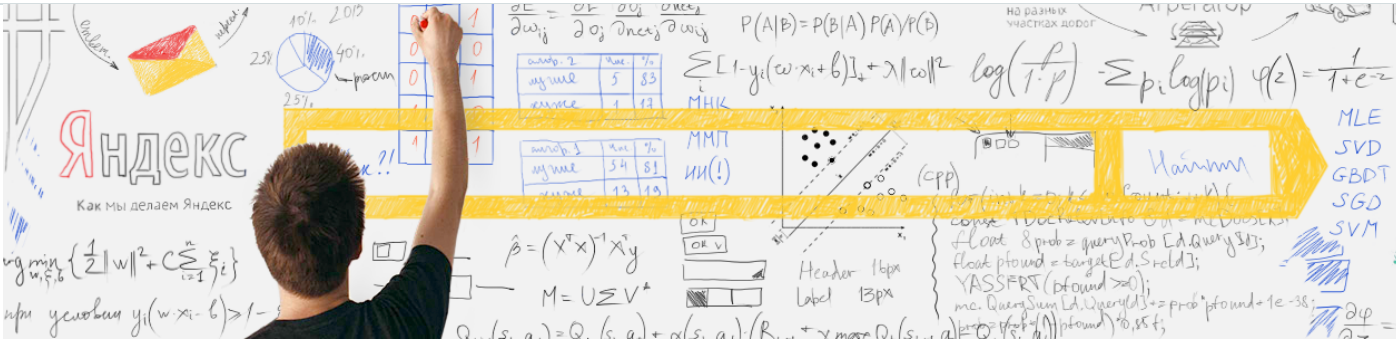


Яндекс 795,02
Как мы делаем Яндекс

22 августа 2013 в 19:16

Разбор всех задач и результаты Яндекс.Алгоритма

Алгоритмы*, Блог компании Яндекс, Спортивное программирование*

Буквально пару часов назад в Санкт-Петербурге завершился открытый чемпионат по программированию Яндекс.Алгоритм 2013. Состязания состояли из нескольких онлайн-раундов по 100 минут, за победу боролись более 3000 программистов из 84 стран. По результатам трёх отбо раундов в финал вышли 25 лучших.

Финалисты должны были решить шесть алгоритмических задач за 100 минут. Первое место занял недавний победитель ACM ICPC 2013 в сос команды НИУ ИТМО Геннадий Короткевич (tourist), который набрал меньше всего штрафного времени. Второе место досталось выпускнику Н ИТМО Евгению Капуну (eatmore). Третье место занял представитель Тайваня Ши Бисюнь.

В подготовке заданий для чемпионата участвовали специалисты из нескольких стран: России, Беларуси, Польши и Японии. Главными составителями задач стали разработчики минского офиса Яндекса (как и все сотрудники компании, к участию в состязаниях они не допуска Мы попросили всех авторов разобрать задания, которые они подготовили для участников Яндекс.Алгоритма. Кстати, все задачи не удалось р никому, лучший результат — три решённые задачи — показали только три участника.

Задача A. Сокращайте!

Автор задачи и разбора: Jakub Pachocki, Warsaw U

Имя входного файла: stdin
Имя выходного файла: stdout
Ограничение по времени: 8 секунд
Ограничение по памяти: 512 мегабайт

Обычно на командных соревнованиях по программированию, помимо полного названия вуза, задаётся сокращённое. На одном очень престиж соревновании команды планируют рассадить так, что из двух команд команда с лексикографически наименьшим сокращённым именем сидит к буфету. С учётом этого тренеры каждого вуза стремятся сократить название вуза так, чтобы результат получился лексикографически минимальным.

Обозначим как *слово* непустую подстроку названия университета, которая состоит из букв, причём ни символ непосредственно перед этой подстрокой, ни символ сразу после неё буквами не являются. При сокращении названия вуза разрешается заменить любой суффикс некотор слова знаком точки («.»).

При этом:
- Не разрешается заменять точкой всё слово.
- Если слово остаётся неизменным (то есть выбран пустой суффикс), точка не ставится.

Разрешается не сокращать название университета вообще, оставив его неизменным.
При сравнении названий все символы, не являющиеся латинскими буквами, удаляются, все заглавные латинские буквы заменяются соответствующими строчными и полученные в результате строки сравниваются между собой.

Формат входного файла. Первая строка входного файла содержит название университета. Название может содержать заглавные и строчные латинские буквы, пробелы, запятые («,») и дефисы («-»).
Гарантируется, что строка не начинается и не заканчивается пробелом и не содержит два последовательных пробела. Количество символов (включая пробелы) во входном файле не превосходит 10^6.

Формат выходного файла. Выведите лексикографически наименьшее сокращённое название университета. Запрещается изменять капитализ букв и положение символов, не являющихся буквами, по сравнению с исходным названием. Если лексикографически наименьших сокращений несколько, выведите любое.

stdin	**stdout**
University of Warsaw	Uni. of W
Saint Petersburg National Research University of IT, Mechanics and Optics	Sain. Pe. Na. Research Uni. of I., M. and O.
University of Illinois at Urbana-Champaign	Uni. of I. at U.-C
,Carnegie--,--MelloN-,-, — University-	,Ca.--,--MelloN-,-, — U.-
Udmurtian University	Udm. U.

Note. «Uni. of W.» является лексикографически наименьшим сокращением для «University of Warsaw», так как строка «uniofw» является лексикографически меньшей, чем строки, соответствующие другим варинтам, к примеру, «uow», «uniofwa», «universityofwarsaw».

Решения задачи A.

