Факультет компьютерных технологий и прикладной математики

Кафедра вычислительных технологий

02.03.02

Информационная безопасность

Лабораторная работа № 1

Тема: Проектирование алгоритмов поддержки информационной безопасности.

Цель работы: научиться проектировать алгоритмы поддержки информационной безопасности.

Задание: необходимо разработать алгоритмы и программы решения двух задач на языке C++.

Каждый студент должен выполнить задачи по вариантам, изложенным ниже. За проделанную работу студент может получить оценку от «неудовлетворительно» до «отлично». На занятии студенты решают первую задачу по вариантам, изложенным ниже. Студент разрабатывает алгоритм и программу решения задачи на языке C++ на лабораторном занятии. За работу на занятии студент может получить оценку от «неудовлетворительно» до «хорошо». При построении алгоритма первой задачи на занятии студент получает оценку «удовлетворительно» или «хорошо». Для получения оценки «хорошо» на занятии студент должен выполнить первую задачу и ответить на теоретические вопросы. Оценка «удовлетворительно» ставиться, если студент на занятии решает первую задачу полностью, но не может ответить на теоретические вопросы или выполняет как минимум половину первой задачи и в состоянии ответить на теоретические вопросы. Оценку «неудовлетворительно» студент получает, если выполняет минимум половину первой задачи или в состоянии ответить на теоретические вопросы.

Оценка «отлично» может быть выставлена ТОЛЬКО если студент получил оценку «хорошо» на занятии и сделал вторую задачу на следующее занятие.

Если студент не получил оценку, то ему будет необходимо подготовить отчёт по задаче и реализовать программу, используя систему управления версиями Git и размещая её репозиторий на GitHub.

При защите отчёта ОБЯЗАТЕЛЬНО ответить на теоретические вопросы по прошедшим лекциям.

ОЦЕНКУ ЗА ПРОДЕЛАННУЮ РАБОТУ МОЖНО ПОВЫСИТЬ ДО СДАЧИ СЛЕДУЮЩЕЙ ЛАБОРАТОРНОЙ РАБОТЫ.

Вариант: ((N - 1) % 20) + 1, где % – деление с остатком, N – номер в подгруппе.

Вариант 1. Задачи 1, 6

Вариант 2. Задачи 7, 11

Вариант 3. Задачи 13, 16

Вариант 4. Задачи 19, 21

Вариант 5. Задачи 1, 5

Вариант 6. Задачи 7, 10

Вариант 7. Задачи 13, 15

Вариант 8. Задачи 19, 20

Вариант 9. Задачи 1, 4

Вариант 10. Задачи 7, 9

Вариант 11. Задачи 13, 14

Вариант 12. Задачи 1, 3

Вариант 13. Задачи 7, 8

Вариант 14. Задачи 1, 2

Вариант 15. Задачи 7, 12

Вариант 16. Задачи 13, 17

Вариант 17. Задачи 19, 22

Вариант 18. Задачи 13, 18

Вариант 19. Задачи 19, 23

Вариант 20. Задачи 19, 24

1. Шифровка. Штирлиц хочет передать очень важное сообщение s в штаб. Для этого он использует беспрефиксный код, зафиксированный по ГОСТ. К сожалению, противнику известен код, а канал связи Штирлица прослушивается. Чтобы отвести подозрения, Штирлиц хочет разбить шифровку на куски, каждый из которых нельзя расшифровать тем же кодом. Для максимальной безопасности Штирлиц хочет, чтобы количество кусков было максимально. Найдите максимальное количество кусков или определите, что разбить шифровку требуемым образом невозможно.

Исходные данные

В первой строке записано одно целое число k $(1 \leq k \leq 52)$ – количество символов в алфавите. Символы пронумерованы от 1 до 52 в порядке A-Za-z, в тексте сообщения будут использованы только символы с номерами от 1 до k.

Во второй строке записана строка s длины до 10^6, состоящая из символов с номерами от 1 до k (нумерация символов определена выше).

В следующих k строках записаны двоичные коды символов алфавита согласно нумерации. Каждый символ алфавита кодируется последовательностью 0 и 1 длины не более k. Гарантируется, что никакой код не является префиксом другого кода.

Результат

Выведите одно число – максимальное количество кусков, либо -1, если разбить на куски указанным способом невозможно.

В таблице 1 приведены примеры.

Таблица 1 – Примеры

исходные данные	результат
3 CACB 011 1 001	2
3 ACBABCAACABCAACC 0 10 110	-1

Замечания

В первом семпле зашифрованный текст выглядит как 0010110011. Единственный способ разбить его на куски, которые нельзя расшифровать – 0 010110011. Следовательно, ответ равен 2.

Во втором семпле можно показать по индукции, что любая строчка, которая заканчивается на 0 и не содержит трёх 1 подряд, расшифровывается данным кодом. Любой суффикс зашифрованного текста имеет такой вид, поэтому ответ – -1.

