в строке — это одна из самых известных и хорошо изученных задач теории строк. Ее стандартная постановка состоит в следующем. Дано две строки — более длинный "текст" s и более короткий "шаблон" p, требуется найти все вхождения шаблона в текст, т. е. все пары (i, j) такие, что $s[i...j] = p$. Эта задача имеет большое прикладное значение для информационного поиска и биоинформатики, и это значение непрерывно возрастает с развитием информационных технологий. Существует множество вариаций данной задачи (например, с ограничениями на используемую память или на доступ к

Rubinchik M.V., Gamzova Yu.V., Two problems about recovering of damaged strings.

Поступила 25 июля 2013 г., опубликована 3 сентября 2013 г.

строкам, с приблизительным поиском, с "неточным" шаблоном, и т.п.). В данной работе мы рассматриваем две задачи, близкие к одному из естественных вариантов задачи поиска подстроки в строке.

Одно из наиболее интересных с практической точки зрения обобщений задачи поиска подстроки в строке состоит в предположении, что текст и/или шаблон могут быть повреждены при передаче данных (например, из-за шума в канале, или при распознавании печатного текста, или при точечных мутациях, если речь идет о биологических "строках", таких как цепочки ДНК). Поврежденные символы нельзя идентифицировать однозначно, но для каждого из них можно указать множество символов исходного алфавита, из которых в результате повреждения мог получиться данный символ. Таким образом, каждому символу алфавита поврежденных строк поставлено в соответствие некоторое множество символов исходного алфавита, т.е. между этими алфавитами определено бинарное отношение — *отношение совместимости*. Две строки совместимы, если их длины равны и буквы, стоящие в одинаковых позициях, совместимы.

Рассмотрим пример. Пусть есть текст на русском языке. Текст был распечатан на струйном принтере, но при передаче попал под дождь и был поврежден. Предположим, что в поврежденном тексте встретился символ «I». Мы можем сделать вывод, что в данной позиции в исходном тексте могла быть, например, буква «Б», «П», «В», но не могла быть буква «О». Алгоритмы решения классической задачи поиска в поврежденных строках будут обсуждены ниже, в пункте 1.2.2.

В данной работе рассматриваются две задачи, в которых поиск играет вспомогательную роль, а цель — восстановить поврежденный текст и поврежденный шаблон оптимальным образом. Постановки задач различаются выбором критерия оптимальности.

Задача 1. *Заданы поврежденный текст и поврежденный шаблон, необходимо среди всех возможных вариантов восстановления исходных неповрежденных текста и шаблона выбрать пару, для которой количество вхождений шаблона в текст максимально.*

Для частных случаев задачи 1 (когда поврежден только текст или только шаблон) мы приведем эффективные (полиномиальные по времени) алгоритмы, а в общем случае докажем, что данная задача NP-трудна даже если алфавит текса и шаблона — бинарный.

Задача 2. *Заданы поврежденный текст и поврежденный шаблон, необходимо среди всех возможных вариантов восстановления исходных неповрежденных текста и шаблона выбрать пару, для которой минимальна «непохожесть», вычисляемая как сумма расстояний Хэмминга для всех пар (шаблон, подстрока той же длины в тексте).*

Для частных случаев задачи 2 (для бинарного алфавита; для случая, когда поврежден только текст или только шаблон; для частичных циклических строк) мы приводим полиномиальные по времени алгоритмы, в общем случае докажем, что задача NP-трудна и даже APX-трудна. Выдвинута гипотеза об NP-трудности задачи для случая частичных строк.

Отметим, что в варианте распознавания (существует ли вариант восстановления с заданным значением критерия оптимальности) и задача 1, и задача 2

принадлежат классу NP: полиномиальным сертификатом является пара (восстановленный текст, восстановленный шаблон), и его корректность очевидно проверяется за полиномиальное время.

2. Предварительные сведения

2.1. **Основные определения.** Пусть заданы конечные множества Σ, Γ ($\Sigma \cap \Gamma = \varnothing$) и бинарное отношение ρ на $\Sigma \cup \Gamma$, состоящее из всех пар (a, a), где $a \in \Sigma$, и некоторых пар (a, b), где $a \in \Sigma$, $b \in \Gamma$, с условием, что каждый элемент из Γ принадлежит хотя бы одной паре из отношения. Будем называть Σ *основным алфавитом* (или просто алфавитом), Γ — *множеством поврежденных символов*, $\Sigma \cup \Gamma$ — *поврежденным алфавитом*, а ρ — *отношением совместимости* на нем.

