Содержание

Вы не умеете читать/выводить данные, открывать файлы? Воспользуйтесь примерами.

В некоторых задачах большой ввод и вывод. Пользуйтесь быстрым вводом-выводом.

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) переопределение стандартного аллокатора ускорит вашу программу.

Обратите внимание на компилятор `GNU C++11 5.1.0 (TDM-GCC-64) inc`, который позволяет пользоваться дополнительной библиотекой. Под ним можно сдать вот это.

Must have

Задача 4A. Подстроки-3 [0.2 sec, 256 mb]

Даны K строк из маленьких латинских букв. Найдите их наибольшую общую подстроку.

Формат входных данных

В первой строке число K ($1 \leqslant K \leqslant 10$). В следующих K строках — собственно K строк (длины строк от 1 до 10 000).

Формат выходных данных

Наибольшая общая подстрока.

Примеры

stdin	stdout
3 abacaba mycabarchive acabistrue	cab

Замечание

Нужно написать хороший код, чтобы пройти TL.

В частности, вам возможно понадобится рукописная хеш-таблица с открытой адресацией.

Задача 4B. Бункеры [2 sec, 256 mb]

Петя и Вася с упоением играют в шпионов. Сегодня они планируют, где будут расположены их секретные бункеры и штаб-квартира.

Пока Петя и Вася решили, что им понадобится ровно n бункеров, которые для секретности будут пронумерованы числами от 1 до n. Некоторые из них будут соединены двусторонними тоннелями, причем для надежности и секретности по тоннелям можно будет попасть из любого бункера в любой единственным образом. Петя и Вася даже решили, какие из бункеров будут соединены тоннелями, но выбрать, какой из них будет штаб-квартирой, они не могут. Мальчики хотят выбрать ее и разделить оставшиеся бункеры между собой таким образом, чтобы им досталось поровну бункеров и к штаб-квартире вело бы ровно два тоннеля: один от бункера, принадлежащего Васе, другой — от бункера, принадлежащего Пете.

Уставший Петя пошел к себе домой, а утром Вася показал ему план, на котором бункеры были обозначены точками, а тоннели отрезками. Кроме того, Вася выбрал штаб-квартиру таким образом, что нарисованный им план был симметричен относительно некоторой прямой, проходящей через точку, которая соответствовала штаб-квартире. При этом бункеры, принадлежащие Пете находились с одной стороны от этой прямов, а бункеры, принадлежащие Васе, с другой стороны.

Хотя Петя почти сразу показал Васе, что тот ошибся и не нарисовал половину бункеров, ему стало интересно, можно ли выбрать штаб-квартиру и нарисовать такой симметричный план.

Формат входных данных

В первой строке входного файла находится одно целое число n ($1 \leqslant n \leqslant 10^5$) — количество бункеров. В следующих $n - 1$ строках находится по два целых числа u_i и v_i ($1 \leqslant u_i, v_i \leqslant n, u_i \neq v_i$) — номера бункеров, которые соединяет i-й тоннель. Гарантируется, что между любыми двумя бункерами существует единственный путь.

Формат выходных данных

В выходной файл выведите «YES», если можно выбрать штаб-квартиру и нарисовать такой план, или «NO» если это невозможно.

Примеры

stdin	stdout
2 1 2	NO
3 1 2 2 3	YES

Обязательные задачи

Задача 4C. Суффиксное дерево [1 sec, 256 mb]

Дана строка s. Постройте сжатое суффиксное дерево для строки s и выведите его. Найдите такое дерево, которое содержит минимальное количество вершин.

Формат входных данных

В первой строке записана строка s ($1 \leqslant |s| \leqslant 10^5$), последний символ строки доллар «\$», остальные символы строки маленькие латинские буквы.

Формат выходных данных

Пронумеруйте вершины дерева от 0 до $n-1$ в порядке обхода в глубину, обходя поддеревья в порядке лексикографической сортировки исходящих из вершины рёбер. Используйтся ASCII-коды символов для определения их порядка.