2. Шифрограмма. Слово зашифровано «циклическим сдвигом на 3 символа назад». Например, слово СЕЛО, шифр – ОСЕЛ, слово КООРДИНАТА, шифр – РДИНАТАКОО. Написать функцию decode, которая по заданному шифру восстанавливает зашифрованное слово. Задан зашифрованный как указано выше текст, например, в виде текстового файла. Слова в тексте разделяются одним или несколькими пробелами. Написать программу, использующую функцию decode, расшифровки этого текста, например, в результирующий текстовый файл.

3. Система оповещения. Руководитель организации решил создать систему оповещения своих сотрудников на случай опасности. Для этого он разбил всех сотрудников на несколько групп следующим образом. В первую группу вошли сотрудники, номера телефонов которых он знал сам. Во вторую группу были включены сотрудники, номера которых в целом знали сотрудники первой группы, и т.д. В результате этого получилась схема, изображенная на рисунке, где представлены три группы сотрудников. Цифра 1 соответствует руководителю. Стрелки определяют знание i-ым сотрудником номера телефона j-го сотрудника.

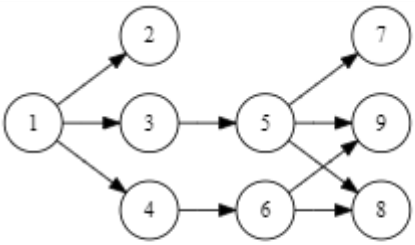


Рисунок 1 – Схема

Система оповещения должна функционировать следующим образом. В случае экстраординарной ситуации руководитель должен последовательно позвонить всем сотрудникам первой группы. Сотрудник, получивший сообщение, должен позвонить по известным ему телефонам сотрудникам другой группы и т.д. В результате этого каждый сотрудник должен быть проинформирован по телефону об экстраординарной ситуации, причем ему могли звонить только один раз. Во время передачи сообщения длительность телефонного разговора у всех сотрудников одинакова и равна 1 минуте.

В зависимости от того, кто кому звонит, общее время оповещения всех сотрудников будет разным. Например, если для изображенной на рисунке 1 схеме будет использоваться последовательность звонков, представленная в таблице 2, то время будет равно 4 минутам, а если в таблице 3, то – 5 минутам.

Руководителю нужно знать такую последовательность звонков, которая обеспечит полное оповещение за минимальное время.

Таблица 2 – Последовательность звонков

Источник	Приемник	Время
1	4	1
1	3	2
1	2	3
3	5	3
4	6	2
5	7	4
6	9	3
6	8	4

Таблица 3 – Последовательность звонков

Источник	Приемник	Время
1	2	1
1	3	2
1	4	3
3	5	3
4	6	4
5	7	4
5	8	5
6	9	5

Написать программу, которая по заданной схеме взаимодействия определяет ту последовательность звонков, которая обеспечивает оповещение всех сотрудников за минимальное время. Схема взаимодействия задается во входном текстовом файле input.txt.

Формат файла input.txt:

В первой строке задано число N ($N \leq 20$) – количество сотрудников.

Последующие N строк содержат описание схемы взаимодействия сотрудников. При этом каждая i-ая строка содержит номер i сотрудника и номера сотрудников, телефоны которых известны i-му сотруднику. Все числа в строке разделены пробелом. Например,

9
1 2 3 4
2
3 5
4 6
5 7 8 9
6 8 9
7
8
9

Результаты формируются в текстовом файле output.txt. В первой строке должно быть одно число – общее время оповещения всех сотрудников в минутах. В каждой из последующих строк должны содержаться пары чисел (номер источника и номер получателя), соответствующие передачи сообщения по телефону в один и тот же период времени. Все числа разделены пробелом.

Например,

4
1 4
1 3 4 6
1 2 3 5 6 9
5 7 6 8

4. Зазеркалье. Написать программу, которая по заданному входному текстовому файлу input.txt формирует результирующий текстовый файл output.txt. Содержимое входного файла – последовательность, разделенных

одним или несколькими пробелами зеркально отраженных слов. Требуется в той же последовательности вывести в output.txt результаты зеркального отображения этих слов. Например, файл input.txt: отЭ ремипр оготсорп атсет. илсЕ ыВ еще ен иляноп, от етишипаз ывкуб огоджак аволс в монтарбо екдяроп.

Файл output.txt: Это пример простого теста. Если Вы еще не поняли, то запишите буквы каждого слова в обратном порядке.

5. Взлом игры. Мальчик Петя очень хочет поиграть в новую компьютерную игру, но при ее запуске игра задает 10 вопросов, на которые нужно ответить «да» (t) или «нет» (f). Если на все вопросы Петя ответил правильно, то игра запускается, а если хотя бы на один вопрос был дан неправильный ответ, то игра не запускается. После каждой неудачной попытки игра сообщает, на сколько вопросов было отвечено неправильно. Петя уже сделал N ($1 \leq N \leq 100$) неудачных попыток. Все свои попытки он записывал. Требуется написать программу, которая по уже известным попыткам определяет ответы на вопросы, а если этого однозначно сделать нельзя, то выводит один из возможных вариантов ответа. Во входном текстовом файле input.txt в первой строчке находится число N. Далее идет N строк длиной 11 символов. Из них первые 10 символов строки – это символы «t» или «f» – информация о том, как отвечал Петя на поставленные вопросы: «t» – если Петя отвечал «да», «f» – если Петя отвечал «нет» на соответствующий вопрос, а 11-ый символ – ответ игры, число (вернее цифра) от 0 до 9 – количество правильных ответов в этой попытке. В выходном текстовом файле output.txt в первой строке должна находиться фраза ONE SOLUTION, если можно однозначно определить правильные ответы, и фраза POSSIBLE SOLUTION, если однозначно определить правильные ответы нельзя. Вторая строка должна содержать 10 символов «f» или «t» (подряд, без пробелов). При первом варианте решения вторая строка должна содержать последовательность ответов, при которой игра запускается. При втором

варианте, вторая строка должна содержать последовательность ответов, при которой игра может запуститься.