Строку над поврежденным алфавитом будем называть *поврежденной*. Строку над исходным алфавитом будем называть неповрежденной или полной. Неповрежденную строку s и поврежденную строку w будем называть *совместимыми*, если их длины равны и $s[i] \rho w[i]$ для любого i от 1 до $|s|$. Две поврежденных строки s и p будем называть *совместимыми*, если существует неповрежденная строка, совместимая с каждой из них. Комбинаторные задачи на алфавитах с отношением совместимости рассматривались в ряде работ (см., например, [8]).

Рассмотрим пример. Пусть множество поврежденных символов состоит из одного символа и этот символ совместим со всеми символами исходного алфавита. Строки с таким отношением совместимости представляют собой так называемые *частичные строки*. Поврежденный символ в этом случае будем называть *джокером* и обозначать $\diamond$. Таким образом, строки с отношением совместимости являются обобщением частичных строк, которые, в свою очередь, являются обобщением обычных строк.

В дальнейшем текст s и шаблон p считаются поврежденными строками. Кроме того, положим $n = |s|$, $m = |p|$.

Расстоянием Хэмминга между строками u и v одинаковой длины называется число $d(u, v) = |\{i \mid u[i] \neq v[i]\}|$. *Префикс-функция* строки w — это функция $\pi(i)$, которая для каждого i от 1 до $|w|$ возвращает длину самого длинного префикса строки u, который одновременно является суффиксом строки $w[1...i]$. *Лексикографическим порядком* на Σ^* называется линейный порядок, определяемый произвольным фиксированным линейным порядком на алфавите. *Суффиксный массив* строки w — это массив, состоящий из всех суффиксов строки w, отсортированных лексикографически. На практике суффиксный массив задается как массив ссылок на суффиксы: $SUF[i] = j$ означает, что суффикс $w[j...|w|]$ имеет номер i в отсортированном лексикографически списке суффиксов. Параллельно с суффиксным массивом строки обычно вычисляют массив наибольших общих префиксов LCP: $LCP[i]$ есть длина наибольшего общего префикса суффиксов $SUF[i]$ и $SUF[i + 1]$, соседних в суффиксном массиве. Вычислить оба массива для строки w можно за время $O(|w|)$ [6].

2.2. **Исторический обзор.** Для задачи поиска подстроки в строке существуют различные алгоритмы, решающие ее за линейное время. Например, алгоритм Бойера-Мура, алгоритм Кнута-Морриса-Пратта, и множество других (См., например, [2]).

Несмотря на обилие алгоритмов решения задачи, ни один из них не обобщается для решения данной задачи для строк с произвольным отношением совместимости. Более того, не существует линейного алгоритма решения задачи поиска даже для частичных строк.

В статье [1] был приведен алгоритм, решающий задачу поиска подстроки для частичных строк за время $O(n \log n \log |\Sigma|)$. Позднее в статье [3] эта идея была использована для построения алгоритма, решающего задачу для произвольного отношения совместимости за время $O(|\Sigma| n \log n)$. Опишем кратко это решение.

За $O(|\Sigma \cup \Gamma|)$ действий можем перебрать все символы алфавита. При фиксированном символе a алфавита выполним следующие действия. Мы хотим найти для каждой подстроки из s длины m количество совпадений символа a в этой подстроке с совместимыми с ним символами из p. Для этого заменим в строке a на 1, а остальные символы на 0. В шаблоне заменим все символы, совместимые с a, на 1, а остальные на — 0. Получим две битовых строки. Строку, полученную из шаблона, перепишем в обратном порядке (справа налево). Возьмем два многочлена, в которых коэффициентами будут элементы полученных битовых строк, и перемножим их. Это действие называется *булевой конволюцией*. Алгоритм Шенхаге-Штрассена, использующий быстрое преобразование Фурье, решает эту задачу за время $O(n \log n)$ [4]. Откинув в полученном произведении первые $m-1$ и последние $m-1$ коэффициентов, мы получим массив количеств совпадений a с совместимыми ему символами для каждой подстроки длины m строки s. Наконец, просуммируем полученные для всех a массивы и проверим каждую ячейку итового массива на равенство длине шаблона. Таким образом, общая сложность алгоритма есть $O(|\Sigma| n \log n)$.

Рассмотрим улучшение приведенного выше алгоритма для случая частичных строк. Закодируем каждый символ основного алфавита битовой строкой, а для джокера зарезервируем два кода: из одних нулей и из одних единиц. Длину кода выберем равной $k = \lceil \log(|\Sigma| + 2) \rceil$. Таким образом, при кодировании текст и шаблон заменяются на битовые строки длины kn и km, соответственно. Для этого хватит длины $O(\log |\Sigma|)$. Заменим в шаблоне все джокеры на строку из всех единиц, а в тексте на строку из всех нулей. Выполним булеву конволюцию. Аналогично, заменим все джокеры в шаблоне на строку из всех нулей, а в тексте — на строку из всех единиц. Еще раз посчитаем булеву конволюцию. Шаблон совместим с подстрокой в том и только том случае, когда соответствующий коэффициент первой конволюции равен числу единиц в коде подстроки, а коэффициент второй конволюции — числу единиц в коде шаблона.