После разбора ввода задача сводится к следующей:

> Для n строк $s_1,\dots s_n$ суммарной длины N найти такие их непустые префиксы $p_1,\dots, p_n$, что строка
>
> $p_1p_2\cdots p_n$
>
> является лексикографически наименьшей.

Предположим, что мы уже нашли $p_{k+1},\dots, p_n$ такие, что строка $p_{k+1}\,p_{k+2}\cdots p_n$ является лексикографически наименьшей. Заметим, что данный $p_{k+1}, \dots, p_n$ оптимален при любом выборе $p_1, \dots, p_k$. Это следует из того факта, что если для двух строк t и t' верно $t < t'$, то $wt < wt'$ для люб строки w.

Каким образом эффективно найти оптимальный префикс p_k по заданным $p_{k+1}, \dots, p_n$?
Положим $t := p_{k+1}p_{k+2}\cdots p_n$. Нам нужно найти такой непустой префикс p строки s_k, что строка pt является лексикографически минимальной.

Существует алгоритм, который позволяет эффективно сравнивать строки pt и $p't$, используя хэш:
- бинарным поиском находим наибольший общий префикс строк pt и $p't$;
- сравниваем символы, идущие сразу за этим префиксом.

При использовании рандомизированного хэша алгоритм работает за ($\log N$) с весьма высокой вероятностью.
Для того чтобы найти p_k, требуется провести $|s_k| - 1$ таких сравнений, что даёт время ($|s_k| \log N$) для вычисления p_k. Суммируя все s_k, полу время работы алгоритма ($N \log N$).

Задача B. Анюта и «побитовое И»

Авторы задачи и разбора — Алексей Толстиков и Роман Удовиченко.
Имя входного файла: stdin
Имя выходного файла: stdout
Ограничение по времени: 2 секунд
Ограничение по памяти: 512 мегабайт

Маленькая Анюта решила изучить битовые операции над числами. Сегодня она изучила операцию «побитовое И». Побитовое И — это бинар операция, действие которой эквивалентно применению логического И к каждой паре битов, которые стоят на одинаковых позициях в двоичн представлениях операндов (в двоичном представлении допустимы лидирующие нули). Другими словами, если оба соответствующих бита операндов равны 1, результирующий двоичный разряд равен 1; если же хотя бы один бит из пары равен 0, результирующий двоичный разр равен 0.

По удивительному стечению обстоятельств, как раз сегодня родители Анюты принесли с работы два набора чисел. Папа принёс с работы наб состоящий из P чисел, а мама — набор B, состоящий из M чисел. Все числа в наборах были целые, неотрицательные, а также не превосходя 65535. Кроме того, числа в наборах были пронумерованы, начиная с 1.

Радостная Анюта тут же принялась применять изученную операцию к числам из этих наборов. Чтобы не обидеть никого из родителей, она применяла побитовое И к тем парам чисел, одно из которых было из папиного набора, а другое — из маминого.

Пока Анюта была увлечена вычислениями, родителям стало интересно, сколькими способами их дочка может получить некоторое число X? Количество различных способов можно посчитать, как количество различных пар (i, j) таких, что $1 \le i \le P$, $1 \le j \le M$, A_i *AND* $B_j = X$. Посколь родители Анюты не менее любознательные, чем их дочка, они хотят найти искомое количество способов не только для одного числа X, а сра K чисел X_i. Помогите родителям!

Формат входного файла. В первой строке находятся три целых числа P, M и K ($1 \le P, M, K \le 10^5$), разделённые пробелами — количество чис наборе папы, количество чисел в наборе мамы, и количество чисел, интересующих родителей соответственно. Вторая строка содержит P чис разделённых пробелами — набор чисел, который принёс папа. Третья строка содержит M чисел B_i, разделённых пробелами —- набор чисел, который принесла мама. Четвёртая строка содержит K чисел X_i, разделённых пробелами — набор чисел, для которых нужно посчитать иском количество пар. Все числа в наборах A, B и X целые, неотрицательные и не превосходят 65535.

Формат выходного файла. Выведите K строк, содержащих по одному числу. Число в i-ой строке должно равняться количеству способов полу число X_i с помощью применения операции побитового И к числам из наборов A и B по правилам, описанным в условии.

Примеры

stdin	stdout
4 4 4 1 2 3 4 1 2 3 4 1 2 3 4	3 3 1 1
1 2 3 0 1 2 0 1 2	2 0 0

Решение задачи B

Даны два массива A и B, состоящие из целых неотрицательных чисел до 2^{16}. Также дан массив запросов Q, тоже содержащий числа до 2^{16}. Для каждого запроса k сказать, сколько существует пар (i,j) таких, что Ai AND $Bj = Qk$.

Количество элементов в каждом массиве до 100к (хотя сразу понятно, что разных не более 2^{16}, но всё равно одинаковые надо обрабатывать пусть будут массивы до 100к).

Будем считать ответ сразу для для всех запросов.

Введем в рассмотрение следующие обозначения $f(x, S)$ — множество элементов из последовательности S, которые содержат все биты из x и, возможно, еще какие-то (т.е. такие числа y, что $x \& y = x$).
Заметим, если $a \in f(x, A)$ и $b \in f(x, B)$, то $(a \& b) \& x = x$, т.е. $a \& b$ содержит все те же биты, что и x, опять же, возможно, еще какие-то дополнительно.

Пусть $g(x)$ — количество пар индексов (i, j) таких, что $A_i \& B_j = x$, несложно видеть, что $g(x) = |f(x, A)| * |f(x, B)| - \sum_{y \neq x, y \& x = x} g(y)$. Таким образом, из всех возможных пар индексов мы вычитаем те, для которых $A_i \& B_j > x$.