Пример. Файл input.txt:

3

ttttttttt5

fffffffff5

ttttffffff0

output.txt:

ONE SOLUTION

fffffttttt

6. Шифр. Способ шифрования Цезаря состоит в следующем: буквы латинского алфавита нумеруются по порядку числами от 1 до 26 (A – 1, B – 2, C – 3, D – 4, E – 5, F – 6, G – 7, …, X – 24, Y – 25, Z – 26). Затем в шифруемом сообщении каждая буква с номером n в алфавите заменяется на $(n + k)$-ую букву алфавита (Цезарь использовал $k = 3$). Пробел считается 0-вой буквой алфавита, а сложение выполняется по модулю 27. Пусть k неизвестно. В этом случае для расшифровки такого сообщения необходимо перебрать 26 вариантов для различных значений k. Количество вариантов сокращается при наличии списка слов, которые могут встретиться в расшифрованном сообщении. Составить программу, которая расшифровывает сообщение, используя данный список слов. Программа считывает текстовый файл input.txt следующего формата: в первой строке находится целое число L $(1 \leq L \leq 100)$ – количество слов в списке. В следующих L строках – список слов, по одному слову в каждой строке (в каждом слове не более 20 символов, как известно Цезарь был краток). В следующей строке – зашифрованное сообщение из не более чем 200 символов. Список слов не обязательно содержит все слова из сообщения и наоборот, сообщение не обязательно содержит все слова из списка. Программа должна вывести текстовый файл output.txt следующего формата: расшифрованное сообщение, содержащее

максимальное количество слов из входного списка (с учетом их повторений в зашифрованном сообщении). Если решений несколько, то вывести одно из них. Зашифрованное и расшифрованное сообщения состоят только из заглавных латинских букв и пробелов. Пример. Файл input.txt:

8
THIS
DAWN
THAT
THE
ZORRO
OTHER
AT
THING
BUUBDLA PSSPABUAEBXO

Файл output.txt:

ATTACK ZORRO AT DAWN

7. RLE. Компрессия RLE – это способ сжатия данных, в которых имеется много подряд идущих символов. При этом более одного вхождения символа заменяется на пару (количество вхождений - символ). В входном файле input.txt задан текст, состоящий из букв латинского алфавита. Примените к нему с помощью написанной программы преобразование RLE, т.е. замените N ($N > 1$) вхождений некоторого символа X, которые подряд идут в строке на NX. Длина входной строки не более 10000 символов. Преобразованный текст программа должна поместить в текстовый файл output.txt. Пример:

Таблица 4 – Пример

input.txt	output.txt
BCCAAAAAAAAAAABBBD	B2C11A3BD

8. Кодовый замок. Есть кодовый замок. На нём N переключателей ($1 \leq N \leq 26$), промаркированных различными буквами. Каждый переключатель может быть или в положении ON, или в положении OFF. При установке замка владелец ставит в соответствие каждой из N букв некоторый код – натуральное число. Различные буквы имеют различные коды $C_1, C_2, \ldots, C_N$ ($1 \leq C_j \leq 64000, 1 \leq j \leq N$). Для того, чтобы открыть замок, владелец устанавливает некоторые переключатели в положение ON, а оставшиеся – в положение OFF. Замок открывается, когда сумма всех чисел-кодов, соответствующих выключателям, находящимся в положении ON, делится без остатка на N, т.е. существует некоторое подмножество кодов $C_{j_1}, C_{j_2}, \ldots, C_{j_k}$ ($k \leq N$), сумма всех членов которого делится на N. Взломщик не знает значений кодов и пытается открыть замок, перебирая некоторые комбинации букв и устанавливая соответствующие переключатели в положение ON. Требуется составить программу, которая должна определить, возможно ли открыть замок, т.е. существует ли такое подмножество букв, что сумма кодов, соответствующих этим буквам, делится без остатка на N (возможные ответы программы: YES или NO). В случае, если это возможно, программа должна выдать какое-то подмножество кодов $C_{j_1}, C_{j_2}, \ldots, C_{j_k}$ и P – количество всех возможных комбинаций кодового замка, которые его открывают. Входные данные задаются в текстовом файле input.txt, а результаты – в текстовом файле output.txt. Формат входного файла input.txt:

N

$C_1\ C_2\ \ldots, C_N$

Формат выходного файла output.txt:

YES/NO

$C_{j_1}\ C_{j_2}\ \ldots\ C_{j_k}$

P

Например, input.txt:

4

15 2 6 19

Тогда output.txt:

YES

2 6

16

9. Фокусник. Старый фокусник решил передать секрет одного фокуса своему ученику. Фокус заключается в следующем: зритель угадывает натуральное число от 100 до 10^{200}, затем вычитает из числа сумму его цифр. В полученном числе он зачёркивает любую ненулевую цифру и называет все оставшиеся цифры фокуснику. Выслушав названные цифры, фокусник, погодя, называет цифру, которая была зачёркнута. Ученик не силён в вычислениях. Поэтому требуется написать программу, которая, получив на вход названные цифры, выдаёт зачёркнутую зрителем цифру. Вход программы представлен текстовым файлом input.txt, в котором в одну строку без пробелов заданы называемые зрителем цифры. Выход программы представлен текстовым файлом output.txt, в который записывается зачёркнутая цифра. Пример:

Таблица 5 – Пример

input.txt	output.txt
12345674	4

10. Тайский шифр. Когда-то король королевства Хай-Тай решил придумать шифр для кодирования секретных посланий своему послу в далёкой стране. Алфавит в Хай-Тае состоит из большого числа иероглифов. Все они нумеруются числами от 1 до N ($1 \leq N \leq 30000$). Исходный текст (набор иероглифов без пробелов или знаков пунктуации) преобразуется в последовательность чисел a_1, a_2, …, a_m ($1 \leq m \leq 30000$) путём замены каждого иероглифа его номером. Кодировщик шифрует эту

последовательность умножением каждого номера на константу D с последующим увеличением на константу C ($0 \le C \le 30000$, $0 \le D \le 30000$). Если результат i больше, чем N, то он (т.е. i) должен быть заменён i-ым номером в последовательности 1, 2, …, N, 1, 2, …, N, 1, 2, …, N, …. Например, результат $i = 35$ заменяется на 5, если $N = 15$. Поэтому результаты шифруются последовательностью чисел $b_1, b_2, \ldots, b_m$ из диапазона $[1; N]$. Заменив число b_k иероглифом с таким номером в алфавите, кодировщик получит зашифрованный текст. Посол, получив зашифрованный текст, хочет расшифровать сообщение. Во избежание международных осложнений требуется написать программу, которая должна определить, может ли посол однозначно расшифровать сообщение. Программа должна выдать вердикт YES или NO. Если программа отвечает YES (т.е. сообщение однозначно расшифровывается), то она выдаёт и расшифрованное сообщение $a_1, a_2, \ldots, a_m$. Входные данные задаются в текстовом файле input.txt. В первой строке этого файла задаются натуральные числа N и m, разделённые одним или несколькими пробелами. Во второй строке задаются целые числа D и C, разделённые одним или несколькими пробелами. В третьей строке записаны целые b_1, b_2, …, b_m, разделённые пробелами. Результат помещается в текстовый файл output.txt. В первой его строке пишется слово YES или NO. В случае YES вторая строка содержит расшифрованную последовательность $a_1, a_2, \ldots, a_m$ чисел, разделённых пробелами. Пример:

Таблица 6 – Пример

input.txt	output.txt
13 7 7 3 5 4 9 2 7 8 1	YES 4 2 12 11 8 10 9

11. Super RLE. Можно пойти и дальше (см. задачу № 0), попытавшись написать программу, реализующую преобразование Super RLE. Для этого

повторяющийся N раз фрагмент F исходного текста заменить на $N(F)$. Например, текст AAAA заменяется на 4(A), текст ABCBCD заменяется на A2(BC)D, а текст AAABBAAABBCCCCD заменяется на 2(3(A)2(B))4(C)D. Сжатый в такой форме текст выводится в файл output.txt. Если возможно несколько вариантов свёртки, то в output.txt помещается свёртка по возможности наименьшей длины.

12. Шпионский сленг. Секретные сообщения между агентами кодируются 25-языком. В этом языке используется 25-алфавит, который совпадает с латинским алфавитом, но в нём отсутствует буква Z, т.е. 25-алфавит содержит 25 латинских букв от A до Y в том же порядке, что и латинский алфавит. Каждое слово такого языка состоит ровно из 25 различных букв. Слово записывается в таблице размера 5×5 строка за строкой. Например, слово ADJPTBEKQUCGLRVFINSWHMOXY будет записано так:

A D J P T
B E K Q U
C G L R V
F I N S W
H M O X Y

Правильным словом 25-языка считается такое слово, буквы в котором (если записать его в виде 5×5-таблицы) расположены по возрастанию их номера в алфавите в каждой строке и в каждом столбце. Поэтому слово ADJPTBEKQUCGLRVFINSWHMOXY является правильным, а слово ADJPTBEGQUCKLRVFINSWHMOXY – нет (возрастающий порядок нарушается во втором и третьем столбцах). Лексикон агента состоит из всех правильных слов 25-языка. Слова расположены в возрастающем лексикографическом порядке (как это принято в словарях) и пронумерованы, начиная с 1. Например, слово ABCDEFGHIJKLMNOPQRSTUVWXY имеет номер 1, а слово ABCDEFGHIJKLMNOPQRSUTVWXY имеет номер 2. Во втором слове, по сравнению с первым, буквы U и T поменялись местами. Для

упрощения нелёгкой жизни агента требуется написать программу, которая по заданному порядковому номеру слова определяет само это слово. В лексиконе агента не более 2^{31} слов. Программа на входе получает порядковый номер некоторого правильного слова 25-языка. Единственная строка на выходе – это слово с заданным порядковым номером. Например, на входе 2, на выходе ABCDEFGHIJKLMNOPQRSUTVWXY.