В статье [5] данный алгоритм был доработан и было доказано, что для произвольного отношения совместимости можно найти все вхождения шаблона в текст за $O(cn \log n)$, где c — минимальное количество двудольных полных подграфов, объединение которых представляет собой граф конфликта (дополнение к двудольному графу, задающему отношение совместимости).

3. Задача максимизации числа вхождений

Итак, у нас есть две поврежденных строки — текст s длины n и шаблон p длины m. Требуется найти такие неповрежденные строки v и u, где v совместима с s, а u совместима с p, чтобы количество вхождений u в v было максимально. Фактически, требуется в каждой из строк (s, p) заменить каждый

поврежденный символ на какой-то из совместимых с ним неповрежденных. Таким образом мы и получим строки v и u соответственно.

Ниже будут представлены полиномиальные по времени алгоритмы для двух частных случаев. В общем виде задача остается NP-трудной даже для случая частичных строк над бинарным алфавитом.

3.1. **Решение для неповрежденного текста.** Пусть текст s не поврежден. Сформулируем простейший алгоритм решения. Переберем i от 1 до $n-m+1$. Для фиксированного i проверим, может ли шаблон входить начиная с i. Если может, то считаем, что одно из вхождений будет начиная с i. В таком случае автоматически определяются замены всех поврежденных символов шаблона на символы алфавита. Теперь текст и шаблон уже неповрежденные строки. Эта задача решается за время $O(n+m)$, например, с помощью префикс-функции. Затем из всех решений (для каждого i) нужно выбрать наилучшее. Итого общая временная сложность решения $O((n-m)(n+m)) = O(n^2)$.

Покажем теперь, что можно добиться более оптимального решения. Возьмем массив a, в который поместим все подстроки текста s длины m. Их будет $n-m+1$. Отсортируем массив лексикографически. Используя обменную сортировку, мы получим не более $O(n \log n)$ сравнений. Одно сравнение производится за $O(m)$, т.е. общая сложность $O(nm \log n)$. Теперь пусть $a[i] = a[j]$ для некоторых $1 \le i \le j \le n$, тогда для всех k таких, что $i \le k \le j$ верно $a[k] = a[i]$. Другими словами равные подстроки образуют непрерывные блоки в массиве. Выберем те блоки, строки которых равны шаблону p, а из них выберем тот, который содержит максимальное количество строк. Строки, находящиеся в этом блоке, и будут ответом на задачу, а количество строк в блоке будет конечным максимальным количеством вхождений шаблона.

Сортировка за $O(nm \log m)$ по-прежнему слишком медленна. Покажем, как ее ускорить.

Предложение 1. *Существует алгоритм, решающий задачу 1 для случая неповрежденного текста за время $O(n \log n \log |\Sigma|)$ для частичных строк и за время $O(|\Sigma| n \log n)$ для произвольного отношения совместимости.*

Доказательство. Применим изложенный в разделе 2.2 алгоритм Фишера-Патерсона и найдем все вхождения шаблона в строку за время, указанное в условии предложения. Теперь введем на основном алфавите произвольный линейный порядок и вычислим для текста суффиксный массив и массив LCP. Все подстроки длины m в тексте — это в точности все префиксы длины m суффиксов текста. Значения в массиве LCP разделяют суффиксный массив на блоки суффиксов, имеющих один и тот же префикс длины m. при любом восстановлении шаблона он совпадет только с одним таким префиксом. Среди блоков, пересекающихся со множеством позиций, найденных алгоритмом Фишера-Патерсона, найдем длиннейший. Общий префикс этого блока и будет ответом. □

3.2. **Решение для неповрежденного шаблона.**

Предложение 2. *Существует алгоритм, решающий задачу 1 для случая неповрежденного шаблона за время $O(nm)$.*

Доказательство. Будем решать задачу методом динамического программирования.

Гранями строки w называются такие числа i, что $w[1...i] = w[|w|-i+1...|w|]$. Вычислив для шаблона p префикс-функцию, мы можем получить все грани строки p: это, в порядке убывания, числа $\pi(|p|), \pi(\pi(|p|)), \ldots$. Известно ([7]), что префикс-функцию можно вычислить за линейное время, т.е. все грани p вычисляются за $O(m)$.

Пусть $a_1 < a_2 < \ldots, a_t$ — последовательность всех индексов, с которых начинаются вхождения шаблона p в текст s; эти индексы можно найти за $O(nm)$ сравнением p со всеми подстроками s длины m.