Пусть B — максимальное количество бит в числах во вводе. Тогда $|f(x, S)|$ может быть вычислено за время $O(|S| + 3^B)$ для всех x (используя перебора всех подмножеств битов x). Величины $g(x)$ могут быть вычислены за время $O(3^B)$, однако в этом случае нужно использовать перебор всех подмножеств битов x, а всех надмножеств.

Общее время работы описанного решения $O(P + M + K + 3^B)$ ($B = 16$).

Задача C. Пчёлы

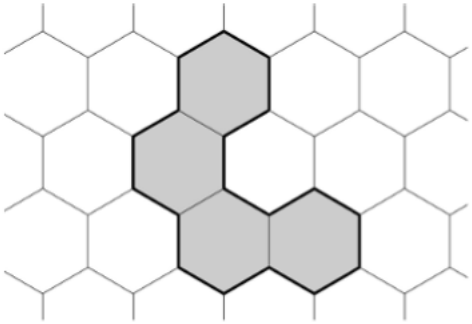
Автор задачи — jakubr (Jakub Radoszewski)
Имя входного файла: stdin
Имя выходного файла: stdout
Ограничение по времени: 5 секунд
Ограничение по памяти: 512 мегабайт

Хобби пчеловода Байтазара — изучение фигур, которые образованы в сотах приносимым пчёлами мёдом. Каждая такая фигура представляет связное множество шестиугольных ячеек со стороной 1, внутри которого нет «дырок».

Более формально, назовём *фигурой без «дырок»* подмножество единичных шестиугольников правильной шестиугольной сетки такое, что любые два единичных шестиугольника, ему принадлежащие, соединены путём, все шестиугольники которого также принадлежат этому множеству и любые два соседних (в смысле этого пути) шестиугольника имеют общую сторону, а любые два единичных шестиугольника, не принадлежащие этому подмножеству, соединены путём, ни одна из клеток которого не принадлежит этому множеству и любые два соседних (в смысле этого шестиугольника имеют общую сторону.

Байтазар хочет составить полный список возможных значений периметра и количества ячеек в подобных шаблонах. Он попросил Вас написать программу, которая по двум заданным числам n и p выясняет, существует ли фигура без «дырок», состоящая из n ячеек, периметр которой равен p.

Например, на приведённой ниже картинке $n = 4$, а $p = 18$.



Формат входного файла. Первая строка ввода содержит количество тестовых примеров T ($1 \le T \le 1000$). Каждая из последующих T строк за тестовый пример и содержит два целых числа n и p ($1 \le n, p \le 1000$) — требуемые площадь и периметр фигуры.

Формат выходного файла. Для каждого тестового примера в порядке их следования во входном файле выведите ответ в отдельной строке. В случае, если фигур с заданными параметрами не существует, выведите «NO». В противном случае выведите строку из p символов, составле букв «L» и «R», задающую обход вдоль периметра фигуры (по часовой стрелке или против неё), при этом «L» обозначает левый поворот, а правый.

Если корректных ответов несколько, выведите любой из них.

stdin	stdout
3 4 18 3 18 3 12	RLRRRRLRLRLRRRRLRL NO LRRRLRRRLRRR

Решение задачи C

Задача заключается в том, чтобы построить многоугольник на шестиугольной решётке,
состоящий из n единичных клеток и p сторон, или же убедиться, что многоугольника с данными параметрами не существует.

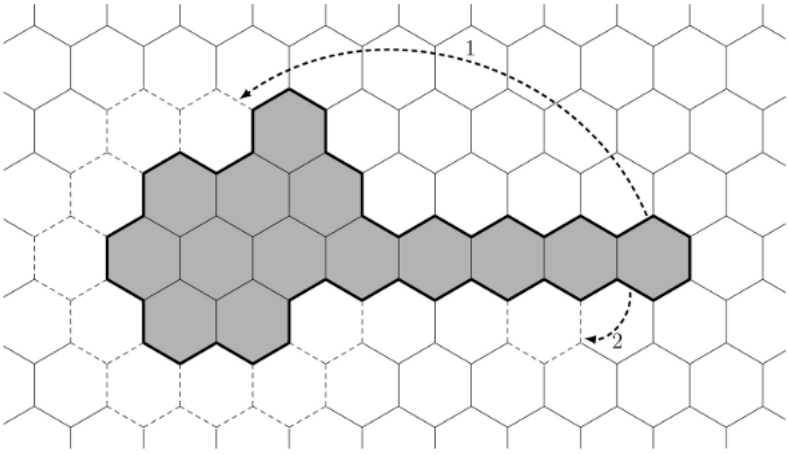
Зафиксируем количество клеток n и изучим возможные значение p такие, что многоугольник с p сторонами существует. Очевидно, что p дол быть чётным.

Также легко заметить, что p не может быть больше, чем $4n + 2$ (периметр многоугольника, состоящего из n клеток, идущих подряд и гранич по стороне; назовём такой многоугольник *цепочкой*).

С другой стороны, из общих соображений ясно, что многоугольник минимального периметра представляет собой некое подобие большого шестиугольника. Формально его периметр задаётся формулой $2\lceil\sqrt{12n-3}\rceil$.

Оказывается, что данные ограничения являются не только необходимыми, но и достаточными.

Алгоритм построения выглядит следующим образом: мы начинаем с цепочки, состоящей из n клеток и имеющего максимальный периметр. Д пока периметр нашего многоугольника больше, чем p, мы удаляем одну клетку из «хвоста» многоугольника и добавляем эту клетку в «голов достраивая большой шестиугольник слой за слоем (см. стрелку 1 на иллюстрации).