13. Шифровка. Мюллер много раз пытался поймать Штирлица с поличным, но тот всё время выкручивался. Как-то раз Штирлиц просматривал электронную почту. В это время незаметно вошел Мюллер и увидел, как у него на экране появился бессмысленный набор символов. «Шифровка», – подумал Мюллер. «UTF-8», – подумал Штирлиц.

Известно, что Штирлиц шифрует текст следующим образом:

Убирает все пробелы и знаки препинания.

Заменяет все подряд идущие одинаковые буквы на одну такую букву.

Многократно вставляет в произвольное место текста две одинаковых буквы.

Попробуйте восстановить текст, каким он был после второго шага. Для этого удалите из текста все пары одинаковых символов, добавленные на третьем шаге.

Исходные данные

В единственной строке записана шифровка Штирлица, состоящая из строчных латинских букв. Длина шифровки не превосходит 200000.

Результат

Выведите восстановленный текст.

В таблице 1 приведен пример файлов входных и выходных данных.

Таблица 7 – Пример файлов входных и выходных данных

исходные данные	результат
wwstdaadierfflitzzz	stierlitz

14. Бинарный код Грэя. Как известно любое подмножество S из n элементов $a_1, a_2, \dots, a_n$ можно представить в виде в виде n-разрядного бинарного числа $d_1 d_2 \dots d_n$, где $d_i = 1$, если a_i принадлежит S т.е. $a_i \in S$ и $d_i = 0$, если a_i не принадлежит S, т.е. $a_i \in S$. Обычный способ построения всех подмножеств из n элементов – начинать с n-разрядного числа 00…0 и последовательно прибавлять 1 к очередному полученному подомным способом n-разрядному числу. Например, для $n = 4$ начиная с 0000 генерируем 4-разрядные числа 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000. Здесь соседние числа могут отличаться более чем в одной позиции (например, два последних числа). В некоторых приложениях это может быть неудобным или даже неприемлемым. Для таких применений более удобным может быть код Грэя. Метод построения бинарного кода Грэя использует следующее наблюдение. Если последовательность $C_1, C_2, \dots, C_m$ содержит все n-разрядные коды, $m = 2^n$, причем каждый следующий код C_{i+1} отличается от предыдущего C_i точно в одной позиции, то последовательность $(n+1)$-разрядных кодов $C_1 0, C_2 0, \dots, C_m 0, C_m 1, C_{m-1} 1, \dots, C_2 1, C_1 1$ представляет все возможные бинарные $(n+1)$-разрядные коды, так что каждый следующий код отличается от предыдущего точно в одной позиции. Например, для $n = 3$ получаем следующую последовательность бинарных кодов Грэя: 000, 100, 110, 010, 011, 111, 101, 001. Требуется написать программу, которая по заданному n ($2 \le n \le 16$) выводит в текстовый файл output.txt последовательность бинарных кодов Грэя. Коды разделяются пробелом или переводом на новую строку.

15. Разнообразные строки. Под разнообразностью строки будем понимать количество символов, которые встречаются в ней ровно один раз. Например, разнообразность строки INFORMATICS – 9, поскольку все символы строки, кроме I, встречаются в ней ровно один раз. Написать программу, которая для заданной строки S находит в ней подстроку, имеющую

наибольшую разнообразность. Если таких строк несколько, то программы находит ту, которая является наименьшей в лексикографическом порядке. Строка A лексикографически меньше строки B, если A – начало строки B или если A отличается от B впервые на i-ом символе ($a_i \neq b_i$) и при этом a_i идет в алфавите раньше чем b_i. Например, слово SOL меньше слов SOLVE, START и TIME. Входной файл input.txt содержит строку, состоящую только из латинских букв. Длина строки не превышает 2000 символов. Выходной файл output.txt содержит подстроку исходной строки, имеющую максимальную разнообразность. Если таких строк несколько, то выводится минимальная в лексикографическом порядке строка.

В таблице 8 приведен пример файлов входных и выходных данных.

Таблица 8 – Пример файлов входных и выходных данных

input.txt	output.txt
ABBAC	BAC
OLYMP	OLYMP
AAA	A

16. Гипер почти перестановочная строка. Строка называется перестановочной, если она состоит из N первых заглавных букв в произвольном порядке (т.е. A, BA, CABED, DCBA для $N = 1, 2, 5, 4$ соответственно). Строка называется почти перестановочной, если из нее возможно получить перестановочную строку удалением ровно одного символа (т.е. BA, ABCCD, BCAB, ATB – почти перестановочные строки для $N = 1, 4, 3, 2$). Строка называется гипер почти перестановочной для некоторого N, если все ее подстроки длины $N + 1$ являются почти перестановочными (т.е. ABCBACB, ABCTABC, CBACAB – гипер почти перестановочные строки для $N = 3$). Написать программу, которая для двух заданных перестановочных строк S_1 и S_2 находит кратчайшую гипер почти перестановочную строку, содержащую S_1 и S_2 в качестве подстрок для

некоторого заданного N. Входные данные для программы задаются в текстовом файле input.txt. Результат – искомую строку, программа выводит в текстовый файл output.txt. При наличии нескольких результатов, программа выводит любой из них.