Рассмотрим некоторый индекс a_i и все возможные неповрежденные строки, совместимые с исходным текстом s и тоже содержащие вхождение шаблона p, начиная с a_i. Среди всех таких неповрежденных строк найдем такую строку v, для которой количество вхождений шаблона p в префикс длины $a_i + m - 1$ максимально. Обозначим это максимальное количество через f_i.

Имеем $f_1 = 1$ по определению a_1. Скажем, что вхождения шаблона p в позициях a_i и a_j *согласованы*, если неповрежденная строка, совместимая с s, может содержать оба указанных вхождения p. Тогда при $i > 1$ получаем $f_i = 1 + \max\{f_j \mid j < i\}$, вхождения в позициях a_i и a_j согласованы.

При $j < i$ согласованность эквивалентна выполнению одного из двух условий: $a_i - a_j \geq m$ или $m - a_i + a_j$ — грань p. Таким образом, согласованность проверяется за константное время. Теперь заметим, что если $i > 2m$, то вхождение шаблона в позиции a_{i-m} согласовано как с вхождением в a_i, так и со вхождением в a_{i-2m}. Тогда $f_{i-m} \geq 1 + f_{i-2m}$, а значит, при вычислении f_i максимум не может быть достигнут на $j = i - 2m$ (и на меньших значениях j). Следовательно, при вычислении f_i можно добавить условие $j > i - 2m$, что гарантирует вычисление f_i за $O(m)$ операций.

Осталось заметить, что требуемое в Задаче 1 количество вхождений есть $\max\{f_i \mid 1 \leq i \leq t\}$ и $t \leq n$. Предложение доказано. □

Замечание 1. *В типичном случае $t \ll n$, а при этом условии решение Задачи 1 с неповрежденным шаблоном оказывается не сложнее поиска всех вхождений шаблона. В частности, если m не слишком мало, можно использовать оценку из алгоритма Фишера-Патерсона.*

3.3. **Доказательство NP-трудности в общем случае.** Для доказательства NP-трудности задачи 1 сведем задачу о поиске максимальной клики в графе к задаче 1 над бинарным алфавитом (при этом, очевидно, поврежденный символ может быть только один — джокер). Пусть у нас есть граф из n вершин и m ребер. Возьмем граф дополнения и занумеруем его ребра от 1 до $k = \frac{n(n-1)}{2} - m$. Поставим в соответствие каждой вершине j частичную строку w_j длины k над алфавитом $0, 1, \diamond$. Пусть ребро с номером i соединяет вершины a и b. Тогда $w_a[i] = 1$, $w_b[i] = 0$ (или наоборот) и $w_j[i] = \diamond$ для $j \notin a, b$.

Лемма 1. *В исходном графе ребро между двумя вершинами a и b существует тогда и только тогда, когда w_a совместима с w_b.*

Доказательство. По построению очевидно, что w_a совместима с w_b тогда и только тогда, когда ни одно из ребер графа дополнения не равно (a, b). □

Пусть вершины графа занумерованы от 1 до n. Положим $v_j = 1 w_j 10^{k-1}$, $s = v_1 ... v_n$, $p = 1 \diamond^k 1$.

Лемма 2. *Все вхождения шаблона p в текст s начинаются с индексов вида $1 + i(2k + 3)$, где i от 0 до $n - 1$.*

Доказательство.

(1) $$s = 1\underbrace{w_1}_{k}1\underbrace{00\ldots0}_{k+1}\ldots1\underbrace{w_n}_{k}1\underbrace{00\ldots0}_{k+1},$$

(2) $$p = 1\underbrace{\diamond\diamond\ldots\diamond}_{k}1.$$

Очевидно, что с каждой позиции указанного вида начинается вхождение p в s, а других вхождений нет из-за длинных блоков нулей. □

Лемма 3. *Пусть дано множество частичных строк и полная строка w, совместимая с каждой частичной строкой множества. Тогда все строки множества попарно совместимы.*

Доказательство. Очевидно, что длины всех частичных строк в множестве равны. Рассмотрим некоторое i от 1 до $|w|$. Тогда $w[i]$ — неповрежденный символ, совместимый с i-ми символами всех строк нашего множества. Значит, эти символы либо совпадают с $w[i]$, либо являются джокерами. □

Теорема 1. *Задача восстановления текста и шаблона с максимизацией количества вхождений шаблона в текст NP-трудна даже в случае бинарного алфавита.*