В первой строке выведите число n – количество вершин дерева. В следующих $n-1$ строках выведите описание вершин дерева, кроме корня, в порядке увелечения их номеров.

Описание вершины дерева v состоит из трёх целых чисел: p, lf, rf, где p ($0 \leqslant p \leqslant n, p \neq v$) – номер родителя текущей вершины. На ребер ведущем из p в v написана подстрока $s[lf..rf)$ ($0 \leqslant lf < rf \leqslant |s|$).

Примеры

stdin	stdout
aaa$	7 0 3 4 0 0 1 2 3 4 2 1 2 4 3 4 4 2 4
b$	3 0 1 2 0 0 2
ababa$	10 0 5 6 0 0 1 2 5 6 2 1 3 4 5 6 4 3 6 0 1 3 7 5 6 7 3 6

Задача 4D. Ненокку [1 sec, 256 mb]

Очень известный автор не менее известной книги решил написать продолжение своего произведения. Он писал все свои книги на компьютере, подключенном к интернету. Из-за такой неосторожности мальчику Ненокку удалось получить доступ к еще ненаписанной книге. Каждый вечер мальчик залазил на компьютер писателя и записывал на свой компьютер новые записи. Ненокку, записав на свой компьютер очередную главу, заинтересовался, а использовал ли хоть раз писатель слово "книга". Но он не любит читать книги (он лучше полазает в интернете), и поэтому он просит вас узнать есть ли то или иное слово в тексте произведения. Но естественно его интересует не только одно слово, а достаточно много.

Формат входных данных

В каждой строчке входного файла записано одна из двух записей.

1. `? <слово>` (<слово> — это набор не более 50 латинских символов);
2. `A <текст>` (<текст> — это набор не более 10^5 латинских символов).

1 означает просьбу проверить существование подстроки <слово> в произведение.
2 означает добавление в произведение <текст>.
Писатель только начал работать над произведением, поэтому он не мог написать более 10^5 символов. Суммарная длина всех запросов не превосходит 15 мегабайт плюс 12140 байт.

Формат выходных данных

Выведите на каждую строчку типа 1 "`YES`", если существует подстрока <слово>, и "`NO`" в противном случае. Не следует различать регистр букв.

Пример

stdin	stdout
? love ? is A Loveis ? love ? WHO A Whoareyou ? is	NO NO YES NO YES

Замечание

Можно толкнуть хеши. Но вообще предполагается Укконен.

Дополнительные задачи

Задача 4E. Помогите, спасите! [4 sec, 256 mb]

Дана строка. Найдите для каждого её префикса количество различных подстрок в нём.

Формат входных данных

В единственной строке входных данных содержится непустая строка S, состоящая из N ($1 \le N \le 2 \cdot 10^5$) маленьких букв английского алфавита.

Формат выходных данных

Выведите N строк, в i-й строке должно содержаться количество различных подстрок в i-м префиксе строки S.

Примеры

stdin	stdout
aabab	1 2 5 8 11
atari	1 3 5 9 14

Задача 4F. Древо жизни (средняя) [5 sec, 256 mb]

Хайди устала расшифровывать древние пророчества и решила отправиться в штаб, чтобы отдохнуть и попытаться снова. Разумеется, она не может выкорчевать дерево и взять его с собой, поэтому он сделала точный рисунок дерева на бумаге. Подумав ещё, она сделала ещё несколько рисунков, которых суммарно получилось n (по числу вершин в Древе Жизни) — кто знает что может случиться?

В самом деле, по пути назад Хайди была атакована группой зомби. Хотя она и смогла с ними справиться, рисункам был причинён очень странный урон: с i-й копии пропала вершина с номером i, вместе со всеми своими рёбрами. На каждом рисунке зомби также стёрли номера вершин и перенумеровали оставшиеся $n-1$ вершину произвольными числами от 1 до n (по крайней мере, номера вершин всё ещё различны). *Более того, рисунки были случайным образом переупорядочены.*

Теперь Хайди хочет восстановить Древо Жизни по имеющемся у неё описаниям и рисункам (спискам рёбер).