Данный ход уменьшает периметр на 2, 4 или 6, так что в конечном итоге нам придётся увеличивать периметр на 2 или 4, перемещая некото клетки (например, перемещение клетки из «хвоста» в соответствии со стрелкой 2 на рисунке).

При некоторых реализациях потребуется отдельно разобрать случаи для малых значений n, а также случаи, когда периметр близок к

минимальному значению для данного *n*.

Задача D. Анюта и кубики

Автор задачи и разбора — Роман Удовиченко.

Имя входного файла: stdin
Имя выходного файла: stdout
Ограничение по времени: 2 секунд
Ограничение по памяти: 512 мегабайт

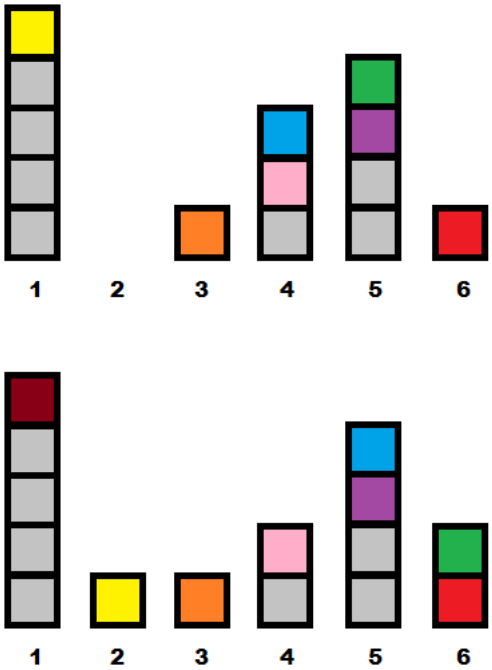
Маленькая Анюта любит играть с кубиками. У неё есть *N* позиций, расположенных в ряд и пронумерованных от 1 до *N*, в которых она любит строить башни из кубиков.
Башня — это несколько кубиков, лежащих один на другом. Высота башни — это количество кубиков в ней. Номер башни соответствует номе позиции, в которой эта башня строится.

Изначально все позиции пусты. Каждую минуту происходит следующее.

1. Анюта смотрит последовательно на позиции с номерами от *N* — 1 до 1.
2. Если в какой-то позиции *i* есть кубики, и высота башни больше либо равна высоте башни в позиции *i* + 1, Анюта перекладывает один к из *i*-й позиции в (*i* + 1)-ю.
3. После того, как Анюта просмотрела все позиции, она добавляет один кубик в башню в позиции с номером 1.

Пример. Пусть количество позиций *N* = 6, а высоты башен равны 5 0 1 3 4 1 (нумерация позиций от 1 до *N* слева направо). Тогда за следую минуту произойдет следующее.

1. Анюта посмотрит на позицию 5: высота башни в ней равна 4, в то время как высота башни в следующей позиции равна 1. Анюта перен один кубик из позиции 5 в позицию 6. Высоты башен станут равными 5 0 1 3 3 2.
2. Анюта посмотрит на позицию 4: высота башни в ней равна 3, а высота башни в следующей позиции равна 3. Анюта перенесёт один ку позиции 4 в позицию 5. Высоты башен станут равными 5 0 1 2 4 2.
3. Из позиции 3 в позицию 4 Анюта не будет переносить кубик, потому что высота башни в позиции 4 больше, чем высота башни в позиц
4. Из позиции 2 в позицию 3 Анюта не будет переность кубик, потому что в позиции 2 нет кубиков.
5. Наконец, из позиции 1 в позицию 2 Анюта перенесёт один кубик. Высоты башен станут равными 4 1 1 2 4 2.
6. После этих действий Анюта добавит один кубик в позицию номер 1, и высоты башен станут равными 5 1 1 2 4 2.



Изменение высот башен в течение минуты в описанном выше примере

Однажды Анюте надоело играть, и она решила посчитать, сколько у неё будет кубиков на определённых позициях после некоторого числа м
Поскольку Анюта ещё маленькая, а кубиков так много, она обратилась за помощью к вам.
Более формально, вам требуется ответить на Q запросов вида «сколько будет в сумме кубиков в башнях с номерами от S_i до F_i включительно
B_i минут». Запросы независимы: в каждом запросе считается, что изначально все позиции пусты, а после этого проходит i>B_i минут.

Формат входного файла. В первой строке ввода находятся два целых числа *N*и *Q* ($2 \le N \le 10^{18}$, $1 \le Q \le 100\,000$) — количество позиций для
и количество запросов соответственно. В каждой из следующих *Q*строк находится три целых числа S_i, F_i и B_i ($1 \le S_i \le F_i \le N$, $1 \le B_i \le 10^{18}$)
начальная и конечная позиции для запроса и количество минут, прошедшее после начала Анютиных действий.

Формат выходного файла. Выведите Q строк, по одной на каждый запрос. В каждой строке выведите одно число — ответ на запрос. Отвечай
запросы в порядке их следования во вводе.

Примеры

stdin	stdout
10 3 1 1 1 1 10 100 10 10 11	1 100 2
1000000000000 2 1 1234 412412314 100000000000 999999999999 1000000000010	1234 900000000006

Решение задачи D

Для начала следует понять, как движутся кубики по позициям. Рассмотрим, к примеру, случай, когда *N* = 5.

В первые пять минут кубики будут <<ползти>> один за одним от позиции номер 1 до позиции *N*, пока в каждой позиции не появится по одн
кубику и высоты башен будут равны [1, 1, 1, 1, 1].
На шестой минуте кубик из четвёртой позиции перейдёт в пятую, кубики левее подвинутся на одну позицию вправо, и в позиции номер 1 по
новый кубик: [1, 1, 1, 1, 2].
На седьмой минуте произойдёт перемещение кубиков только из позиций (3,2,1) в позиции (4,3,2) соответственно, и высоты станут [1, 1, 1,

На восьмой минуте передвинутся кубики из позиций 1-4: [1, 1, 1, 2, 3].