Доказательство. По лемме 2 все вхождения шаблона p в текст s начинаются только с индексов $1{+}i(2k{+}3)$ где $i = 0..n{-}1$. Поскольку шаблон имеет длину $k{+}2$, вхождения согласованы. Наша задача состоит в том, чтобы найти наибольшую клику в заданном графе, что по лемме 1 эквивалентно поиску наибольшего подмножества попарно совместимых строк из множества $\{w_1, w_2, \ldots, w_n\}$. Заметим, что w_i совместимо с w_j тогда и только тогда, когда v_i совместимо с v_j. Теперь пусть мы решили задачу максимизации вхождения шаблона p в текст s. Другими словами, мы нашли такую полную строку u, совместимую с p, что количество совместимых с ней строк из множества $\{v_1, v_2, \ldots, v_n\}$ максимально. Заметим, что из того, что v_i совместимо с u и v_j совместимо с u, по лемме 3 следует совместимость v_i с v_j. Значит, мы решили исходную задачу. Таким образом, за полиномиальное время задача о поиске максимальной клики графа (которая, как известно, NP-трудна, а как задача распознавания NP-полна) сведена к задаче 1 над бинарным алфавитом. Теорема доказана. □

4. Задача минимизации расстояния Хэмминга

Определим $t(s,p)$ — *непохожесть* текста s длины n и шаблона p длины m — следующим образом. Возьмем все подстроки длины m в тексте s. Найдем расстояние Хэмминга от каждой такой подстроки до шаблона p. Их сумма и будет непохожестью шаблона и текста.

Напомним задачу 2: даны две поврежденных строки: текст s и шаблон p. Требуется найти неповрежденные строки v и u, совместимые с s и p, соответственно, и такие, что $t(v,u)$ минимальна.

Сопоставим паре (s,p) двудольный граф $G_{s,p}$ следующим образом: в первой доле пусть будет n вершин, во второй m. Будем соединять ребром вершину i первой доли и j второй доли тогда и только тогда, когда соответствующие им позиции в тексте и шаблоне накладываются друг на друга при совмещении шаблона p и некоторой подстроки текста s. Переформулируем задачу 2 в терминах графа $G_{s,p}$. Сопоставим каждому символу a поврежденного алфавита цвет. Покрасим вершину k первой доли в этот цвет, если $s[k]=a$, аналогично для вершин второй доли; полученную раскраску назовем *исходной*. Назовем раскраску *полной*, если все вершины покрашены в цвета, сопоставленные неповрежденным символам. Назовем две раскраски *совместимыми*, если для каждой вершины графа цвета, в которые она покрашена в этих раскрасках, соответствуют совместимым символам. Тогда задача 2 формулируется следующим образом: *из всех полных раскрасок графа $G_{s,p}$, совместимых с исходной раскраской, найти такую, в которой количество ребер, инцидентных вершинам разных цветов, минимально.*

4.1. Случай неповрежденного текста.

Предложение 3. *Существует алгоритм, решающий задачу 2 для неповрежденного текста за время $O(n+|\Sigma|m)$.*

Доказательство. Для поврежденного символа в i-й позиции шаблона лучший вариант замены — тот из совместимых символов, который встречается чаще всего в строке $s[i...n-m+i]$. Заведем массив длины $|\Sigma|$ для счетчиков неповрежденных символов. При обработке одного символа из s обновляется один счетчик (для последних $m-1$ символов — два) и массив частых символов (одна операция на элемент массива). Получаем требуемую оценку. В ходе последних m шагов (на каждом за время $O(|\Sigma|)$) устанавливаем значения для всех поврежденных символов из p. □

Замечание 2. *Для случая частичных строк значение поврежденного символа каждый раз выбирается среди всех символов алфавита. Поэтому мы можем хранить количества вхождений символа не в массиве, а в сбалансированном дереве или бинарной куче(пирамиде). Тогда изменение и извлечение максимума и добавление/удаление элемента будет выполняться за время $O(\log|\Sigma|)$. Общее время работы алгоритма в этом случае $O(m\log|\Sigma|+n)$.*

4.2. Случай неповрежденного шаблона.

Предложение 4. *Существует алгоритм, решающий задачу 2 для неповрежденного шаблона за время $O(|\Sigma|n)$.*

Доказательство. Для поврежденного символа в i-й позиции строки лучший вариант замены — тот из совместимых символов, который встречается чаще

всего в строке $p[max(1, i-n+m)...min(m, i)]$. Заведем массив длины $|\Sigma|$ для счетчиков неповрежденных символов. При обработке одного символа из p обновляется один счетчик (для символов с m по $n-m+1$ — ни одного) и массив частых символов (одна операция на элемент массива). Получаем требуемую оценку. В ходе процесса устанавливаем значения для всех поврежденных символов из s. □

Замечание 3. *Для частичных строк аналогично случаю неповрежденного текста можно решить задачу за общее время* $O(m \log |\Sigma| + n)$.