В таблице 9 приведен пример файлов входных и выходных данных.

Таблица 9 – Пример файлов входных и выходных данных

input.txt	output.txt
3 ABC ACB	ABCACB

17. Опечатки. При наборе текста довольно часто возникают опечатки из-за неправильного нажатия на клавиши. Например, некоторые буквы оказываются заменены на другие, появляются лишние буквы, некоторые буквы исчезают из слов. В большинстве случаев эти опечатки можно исправить автоматически. В частности, существуют методы проверки орфографии, основанные на поиске в словаре слов, похожих на проверяемые. Два слова называются похожими, если можно удалить из каждого слова не более одной буквы, так, чтобы слова стали одинаковыми, возможно пустыми. Например, слова spot и sport похожи, так как одно и то же слово spot можно получить из первого слова без удаления буквы, а из второго – удалением буквы r. Требуется написать программу, которая для каждого слова проверяемого текста определяет количество похожих на него слов в словаре. Входные данные для программы задаются в текстовом файле input.txt, а результаты программа выводит в текстовый файл output.txt. Формат входных данных: в первой строке записаны натуральные числа N – общее число слов в словаре ($N \geq 1$) и M – количество слов в проверяемом тексте ($M \geq 1$). При этом $N + M \leq 20000$. В последующих N строках записаны слова, входящие в словарь, по одному слову на строке. Все слова словаря различны. Далее

следуют M строк, в которых записаны слова проверяемого текста, по одному слову на строке. Слова состоят из строчных и прописных букв (прописные и строчные буквы считаются различными). Формат выходных данных: для каждого слова из текста в выходной файл помещается строка, содержащая это слово и через пробел количество слов из словаря, на которые оно похоже. Если в словаре имеется единственное похожее слово, то оно выводится в той же строке через пробел.

В таблице 10 приведен пример файлов входных и выходных данных.

Таблица 10 – Пример файлов входных и выходных данных

input.txt	output.txt
5 8 father and or mother a Father and mather go o for e walk	Father 1 father and 1 and mather 2 go 1 or o 2 for 1 or e 1 a walk 0

18. Расшифровка. Компания по защите интеллектуальной собственности решила повысить уровень защищенности своих операционных систем путем шифрования всех сообщений, передаваемых внутри ее локальных сетей. Любое допустимое в компании сообщение представляет собой строку $S = s_1 s_2 \dots s_n$ состоящую исключительно из букв латинского алфавита. Шифрование сообщения осуществляется в K фаз. На каждой фазе

строка S заменяется строкой, в которой сначала располагаются все буквы строки S, стоящие на позициях с номерами, являющимися простыми числами (первый блок), а затем – все остальные буквы (второй блок). Напомним, что число называется простым, если оно натуральное и имеет ровно два натуральных делителя. Относительный порядок букв в каждом из двух блоков остается неизменным. Например, строка $S = abcdefgh$ на первой фазе шифруется в строку $S = bcegadfh$. Если осуществляется вторая фаза шифрования, то строка S примет вид $S = ceafbgdh$. После передачи зашифрованного сообщения по сети оно должно быть дешифровано, чтобы получатель смог прочитать исходную запись.

Требуется написать программу, осуществляющую дешифрование заданной строки, являющейся результатом шифрования строки S после K фаз. Имя входного файла: input.txt. Имя выходного файла: output.txt. Формат входных данных:

В первой строке входного файла input.txt содержится натуральное число K $(1 \leq K \leq 100)$, вторая строка содержит сообщение S после K фаз шифрования, состоящее из n $(1 \leq n \leq 255)$ как прописных, так и строчных букв латинского алфавита. Регистр букв в процессе шифрования остается неизменным. Формат выходных данных:

Выходной файл output.txt должен содержать только одну дешифрованную строку S.

В таблице 11 приведен пример файлов входных и выходных данных.

Пример 11 – Пример файлов входных и выходных данных

INPUT.TXT	OUTPUT.TXT
2 CEAFBGDH	ABCDEFGH
1 BAAAACB	ABACABA

19. Криптоанализ. Ключ криптограммы состоит из блоков длиной 2 слова (4 байта), записанных в 16 системе счисления (0, 1,…, 9, A, B, C, D, E, F). Криптоаналитику стало известно, что один из блоков состоит из символов 2, 5, 7. Причем символы 2 и 5 повторяются ровно по два раза. Необходимо написать программу, которая выводит в файл output.txt все возможные варианты этого блока.