Формат входных данных

В первой строке входных данных записано число $z \leq 20$ — количество тестовых примеров. Далее следуют описания z тестов.

Описание каждого теста начинается со строки содержащей числа n ($2 \leq n \leq 100$) и k (k является количеством рисунков, $k = n$ в данной задаче). В следующих строках содержится описание k рисунков. Описание i-го рисунка даётся строкой содержащей m_i — количество рёбер на соответствующем рисунке, а затем m_i строк описывающих рёбра, каждое из которых даётся парой целых чисел — индексы соединённых вершин.

Формат выходных данных

Если рисунки Хайди никак не могут описывать одно и то же дерево, то выведите `NO`. В противном случае выведите слово `YES` и $n-1$ строку с описанием любого дерева, которое могло послужить первоисточником для рисунков Хайди. Для каждого ребра выведите два целых числа — индексы соединённых вершин. Если подходящих решений несколько, разрешается вывести любое.

Пример

stdin	stdout
1 5 5 2 4 1 2 1 1 3 1 3 4 1 4 3 2 1 3 3 1 3 2 4 1 3 2 1 3 2 4 2	YES 2 5 4 2 3 2 5 1

Задача 4G. Идеальное хеширование [2 sec, 256 mb]

Вам задана архитектура компьютера, в рамках которой от вас требуется сделать программу, которая является идеальным хешом для заданного вам множества целых чисел.

Опишем архитектуру компьютера. У него есть 4 мебибайта $= 4 \times 2^{20}$ памяти, доступной только для чтения, содержащей программу, которую будет исполнять компьютер, и данные. Процессор компьютера содержит 256 регистров с названиями от `r0` до `r255`, каждый из них может сохранить 32-битное целое число. Перед исполнением программы значение каждого регистра кроме `r0` равно 0, а в `r0` содержится входные данные программы.

У компьютера есть специальный регистр, который называется *instruction pointer*. Программа хранится в памяти и изначально значение регистра `instruction pointer` равно 0. Выполнение программы проходит по следующему сценарию. Сначала читается номер инструкции, который хранится в памяти по адресу значения `instruction pointer`, после чего читаются аргументы инструкции и инструкция выполняется. Если во время выполнения инструкции не случился `jump`, то значение `instruction pointer` увеличивается на размер прочитанной инструкции, включая аргументы.

Программа должна быть отображением из множества заданных n целых чисел $A = \{a_1, a_2, \ldots, a_n\}$ в множество целых чисел от 0 до $2n - 1$. Она должна завершаться после не более чем 30 выполненных инструкций. Отображение не должно иметь коллизий. Формально, надо сделать программу в этой архитектуре, которая удовлетворяет следующим критериям:

- Если программе на вход подать одно из чисел из множества A, то она должна завершиться за не более чем 30 инструкций и вывести целое число от 0 до $2n - 1$.
- Для любых двух различных чисел a_i и a_j из множества A значения на выходе должны быть различны.

Вам задана некоторая документация.

- Все регистры содержать 32-битные целые числа
- Для сложения, вычитания и битовых операций не важно, числа знаковые или нет
- Команды `jump` с условием представляют числа как знаковые
- Результатом умножения или деления является знакового 64-битное число, которое сохраняется в два регистра. Регистр, который получает младшую часть результата сохраняет его как беззнаковое число
- Деление и взятие остатка принимают делимое в качестве 64-битных чисел, младшая часть интерпретируется как беззнаковое число, а старшая — как знаковое. Делитель — знаковое число.
- Деление и взятие остатка знаковых чисел производится так же, как это делается в архитектуре x86.
- Все числа в памяти сохраняются в порядке от младших байт к старшим (little endian).