4.3. **Случай частичных строк с циклическим текстом.** *Циклической строкой* называется последовательность букв, упорядоченная не линейно, а циклически. Тем самым, циклическая строка получается из обычного "склеиванием" концов. Подстрока циклической строки — это обычная строка, отличие только в том, что циклическая строка $u[1...n]$ имеет, для любых $i < j \leq n$, не только подстроку $u[i...j]$, но и подстроку $u[j...i] = u[j...n]u[1...i]$. Пусть текст s — циклический, т.е. является циклической строкой.

Предложение 5. *Существует алгоритм, решающий задачу 2 для случая, когда шаблон — частичная строка, а текст — частичная циклическая строка, за время* $O(n)$.

Доказательство. Заметим, что в данном случае граф $G_{s,p}$ будет полный. А значит, все списки смежности вершин первой доли совпадают. Тогда существует оптимальное решение, в котором все вершины первой доли, сопоставленные поврежденным символам, т.е. джокерам, покрашены в один цвет; то же самое верно и для второй доли.

Вычислим для каждого $a \in \Sigma \cup \Gamma$ количества $S(a)$ и $P(a)$ вхождений a в текст и в шаблон соответственно; требуемое время — $O(n + m + |\Sigma \cup \Gamma|) = O(n + m + |\Sigma|) = O(n)$.

Далее, за $O(|\Sigma|)$ вычислим символы a_1, a_2 и a_3, на которых достигается максимум $S(a)$, $P(a)$ и $S(\diamond)P(a) + P(\diamond)S(a)$ соответственно. Если присвоить джокерам в тексте и шаблоне одно и то же значение, то максимальное количество полученных за счет этого совпадений символов равно $S(\diamond)P(a_3) + P(\diamond)S(a_3) + S(\diamond)P(\diamond)$. Если же двум указанным группам джокеров присвоены разные значения, то максимальное количество полученных совпадений есть $S(\diamond)P(a_2) + P(\diamond)S(a_1)$. Сравнив два полученных количества, выберем оптимальный вариант присвоения. □

4.4. **Случай бинарного алфавита.**

Предложение 6. *Для случая бинарного алфавита существует полиномиальное сведение задачи 2 к задаче о разрезе.*

Доказательство. Как уже упоминалось в разделе 3.3, поврежденный символ в случае бинарного алфавита является джокером.

Пусть в графе $G_{s,p}$ существует путь из вершины первого цвета в вершину второго цвета. Тогда легко видеть, что хотя бы одно ребро каждого такого пути будет в итоге (т.е. после раскраски всех джокеров в первый и второй цвет) иметь разноцветные концы. Пусть мы удалили некоторое количество ребер так, чтобы не существовало пути между двумя разнопокрашенными вершинами. Тогда в каждой компоненте связности содержатся вершины только

какого-то одного цвета и "прозрачные" (цвет джокера). Очевидно, что после этого нужно покрасить прозрачные вершины в цвет, который присутствует в ее компоненте связности (а если такого нет, то в любой). Таким образом, все ребра графа будут иметь одинаково покрашенные концы. Значит, наша задача свелась к следующей: *удалить из графа минимальное количество ребер, чтоб не существовало путей между некоторыми двумя вершинами разных цветов.*

Это — в точности задача о минимальном разрезе между двумя множествами вершин. Решим ее следующим образом. Объединим все вершины первого цвета в одну и назовем ее стоком (ребра, которые были смежны с вершинами первого цвета, будут смежны со стоком). Аналогично все вершины второго цвета объединим в исток. Присвоим всем ребрам полученного графа единичный вес. Найдем минимальный разрез между вершинами сток и исток. Для получения ответа к исходной задаче, прозрачные вершины, из которых достижим сток после удаления ребер разреза, раскрашиваются в первый цвет, а вершины, достижимые из источника — во второй.

Таким образом мы показали, что для рассматриваемой задачи существует полиномиальный алгоритм, например, ее можно решить алгоритмом Форда-Фалкерсона за $O(m^2n)$. □

4.5. **Случай частичных строк и задача о мультиразрезе.** Аналогично предыдущему разделу, мы можем свести задачу 2 для случая частичных строк над произвольным алфавитом к нахождению минимального мультиразреза в графе. Пусть в графе $G_{s,p}$ существует простая цепь, содержащая не менее двух вершин, где концевые вершины покрашены в разные цвета, а все промежуточные вершины — прозрачные. В нашей задаче необходимо покрасить все вершины так, чтобы максимизировать количество ребер с одноцветными концами. В каждой такой цепи есть ребро с разноцветными концами. Значит, нужно удалить минимальное количество ребер из графа так, чтоб не существовало пути между двумя покрашенными вершинами разных цветов.