20. Забывчивый клиент. Клиент банка забыл четырёхзначный шифр своего сейфа (четырёхзначное десятичное число), но помнил, что этот шифр – простое число, а произведение его цифр равно 243. За какое наименьшее гарантированное число попыток он наверняка сможет открыть сейф? Необходимо составить программу, которая решает эту проблему. В текстовый файл output.txt сначала вывести количество всех необходимых проверяемых четырёхзначных комбинаций, а затем через пробел все возможные четырёхзначные комбинации, которых минимально достаточно.

21. Сигнализация. Подземный бункер состоит из n комнат, соединённых $n-1$ коридорами. Каждый коридор соединяет две различные комнаты и имеет определённую длину. Бункер устроен таким образом, что из любой комнаты i можно дойти в любую комнату j. Заметим, что существует единственный такой путь, не проходящий по одному и тому же коридору дважды. Сумма длин коридоров, составляющих этот путь, называется расстоянием между комнатами i и j обозначается $\rho(i, j)$.

Каждая комната бункера оборудована звуковой сигнализацией, состоящей из сирены и датчика звука, который её включает. Сирена, включённая в комнате i, активирует датчик звука в каждой комнате, расстояние до которой не превосходит расстояние d_i, определяемое мощностью этой сирены. Другими словами, включение сирены в комнате i автоматически включает сирену во всех комнатах j, таких что $\rho(i, j) \leq d_i$. Эта

сирена, в свою очередь, может вызвать автоматическое включение других сирен и так далее.

В случае возникновения чрезвычайной ситуации некоторые сирены необходимо включать вручную, после чего звук от них автоматически включит сирены в других комнатах. Правила безопасности предписывают выбор такого набора сирен для ручного включения, который в конце концов приведёт к автоматическому включению сирен во всех комнатах.

Требуется написать программу, которая определяет минимальное количество сирен в наборе, удовлетворяющем правилам безопасности.

Первая строка входных данных, которые берутся из файла alarm.in, содержит единственное число n – количество комнат. Вторая строка содержит последовательность из n целых чисел d_i, i-ое из них равно максимальному расстоянию, на котором расположенная в комнате i сирена активирует датчики ($0 \le d_i \le 10^9$).

Последующие $n-1$ строк описывают коридоры бункера. В i-ой из них находятся три целых числа: u_i, v_i, l_i, где u_i, v_i – номера различных комнат, соединённых коридором i, а l_i – длина этого коридора ($1 \le u_i, v_i \le n$, $1 \le l_i \le 10^9$).

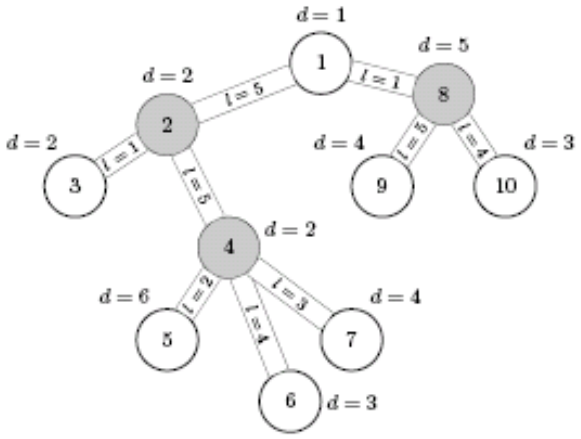
Выходные данные, которые нужно записать в файл alarm.out, должны состоять из единственного числа – минимального количества сирен, которые необходимо включить вручную.

Примеры

alarm.in	alarm.out
10 1 2 2 2 6 3 4 5 4 3 1 2 5 2 3 1 2 4 5 4 5 2 4 6 4 4 7 3 1 8 1 8 9 5 8 10 4	3

Замечания

В тесте из примера сирена в комнате 4 включает сирену в комнате 5, которая, в свою очередь, включает сирены в комнатах 6 и 7. Сирена в комнате 2 включает сирену в комнате 3. Сирена в комнате 8 включает сирены в комнатах 1, 9 и 10.



22. Сеть друзей. Студенты ФКТиПМ КубГУ решили разработать собственную социальную сеть, которая должна автоматически подбирать для каждого пользователя потенциальных друзей. При регистрации каждому пользователю предлагается пройти психологическое тестирование, по результатам которого определяются значения трёх психологических характеристик этого пользователя. Значение каждой характеристики – натуральное число.

Считается, что если у двух пользователей различаются значения всех трёх характеристик, то они будут постоянно ссориться, а если совпадают значения всех трёх или двух характеристик, то им будет скучно. Таким образом, потенциальными друзьями являются только такие пары пользователей, у которых совпадают значения ровно одной характеристики, а значения двух других – различаются.

Требуется написать программу, которая по данным n тройкам (a_i, b_i, c_i) значений характеристик каждого из пользователей определяет количество пар потенциальных друзей, то есть таких пар индексов $i < j$, что из трёх равенств $a_i = a_j$, $b_i = b_j$, $c_i = c_j$ выполняется ровно одно.

Входные данные берутся из файла onlyone.in. Первая строка входных данных содержит число n – количество пользователей. Каждая из последующих n строк содержит три натуральных числа a_i, b_i, c_i – значения характеристик i-го пользователя.

