Лабораторная работа №8

В рамках данной лабораторной работы необходимо решить предложенные задачи на языке программирования высокого уровня из предложенного перечня. Варианты задач находятся в отдельном файле. В качестве результата выполнения практической работы необходимо приложить **архив с файлами решенных задач, а также отчет о выполнении работы**. Файлы решенных задач представляют собой файлы с исходным кодом с установленным расширением. При решении задач необходимо писать свои собственные структуры данных и методы сортировки, использование встроенных не допускается (кроме массивов).

Отчет о выполнении практической работы должен содержать:

1. Титульный лист
2. Содержание
3. 4 параграфа, в которых раскрыто решение каждой задачи. По каждой задаче необходимо представить следующую информацию:

3.1. *Условие задачи*. Берется из файла.

3.2. *Ход решения задачи*. Студент описывает логику решения данной задачи. Какие алгоритмы и для чего использованы, как построена программа. Данная часть является описательной. Здесь следует говорить именно о построении алгоритма, опуская процессы ввода и вывода данных (если это не является основной сутью алгоритма).

3.3. *Листинг программы с комментариями*. Копируется весь программный код.

3.4. *Тестирование программы*. Если для задачи предусмотрены автотесты, приложить скриншот их прохождения. Если нет: Составляется таблица, содержащая следующие поля: номер теста, входные данные, результат выполнения программы, корректность, время выполнения (мс), затраченная память (Мб). Составляется не менее 10-ти тестовых наборов данных согласно условию задачи. Тестовые наборы входных данных студент составляет самостоятельно. В обязательном порядке программа тестируется на граничных наборах входных данных (например, если N варьируется от 0 до 10^9, то обязательно рассмотреть решение задачи при N=0 и при N близком к 10^9). Если написанная программа не позволяет решить задачу при граничных входных данных, все равно включить в тест и в качестве результата написать "Не решено". В столбце "входные данные" данные впечатываются вручную, в столбце "результат..." представляется скриншот выполнения программы (если не влезает на одну страницу, делать несколько скриншотов).

Каждая лабораторная работа защищается на занятии преподавателю.

Задачи по теме 8. Строковые алгоритмы

Задача 1. Packed Prefix

Ограничение по времени: 0.5 с.

Ограничение по памяти: 64 Mb.

Вам даны строки в запакованном виде. Определим запакованную строку (ЗС) рекурсивно. Строка, состоящая только из строчных букв английского алфавита, является ЗС. Если A и B - корректные ЗС, то и AB является ЗС. Если A – ЗС, а n – однозначное натуральное число, то $n[A]$ тоже ЗС. При этом надпись $n[A]$ означает, что при распаковке строка A записывается подряд n раз. Найдите наибольший общий префикс распакованных строк и выведите его (в распакованном виде).

Иными словами, пусть сложение – это конкатенация двух строк, а умножение строки на число – повтор строки соответствующее число раз. Пусть функция f умеет принимать ЗС и распаковывать её. Если ЗС D имеет вид $D = AB$, где A и B тоже ЗС, то $f(D) = f(A) + f(B)$. Если $D = n[A]$, то $f(D) = f(A) \times n$.

Формат ввода

В первой строке записано число n $(1 \leq n \leq 1000)$ – число строк. Далее в n строках записаны запакованные строки. Гарантируется, что эти строки корректны, то есть удовлетворяют указанному рекурсивному определению. Длина строк после распаковки не превосходит 10^5.

Формат вывода

Выведите наибольший общий префикс распакованных строк.

Примеры

Ввод	Вывод
3 2[a]2[ab] 3[a]2[r2[t]] a2[aa3[b]]	aaa
3 abacabaca 2[abac]a 3[aba]	aba

Примечания

Сложение подразумевается, как конкатенация двух строк. Умножение строки на число – повтор строки соответствующее число раз. Пусть функция f умеет принимать ЗС и распаковывать её. Если ЗС D имеет вид $D = AB$, где A и B тоже ЗС, то $f(D) = f(A) + f(B)$. Если $D = n[A]$, то $f(D) = f(A) \times n$.

Задача 2. Шпаргалка

Ограничение по времени: 0.7 с.

Ограничение по памяти: 256 Mb.

Вася готовится к экзамену по алгоритмам и на всякий случай пишет шпаргалки. Чтобы уместить на них как можно больше информации, он не разделяет слова пробелами. В итоге получается одна очень длинная строка. Чтобы на самом экзамене из-за нервов не запутаться в прочитанном, он просит вас написать программу, которая по этой длинной строке и набору допустимых слов определит, можно ли разбить текст на отдельные слова из набора.

Более формально: дан текст T и набор строк $s_1, \dots, s_n$. Надо определить, представим ли T как $s_{k_1} s_{k_2} \dots s_{k_\gamma}$, где k_i - индексы строк. Индексы могут повторяться. Строка s_i может встречаться в разбиении текста T произвольное число раз. Можно использовать не все строки для разбиения. Строки могут идти в любом порядке.

Формат ввода

В первой строке дан текст T, который надо разбить на слова. Длина T не превосходит 10^5. Текст состоит из строчных букв английского алфавита. Во второй строке записано число допустимых к использованию слов $1 \leq n \leq 100$. В последующих n строках даны сами слова, состоящие из маленьких латинских букв. Длина каждого слова не превосходит 100.

Формат вывода

Выведите «YES», если текст можно разбить на слова из данного словаря, или «NO» в ином случае.

Примеры

Ввод	Вывод
examiwillpasstheexam 5 will pass the exam i	YES
abacaba 2 abac caba	NO
abacaba 3 abac caba aba	YES

Задача 3. Книга Сандро

Источник задачи.

Задача 4. Палиндромы

Источник задачи.

Задача 5. Палиндром. Он же палиндром

Источник задачи.

Задача 6. Странный диалог

Источник задачи.

Задача 7. Басня о строке

Источник задачи.

Задача 8. Шифр Бэкона

Источник задачи.

Задача 9. Свобода выбора

Источник задачи.

Задача 10. Последнее слово Джека

Источник задачи.