От вас требуется понять, что должна содержать память, чтобы исполнялась программа, удовлетворяющая условиям, описанным выше.

Для любых входных данных из множества A ваша программа не должна пытаться читать данные извне 4МБ-диапазона и не должна делить на ноль. Значение регистра `instruction pointer` так же должно быть внутри 4МБ-диапазона.

Таблица инструкций

Инстр.	**Размер**	**Opcode**	**Что делает инструкция**
nop	1 байт	`00`	Ничего
add	4 байта	`01 r1 r2 r3`	`r3 := r1 + r2`
sub	4 байта	`02 r1 r2 r3`	`r3 := r1 - r2`
mul	5 байт	`03 r1 r2 r3 r4`	`(r3, r4) := r1 * r2` r3 получает младшие биты, а r4 — старшие биты произведения
div	6 байт	`04 r1 r2 r3 r4 r5`	`(r3, r4) := (r1, r5) div r2` r1 содержит младшие биты, r5 содержит старшие биты делимого, r3 получает младшие биты, а r4 — старшие биты частного
mod	5 байт	`05 r1 r2 r3 r4`	`r3 := (r1, r4) mod r2`, r1 содержит младшие биты, r4 содержит старшие биты делимого
and	4 байта	`10 r1 r2 r3`	`r3 := r1 and r2`
or	4 байта	`11 r1 r2 r3`	`r3 := r1 or r2`
xor	4 байта	`12 r1 r2 r3`	`r3 := r1 xor r2`
neg	2 байта	`20 r1`	`r1 := -r1`
not	2 байта	`21 r1`	`r1 := ~r1`
load	3 байта	`30 r1 r2`	`r1 := memory[r2]` 4 байта, начиная с адреса r2 копируются в регистр r1, первым копируется младший байт
put	6 байт	`31 r1 b0 b1 b2 b3`	`r1 := (b0, b1, b2, b3)` здесь b0 — младший байт числа, положенного в регистр, а b3 — старший
jmp	5 байт	`40 b0 b1 b2 b3`	Сделать `jump` к инструкции (b0, b1, b2, b3)
jz	6 байт	`41 r1 b0 b1 b2 b3`	Если `r1 == 0` сделать `jump` к инструкции (b0, b1, b2, b3)
jnz	6 байт	`42 r1 b0 b1 b2 b3`	Если `r1 != 0` сделать `jump` к инструкции (b0, b1, b2, b3)
jg	6 байт	`43 r1 b0 b1 b2 b3`	Если `r1 > 0` сделать `jump` к инструкции (b0, b1, b2, b3)
jge	6 байт	`44 r1 b0 b1 b2 b3`	Если `r1 >= 0` сделать `jump` к инструкции (b0, b1, b2, b3)
jl	6 байт	`45 r1 b0 b1 b2 b3`	Если `r1 < 0` сделать `jump` к инструкции (b0, b1, b2, b3)
jle	6 байт	`46 r1 b0 b1 b2 b3`	Если `r1 <= 0`сделать `jump` к инструкции (b0, b1, b2, b3)
ret	1 байт	`ff`	Завершить программу. Выходные данные содержатся в регистре `r0`

Формат входных данных

Первая строка содержит целое число n ($1 \leqslant n \leqslant 100\,000$).

Вторая строка содержит n различных целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

Выведите данные, содержащиеся в памяти компьютера. Вам разрешается вывести любое число байт от 1 до 4 194 304. Остальная часть памяти заполнится нулями. Каждый байт должен быть выведен как шестнадцатеричное число из двух цифр. Не печатайте пробелы.

Пожалуйста, выводите только требующийся префикс памяти вашей программы: не выводите конечные нули, чтобы тестирование проходило быстрее.

Примеры

stdin	stdout
3 2 3 9	3101040000000500010003ff
2 6 10	300000ff00000000000001000000