Продолжая моделирование, получим:
- 9-я минута: [1, 1, 2, 2, 3]
- 10-я минута: [1, 1, 2, 3, 3]
- 11-я минута: [1, 1, 2, 3, 4]

Спустя ещё 4 минуты высоты башен станут равны [1, 2, 3, 4, 5]. Спустя ещё 5 минут высоты станут [2, 3, 4, 5, 6], спустя ещё 5~ — [3, 4, 5,

Таким образом, нетрудно заметить следующую закономерность:

1. В первые N минут высоты башен равны [1, ..., 1, 0, ..., 0];
2. Далее высоты башен имеют вид либо [1, ..., 1, 2, 3, ..., x], либо [1, ..., 1, 2, 3, ..., $k - 1$, k, k, $k + 1$, ..., x];
3. Спустя $\frac{N \cdot (N+1)}{2}$ минут после начала высоты башен имеют вид [1, 2, 3, ..., $N - 1$, N];
4. После этого по одному кубику добавляется во все башни по кругу.

Учитывая вышесказанное, для ответа на каждый запрос нужно сначала вычислить, какой вид сейчас имеют высоты башен, а затем применит несколько несложных математических формул, связанных с арифметическими прогрессиями.

Задача E. Математическая магия

Автор задачи и рабора — Олег Христенко
Имя входного файла: stdin
Имя выходного файла: stdout
Ограничение по времени: 2 секунд
Ограничение по памяти: 512 мегабайт

Как-то раз Вася готовился к олимпиаде по математике и заснул. Васе приснился Дэвид Блейн, говорящий следующее:

«Сейчас я продемонстрирую математическую магию. Вот смотрите: я выписываю на доске в ряд N целых положительных чисел, больших еди и меньших 2^{63}. Некоторые из этих чисел повторяются: хотя бы одно из них равно какому-то другому числу в ряду. Под ними я выписываю н ряд из $K < N$ целых положительных чисел, больших единицы и меньших 2^{63}, некоторые из них тоже повторяются. Перемножаем числа в обо наборах… произведение чисел из второго набора ровно на единицу больше, чем произведение чисел из первого набора.

А сейчас я убираю некоторые повторения в верхнем ряду чисел: множество чисел, присутствующих в верхнем ряду, осталось таким же, но х один элемент ряда я стираю. Делаю то же самое с нижним рядом… вновь ни одно число ряда не пропало совсем, просто некоторых чисел ста меньше. Снова перемножаем наборы… и видим, что произведение чисел из второго набора снова ровно на единицу больше, чем произведен чисел из первого набора. Магия!»

Вася проснулся, но смог вспомнить только N. Помогите ему восстановить фокус.

Формат входного файла. В первой строке входного файла задано одно целое число N ($2 \le N \le 100$) — количество чисел в первоначально выписанном Дэвидом Блейном наборе.

Формат выходного файла. В первой строке выведите первоначально выписанный Дэвидом Блейном набор, во второй — тот, который был нап снизу, в третьей и четвёртой — варианты этих наборов после произведённых Дэвидом вычёркиваний. Числа внутри каждого набора должны отсортированы по неубыванию, должны быть больше единицы и не должны превышать $2^{63} - 1$.

Если наборов, удовлетворяющих условию и ограничениям, не существует, выведите вместо них одно слово «Impossible». Если таких наборо несколько, выведите любой из них.

stdin	stdout
2	Impossible
3	2 2 2 3 3 2 3

Разбор задачи E

Для $N = 2$ решения не существует (так как $K<N$, то в нижней части выписано одно число, и вычеркнуть кратные не получается). Для $N = 3$ решение — 8, 9, 2, 3 ({222}, {33}, {2}, {3}).Можно показать, что данное решение единственно. Для $N = 4$ решения не существует. Доказательство приводится в конце разбора.

Построим решение для $N \geq 5$.

Для любого $N > 4$ берём числа $2(2^{N-3}-1)$и $2^{N-1}(2^{N-3}-1)$ в качестве b и a.
Очевидно, что они представимы как произведение некоторого количества двоек и числа $2^{N-3}-1$, a представлено в виде произведения N сомножителей, и a делится на b. Проверим условия на a+1 и b+1. $a + 1 = (2^{N-2} - 2) + 1 = 2^{N-2} - 1$, а $b + 1 = 2^{N-2}(2^{N-2} - 2) + 1 = (2^{N-2})^2 - 2 * 2^{N-2} + 1 = (2^{N-2} - 1)^2$, то есть получаем, что $a + 1 = (b + 1)^2$, и тем самым получаем наборы из двух и одного числа соответственно, удовлетворяющие условию задачи.

Для $5 \leq N \leq 65$ сгенерированный таким образом ответ подходит автоматически.

Далее заметим, что если набор $q_1,..., q_k$ является решением задачи для $N = k$, и $q_k = r_1 \cdot r_2$, то набор $q_1,..., q_{k-1}, r_1, r_2$ является решением зад для $N=k+1$ (просто представляем в разложении числа b число $\{q_k\}$ как произведение чисел r_1 и r_2). Аналогично разложение одного из множителей на произведение двух меньших переводит пару (a+1), также в корректную пару.

Поэтому будем пытаться раскладывать $2^{N-3} - 1$ на множители, не превосходящие $2^{63} - 1$, тем самым получая из той же формулы решения д чисел от $N + 1$ до N +D — 1, где D — число простых делителей числа $2^{N-3} - 1$. В случае $N > 65$ требуется ещё разложить $2^{N-2} - 1$ на множи не превосходящие $2^{63} - 1$.