Выходные данные, которые нужно записать в файл onlyone.out, содержат только искомое количество пар потенциальных друзей.

onlyone.in	onlyone.out
3 1 2 3 1 4 5 1 2 4	2
4 100 100 100 100 100 100 100 99 99 99 99 100	5

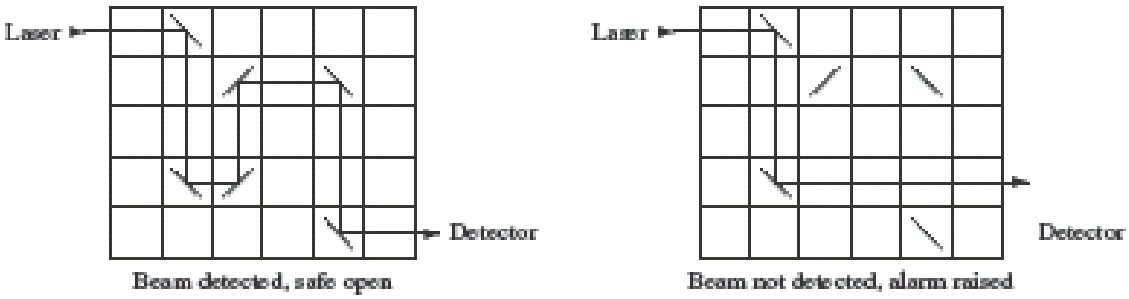
23. Множители. Основная теорема арифметики утверждает, что каждое целое число, большее 1, может быть однозначно представлено как произведение одного или нескольких простых чисел. При уникальном наборе простых множителей возможно несколько их упорядочений. Например, 10 = 2*5, 10 = 5*2. 20 = 2*2*5, 20 = 2*5*2, 20 = 5*2*2. Пусть $f(k)$ будет числом различных расположений простых множителей натурального числа k. Так что $f(10) = 2$, $f(20) = 3$. Дано натуральное число n, всегда существует по крайней мере одно такое натуральное k, для которого $f(k) = n$. Требуется вычислить по заданному n такое наименьшее число k.

Входные данные берутся из файла naturals.in. Файл содержит какое-то количество произвольных натуральных чисел из диапазона $2, \ldots, 2^{63}$. В одной строке содержится ровно одно число. Выходные данные помещаются в файл

naturals.out. В каждой строке находится два числа: число из входного файла n и соответствующее ему значение k, такое, что $f(k) = n$.

1 2 3 105	1 2 2 6 3 12 105 720

24. Надёжность сейфа. Последним изобретением компании, производящей сейфы, является механизм оптического закрытия, который использует лазерный луч, проходящий с помощью прямоугольной решётки с несколькими зеркалами.



Когда лазер включён, луч горизонтально входит в верхней строке решётки слева. Луч отражается каждым зеркалом, на которое он падает. Каждое зеркало имеет 45-градусную ориентацию, либо /, либо \. Если луч горизонтально выходит из нижней строки направо, то он фиксируется (опознаётся) и сейф открывается. В противном случае сейф остаётся закрытым и включается тревога. Каждый сейф имеет недостающее зеркало, которое не позволяет лазерному лучу пройти через решётку (см. правый рисунок выше). Сейф имеет механизм, позволяющий пользователю опустить одно зеркало в пустую клетку решётки. Законный пользователь знает правильную позицию и ориентацию недостающего зеркала (/-зеркало в 4-ой строке и 3-ей колонке на рисунке выше слева) и может таким образом открыть сейф. Без этого знания пользователь должен угадывать правильную комбинацию, что может быть затруднительным для сейфов с большими решётками.

Необходимо написать программу для определения надёжности конкретного сейфа. Безопасный сейф не открывается без вставки зеркала, и есть по крайней мере одна позиция и ориентация для недостающего зеркала. На деле таких позиций и ориентаций может быть несколько.

Информация о сейфе задаётся в файле safe.in. В первой строке этого файла задаются 4 числа: r, c, m, n ($1 \leq r, c \leq 1000000$; $0 \leq m, n \leq 20000$). Сетка механизма сейфа имеет r строк и c столбцов. Каждая из следующих m строк содержит два целых числа r_i и c_i ($1 \leq r_i \leq r$ и $1 \leq c_i \leq c$), которые задают строку и столбец для /-зеркала. Следующие n строк задают положения для \-зеркал таким же образом. $m + n$ позиций зеркал попарно различны.

Выход формируется в файле safe.out. В этот файл помещаются следующие результаты: 0, если сейф открывается без недостающего зеркала; k, r, c, если сейф не открывается без недостающего зеркала и существует точно k позиций, где вставка недостающего зеркала открывает сейф, а (r, c) – лексикографически наименьшая такая пара номеров строки и столбца для недостающего зеркала; позиции, где /-зеркало и \-зеркало открывают сейф, считаются только однажды; impossible, если сейф не открывается как с недостающим зеркалом, так и без него.

5 6 1 4 2 3 1 2 2 5 4 2 5 5 100 100 0 2 1 77 100 77 100 100 0 0	Case 1: 2 4 3 Case 2: 0 Case 3: impossible