Объединим все вершины одного цвета в одну: вместо всех вершин данного цвета (удалим их) добавим одну новую, соединив ее ребрами с теми вершинами, с которыми была соединена хотя бы одна из исходных. После этого у нас в графе останутся покрашенные и прозрачные вершины, причем нет двух вершин одного цвета. Теперь в данном графе необходимо удалить минимальное количество ребер, чтобы не существовало путей между двумя покрашенными вершинами. Эта задача и называется задачей о минимальном мультиразрезе. К сожалению, она является NP-трудной (в варианте распознавания — с вопросом "существует ли мультиразрез из k ребер?" — NP-полной).

Гипотеза 1. *Задача восстановления текста и шаблона с минимизацией непохожести строки и шаблона для случая частичных строк NP-трудна.*

Замечание 4. *Для задачи о мультиразрезе существует приближенный полиномиальный алгоритм, находящий мультиразрез, мощность которого менее чем вдвое превышает мощность минимального мультиразреза* [9]. *Следовательно, приближенный алгоритм с такими же характеристиками есть и для задачи 2.*

4.6. **Доказательство NP-трудности в общем случае.**

Теорема 2. *Задача 2 в общем случае и в случае циклического текста NP-трудна.*

Доказательство. Начнем с циклического текста. Напомним, что в данном случае граф $G_{s,p}$ — полный. Значит, порядок следования символов в строках не влияет на решение, что дает возможность рассматривать строки как мультимножества символов. В отличие от случая частичных строк, нам для каждой вершины графа дополнительно дано множество допустимых для нее цветов (т.е. совместимых с данным символом символов из Σ). Пару, состоящую из восстановленного текста S' и восстановленного шаблона P' мы отождествляем с мультимножеством пар $(s'[i], p'[j])$, имеющим мощность mn. Процесс восстановления состоит в выборе допустимого цвета для каждой вершины, а «показателем качества» восстановления является число пар вида (a, a).

Назовем Задачу 2 дизъюнктной, если для любой пары натуральных $1 \leq i < j \leq n$ символы $s[i]$ и $s[j]$ несовместимы. Покажем, что даже дизъюнктная задача 2 для циклического текста NP-трудна.

Замечание 5. *Если наложить условие дизъюнктности и на шаблон, то задача очевидно решится паросочетанием за кубическое от размера алфавита (который в данном случае асимптотически равен длине обоих строк).*

Из отсутствия совместимых символов в тексте следует, что каждый символ шаблона (после восстановления) может образовать пару равных элементов максимум с одним символом текста (символ, образующий такую пару, назовем *удовлетворенным*). В частности, показатель качества решения задачи теперь не превосходит длины шаблона.

Рассмотрим задачу в еще более частном случае: каждый символ текста поврежден и совместим ровно с двумя символами основного алфавита. Обозначим эти символы для $s[i]$ за x_i и $\bar{x}_i$. Таким образом, восстановление $s[i]$ состоит в присвоении значения булевой переменной. Каждому символу шаблона сопоставим дизъюнкцию литералов, соответствующих допустимым цветам для этого символа. Эта дизъюнкция истинна тогда и только тогда, когда символ шаблона удовлетворен. Тем самым, задача максимизации числа удовлетворенных символов шаблона — это в точности задача $maxSAT$ с n переменными и m элементарными дизъюнкциями (клозами). Из NP-трудности $maxSAT$ вытекает NP-трудность нашей задачи. Ниже мы даем точную формулировку задачи $maxSAT$ и строгое сведение ее к частному случаю Задачи 2.

Задача maxSAT (или задача об оптимальной выполнимости) формулируется следующим образом. Дано m выражений(клозов), каждое из них — дизъюнкция нескольких литералов (переменная, либо ее отрицание), всего в них участвует не более n булевых переменных. Требуется выбрать значения переменных таким образом, чтоб максимизировать количество истинных клозов.

Пусть дан пример задачи $maxSAT$ с переменными $x_1, \ldots, x_n$ и клозами $C_1, \ldots, C_m$. Построим по ней пример Задачи 2 для циклического текста. Положим $\Sigma = \{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n\}$, $\Gamma = \{b_1, \ldots, b_n, c_1, \ldots, c_m\}$, причем каждый поврежденный символ b_i совместим в точности с литералами x_i и $\bar{x}_i$, а каждый поврежденный символ c_i — с литералами, входящими в клоз C_i. Далее, $s = (b_1 \ldots b_n)$ (циклическая строка), $p = c_1 \ldots c_m$.

Пусть теперь мы решили построенный пример Задачи 2. Тогда восстановленный текст дает решение для $maxSAT$ (если $b_i = x_i$, присваиваем $x_i = 1$,

а если $b_i = \bar{x}_i$, то $x_i = 0$), а показатель качества восстановления равен числу истинных клозов. Задача в случае циклического текста NP-трудна.