Если N имеет остаток 0 по модулю 6, то $N - 2$ делится на 2, $N - 3$ делится на 3, и тем самым $2^{N-2} - 1$ представляется как $(2^{(N-2)/2} - 1)(2^{(N-2)/2} + 1)$, а $2^{N-3} - 1$ — как $(2^{(N-3)/3} - 1)(2^{(2N-6)/3} + 2^{(N-3)/3} + 1)$. Легко увидеть, что при $N \leq 95$ оба сомножителя не прево $2^{63} - 1$. А раз так, то и все последующие их разложения также удовлетворяют требованиям задачи.

Если N имеет остаток 5 по модулю 6, то $N - 2$ делится на 3, а $N - 3$ — на 2, соответственно, используем аналогичное разложение.

Таким образом, начинаем с $N = 63$ (2^{60} -1 должно раскладываться на большое количество множителей, а 2^{61}-1 требованию задачи удовлетв а затем перебираем все N, не превосходящие 96 и дающие 0 или 5 при делении на 6, делая первое разложение $2^{N-3} - 1$ на множители опис в предыдущем абзаце способом (после чего раскладывая каждый из множителей <<до упора>>) и, также в соответствии с этим способом, раскладывая $2^{N-2} - 1$ на два множителя. При этом, количество множителей K в наборе $a + 1$ будет равно четырём (квадрат числа переходит квадрат произведения двух чисел), тем самым условие $K < N$ выполняется.В результате перебора получаем, что:

- При $N = 63$ получаем $D = 13$ множителей, то есть генерируем ответы для $64 \leq N \leq 75$.
- При $N = 71$ получаем $D = 7$ множителей, генерируем ответы для N от 72 до 77 включительно.
- При $N = 77$ получаем $D = 4$ множителя, генерируем ответы для N от 78 до 81 включительно.
- При $N = 78$ получаем $D = 7$ множителей, генерируем ответы для N от 79 до 84 включительно.
- При $N = 83$ получаем $D = 11$ множителей, генерируем ответы для N от 84 до 93 включительно.
- При $N = 90$ получаем $D = 6$ множителей, генерируем ответы для N от 91 до 95 включительно.
- При $N = 95$ получаем $D = 9$ множителей, генерируем ответы для N от 96 до 100 (даже до 104) включительно.

Тем самым алгоритм находит решения для всех $N \leq 100$.

Доказательство невозможности решения при $N = 4$

Так как $N = 4$, то, по условию задачи, $a + 1$ раскладывается на три множителя или менее. Чтобы было, что вычёркивать, как минимум два из должны повторяться. Также, по условию задачи, a делится на b, то есть $a = xb$, $x > 1$.

Сначала разберём все варианты набора для $a + 1$ (и соответствующего набора для $b + 1$):

1. u^2, $\sim u$;
2. u^3, $\sim u$;
3. uv^2, uv;
4. u^3, u^2.

В случае 3 $a + 1 = uv^2$. Тогда $b + 1 = uv$; $xb + 1 = uv^2$, вычитаем, $uv(v - 1) = b(x - 1)$. Так как uv и b взаимно просты, то $v - 1$ делится на как $v > 1$, то $v - 1 = yb \geq b$nobr>, $v \geq b + 1$. Так как $u > 1$, то $uv > v \geq b +$. Противоречие.

Так как в данном рассуждении не использовалось то, что $u \neq v$, то случай $a + 1 = u^3$ и $b + 1 = u^2$ рассмотрен автоматически.

Остались случаи 1 (u^2,$\sim u$) и 2 (u^3,$\sim u$). Для них рассмотрим варианты набора из четырёх множителей для a и соответствующего им набора д

I. $a = x^2yz$, $b = xyz$;

II. $a = x^2y^2$, $b = xy^2$;

III. $a = x^3y$, $b = x^2y$;

IV. $a = x^4$, $b = x^3$;

V. a = x^2y^2, $b = xy$;

VI. $a = x^4$, $b = x^2$;

VII. $a = x^3y$, $b = xy$;

VIII. $a = x^4$, $b = x$.

Варианты II-IV получаются из варианта I путём подстановки $z = y$, $z = x$ и $z = y = x$ соответственно. Аналогично, вариант VI есть частный сл варианта V.

В *варианте I* заменим yz на t.

1. $x^2t + 1 = v^2$, $xt + 1 = v$, откуда $x^2t + 1 = x^2t^2 + 2xt + 1$. Так как все слагаемые положительны и $x^2t^2 > x^2t$ (так как $t > 1$), то правая часть больше левой.

2. $x^2t + 1 = v^3$, $xt + 1 = v$. Аналогичной подстановкой и раскрытием скобок получаем $x^2t = x^3t^3 + 3x^2t^2 + 3xt$, все слагаемые положительн $< x^3t^3$.

Вариант V

1. $x^2y^2 + 1 = v^2$. Сразу не бывает при целых x, y, $v > 1$.

2. $x^2y^2 + 1 = v^3$, $xy + 1 = v$. При подстановке и приведении подобных получим, что сумма $(xy)^3 + 2(xy)^2 + 3xy$ равна 0, что невозможно при 1.

Вариант VII

1. $x^3y + 1 = v^2$, $xy + 1 = v$, перепишем как $x^3y + 1 = x^2y^2 + 2xy + 1$ и затем как $xy(x^2 - xy) = 2$. Получается, что 2 делится на xy, чего при $y > 1$ не может быть.

2. $x^3y + 1 = v^3$, $xy + 1 = v$. Подставляем, получаем, что $x^3y = x^3y^3 + \ldots$ (многоточием обозначены заведомо положительные числа), что при невозможно.

Вариант VIII

Пусть $x^4 + 1$ делится на $x + 1$. Но $x^4 + 1 = (x + 1)(x - 1)(x^2 + 1) + 2$, то есть получаем, что 2 делится на $x + 1$. Противоречие с тем, что $x >$

Мы разобрали все случаи и везде пришли к противоречию, то есть при $N = 4$ решения не существует.

Задача F. Зебра Гиппо и полоски

Автор задачи и рабора — rng58 (Makoto Soejima)
Имя входного файла: stdin
Имя выходного файла: stdout
Ограничение по времени: 2 секунд
Ограничение по памяти: 512 мегабайт

У зебры Гиппо есть 10^{18} полосок, занумерованных последовательными целыми числами от 0 до $10^{18} - 1$. В момент времени 0 только полоск номером $5 \cdot 10^{17}$ является чёрной, остальные являются белыми. В момент времени t ($t > 0$) цвет полосок удовлетворяет следующим правила

- Если x-я полоска в момент времени $t - 1$ является чёрной, то она будет чёрной и в момент времени t.
- Если в момент времени $t - 1$ все полоски, расположенные на расстоянии не более, чем D от x-й полоски, являются белыми (иначе гово для каждого целого неотрицательного y, меньшего 10^{18} и удовлетворяющего неравенству $|x - y| \le D$, y-я полоска в момент времени t является белой), то в момент времени t x-я полоска также будет белой.
- В ином случае цвет x-й полоски в момент времени t выбирается произвольным образом.

Вычислите количество возможных различных раскрасок зебры по модулю $10^9 + 7$ в момент времени t. Две раскраски зебры считаются различными, если цвет хотя бы одной полоски различается.

Формат входного файла. В первой и единственной строке входа заданы два целых числа t и D ($1 \le t \le 10^9$, $1 \le D \le 50$), разделённые пробе

Формат выходного файла. Выведите одно целое число — остаток от деления количества возможных раскрасок в момент времени t на $10^9 +$

Пример

stdin	stdout
5 1	36
2 3	1849

Разбор задачи F

Из соображений симметрии очевидно, что количество раскрасок зебры является квадратом количества раскрасок её правой половины. Так ч
далее будем рассматривать только правую половину зебры. Для упрощения перенумеруем полоски следующим образом:
500 000 000 000 000 000+*x*-ю занумеруем как *x*-ю.

Для начала решим более простую задачу: дана раскраска, требуется вычислить минимальное время *t*, по истечении которого данная раскра
будет достижима. Эта задача решается следующим алгоритмом:

Изначально указатель находится на полоске с номером 0, а множество *A* пусто.

Цикл:
- Если указатель находится на самой правой полоске, выходим из цикла.
- Пусть *x* — текущая позиция указателя.
- Найдём наибольшее *d* ($0 < d \le D$) такое, что (*x*+*d*)-я полоска чёрная. Если такого *d* нет, алгоритм завершается неудачно.
- Добавляем *d* в множество *A* и передвигаем указатель на (*x*+*d*)-ю полоску.

Конец цикла.

Если этот алгоритм успешно завершается, минимальное время *t* равно количеству элементов множества *A*.

Пусть $f(a_1,a_2,\ldots, a_k)$ — количество раскрасок, для которых вышеприведённый алгоритм возвращает
$A = \{a_1, a_2,\ldots, a_k\}$.

Алгоритм возвращает $\{a_1, a_2,\ldots, a_k\}$ для данной раскраски тогда и только тогда, когда:

- 0-я, a_1-я, (a_1+a_2)-я и так далее полоски — чёрные.
- Для каждого *i* полоски между (a_1+...+a_i+1)-й и (a_1+...+a_{i-1}+*D*)-й являются белыми.

Отсюда получаем $f(a_1, a_2,\ldots, a_k) = 2^{a_1-1 + a_1+a_2-D-1 + a_2+a_3-D-1 +\ldots}$.

В первоначальной задаче нам нужно посчитать количество раскрасок, то есть сумму $f(a_1, a_2,\ldots, a_k)$ (при $k \le T$).

Пусть $dp[t][a]$ ($t \le T$, $a \le D$) — сумма $f(a_1, a_2,\ldots, a_{t-1}, a)$.

Получаем следующее рекуррентное соотношение для динамического программирования:

$$dp[t][a] = \sum_{b=0}^{D} (dp[t-1][b] \cdot 2^{a+b-D-1})$$

Ответом к задаче является $1 + dp[1][D] +\ldots + dp[T][D]$, тем самым сложность задачи равна $O(TD)$.

Чтобы ускорить решение, можно использовать возведение матрицы в степень. Пусть v_t — вектор ($dp[t][0],\ldots, dp[t][D]$). Из рекуррентного соотношения мы знаем, что $v_{t+1} = v_t \cdot A$ для всех *t* (*A* — некоторая матрица размерности
(D+1) × (D+1)).

Ответ к задаче будет равен последнему элементу $v_0 \cdot (A + A^2 +\ldots + A^T)$, увеличенному на единицу.

Сложность этого алгоритма — $O(D^3 \cdot \log T)$.

Конкуренция на чемпионате была очень высокой. В финале участвовали коллеги Геннадия Короткевича по команде НИУ ИТМО чемпионы АС 2013 Нияз Нигматуллин (Niyaz Nigmatullin) и Михаил Кевер (mikhailOK), победитель Facebook Hacker Cup, Google Code Jam и Яндекс.Алгорит 2011, один из самых титулованных спортивных программистов мира Пётр Митричев (Petr), вице-чемпион VK Cup 2012 Ван Циньши (s-quark) Китая и другие сильные спортивные программисты.