Для доказательства теоремы осталось свести задачу о циклической строке к задаче в общем виде. Этого можно добиться, расширив основной алфавит на один символ (несовместимый ни с одним поврежденным) и приписав m таких символов слева и справа к тексту. Очевидно, после этого каждый символ текста совместится с каждым символом шаблона, при этом совмещение с новым символом никак не повлияет на ответ. Теорема доказана. □

Замечание 6. *В класс сложности APX входят все задачи оптимизации, для которых существует полиномиальный алгоритм, приближенно решающий данную задачу с ограниченной относительной погрешностью приближения оптимизируемого критерия (В случае задачи максимизации: существует константа c, $0 < c < 1$ такая, что алгоритм находит решение со значением критерия не меньше cM, если оптимальное значение равно M). Задача $maxSAT$, эквивалентная рассмотренному выше специальному случаю задачи 2, является APX-полной. Следовательно, задача 2 является APX-трудной, а в случае частичных строк, с учетом замечания 4 принадлежит классу APX.*

5. Итоги

В данной работе рассмотрены две задачи о восстановлении повреждений в тексте и шаблоне. Для задачи о максимизации количества вхождений шаблона в текст были найдены эффективные алгоритмы, решающие частные случаи этой задачи. Для случая, когда шаблон не поврежден, задача решена методом динамического программирования за время $O(nm)$. Для неповрежденного текста задачу удалось решить с помощью алгоритма Фишера-Патерсона за время $O(n \log n \log |\Sigma|)$ с предвычислениями за $O(n)$ с помощью построения суффиксного массива текста. Если же повреждены и текст, и шаблон, то задача оказалась NP-трудной даже для бинарного алфавита, что было доказано путем сведения к ней задачи о максимальной клике в графе.

Задача о минимизации непохожести шаблона и текста была решена в ряде частных случаев. Когда текст или шаблон — неповрежденные строки, задача решается простым жадным алгоритмом. Несколько более сложная идея используется при решении задачи для частичных строк, когда текст является циклическим. Для бинарного алфавита задача решена с помощью алгоритма нахождения минимального разреза в графе. В общем виде доказана NP-трудность задачи путем сведения к ней задачи $maxSAT$. Для случая частичных строк задача не изучена до конца, выдвинута гипотеза о ее NP-трудности.

Дальнейшее исследование предполагает продолжение изучения задачи о минимизации непохожести шаблона и текста для частичных строк, а также выделение дополнительных частных случаев обеих задач с целью выявления более простых подходов к решению.

Список литературы

[1] M. Fischer, M. Paterson, *String matching and other products* // SIAM-AMS Proceedings, **7** (1974), 113–125. MR0400782

[2] D. Gusfield, *Algorithm on Strings, Trees, and Sequences*. Cambridge University press, 1997. MR1460730

[3] S. Muthukrishnan, H. Ramesh, *String matching under a general matching relation*. In Proc. 12th Conference on Foundations of Software Technology and Theoretical Computer Science, **652**. Berlin, Springer-Verlag, 1992. 356–367. MR1229111

[4] Д. Кнут, *Искусство программирования, том 2. Получисленные алгоритмы*. М., Вильямс, 2007.

[5] S. Muthukrishnan, K. Palem, *Non-standard stringology: algorithms and complexity*. In *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC'94)*. New York, ACM, 1994. 770–779.

[6] Ge Nong, Sen Zhang, Wai Hong Chan, *Linear Time Suffix Array Construction Using D-Critical Substrings* // Lecture Notes in Computer Science, **5577** (2009), 54–67. Zbl 1247.68071

[7] D.E. Knuth, J.H. Morris (Jr.), V.R. Pratt, *Fast Pattern Matching in Strings* // SIAM J. Comput., **6**:2 (1977), 323–350. MR0451916

[8] T. Kärki, T. Harju, V. Halava, *Interaction Properties of Relational Periods* // Discrete Mathematics & Theoretical Computer Science, **10**:1 (2008), 87–111. MR2393230

[9] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, M. Yannakakis, *The Complexity of Multiterminal Cuts* // SIAM J. Comput., **23** (1994), 864–894. MR1283579

Михаил Валентинович Рубинчик
Институт математики и компьютерных наук,
Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина,
620000, Екатеринбург-83, пр. Ленина, 51
E-mail address: mikhail.rubinchik@gmail.com

Юлия Васильевна Гамзова
Институт математики и компьютерных наук,
Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина,
620000, Екатеринбург-83, пр. Ленина, 51
E-mail address: julia.gamzova@gmail.com