В финал вышли представители России, Беларуси, Украины, Польши, Словакии, Тайваня, Китая и Южной Кореи. Среди них — разработчики и Google, «Вконтакте» и Facebook.

яндекс, яндекс.алгоритм, спортивное программирование, разбор задач

+71 94,6k 384

Автор: @EgorK

рейтинг
Яндекс 795,02
Как мы делаем Яндекс
Github

ПОХОЖИЕ ПУБЛИКАЦИИ

4 августа 2014 в 13:51
Разбор финальных задач Яндекс.Алгоритма 2014
+67 52,8k 270 32

12 июля 2013 в 12:08
Разбор задач финала чемпионата мира про программированию ACM ICPC 2013
+107 88,5k 384 13

21 июня 2013 в 14:42
Яндекс.Алгоритм 2013: новая платформа Яндекс.Contest и правила TCM/Time
+37 11,6k 43 20

Комментарии (30)

nSnayp 22 августа 2013 в 19:21 #

Круто, круто)))

Sollaes 22 августа 2013 в 19:41 #

ИТМО молодцы, с победой!

Strizhevich 22 августа 2013 в 19:55 #

Интересно, Геннадий Короткевич есть в англоязычной википедии —http://en.wikipedia.org/wiki/Gennady_Korotkevich. А в русскоязычной нет, обидно.

Informatik 23 августа 2013 в 07:28

А чем он отличается от тысячи других чемпионов по всему миру?

yeputons 23 августа 2013 в 09:40

Например, выдающимися результатами на международной олимпиаде школьников по информатике (IOI): 7 участий (5-11 классы), 6 золотых медалей, из три абсолютных чемпионства, одно — с полным решением всех предложенных задач. Вообще говоря, само попадание туда в пятом классе уже о многом говорит.

SkidanovAlex 23 августа 2013 в 10:28

Например тем, что он возглавляет рейтинг ТопКодер. Начиная с Дэрека Кисмана на вершине рейтинга TopCoder было всего пять разных человек, так что о «тысячах» речи не идет.
Вообще, сложно сомневаться в том, что Гена сейчас самый сильный спортивный программист на планете.

vladon 27 августа 2013 в 09:21

Википедия создаётся нашими общими руками. Хотите статью — напишите.

Хотя в русской 99% вероятность, что удалят.

nickme 22 августа 2013 в 21:46

Хм, а первая задача разве за линейное время не решается? В каждом слове ищем первую слева букву (начиная при этом со второй буквы, т.к. префикс не дол быть пустым), большую первой буквы в следующем слове и обрезаем слово по найденной букве (не включая ее), от последнего слова оставляем только перву На всех приведенных примерах это вроде работает…

sabio 22 августа 2013 в 22:51

Но если очередная буква совпадает с первой буквой следующего слова, то надо переходить к сравнению следующей буквы со второй буквой следующего с т.д. В итоге, получим квадратичную сложность.

Например:
Unnnnnnnnnnnnnniversity Nnnnnnnnnnnnichego Nnnnnnnnnnnne Delanya
Или менее тривиальный вариант:
Uninininininininininiversity Ninininininininininichego Ninininininininini Delanya

nickme 22 августа 2013 в 23:19

Понял, спасибо!

Ogoun 22 августа 2013 в 21:47

Фамилия у Анюты, наверное, Рутковская.

Prometheus 22 августа 2013 в 23:17

Да, меня всегда забавляло, когда математические задачки описывались в гуманитарном русле :)

Это просто шикарно:
«По удивительному стечению обстоятельств, как раз сегодня родители Анюты принесли с работы два набора чисел. Папа принёс с работы набор A, состоящий чисел, а мама — набор B, состоящий из M чисел. Все числа в наборах были целые, неотрицательные, а также не превосходящие 65535. Кроме того, числа в на были пронумерованы, начиная с 1.»

Прочитав данный абзац — возникает попутная задача на дедукцию, а именно — где работают родители Анюты, что с работы они принесли наборы с пронумерованными числами :)))))

spacediver 23 августа 2013 в 02:24 *(комментарий был изменён)*

Ага, математикам зарплату наборами чисел выдали. А дети давай их побитово умножать вместо ужина.

Mario_Z 23 августа 2013 в 08:32

В банке работают вероятно.

hombre 23 августа 2013 в 09:50

майнерами работают)

НЛО прилетело и опубликовало эту надпись здесь

BalinTomsk 11 апреля 2014 в 06:01

когда я работал на литейно-механическом заводе в 1985 году, мы делали корпуса для подшибников, корпус состоял двух половинок, которые были уникаль

чтобы не потерять половинки мне поставили задачу выбивать цифробуквенные сочетания для каждой пары.
через месяц я уже истощил свою фантазию.
тогда о гуидах я еще тогда не знал

j2k6 23 августа 2013 в 09:06 +1

Спасибо, почувствовал себя дебилом :)

kirias 23 августа 2013 в 09:32 +

> так как строка «uniofw» является лексикографически меньшей, чем строки, соответствующие другим варинтам, к примеру, «uow»

Вот тут-то меня и накрыло.

intet 23 августа 2013 в 10:15 +

Всегда интересовало, а насколько широк спектр задач где надо применять подобные методы подхода. По ощущениям в прикладном программировании какой-бизнес системы большая часть времени уходит на сборку функционала из стандартных кубиков. Внутри которых и скрыты все подобные хитрые алгоритмы ск уже внутри них, а сам программист должен только прописывать связи.
Просто создается ощущение, что спортивное программирование это вещь вообще мало связанная с реальными задачами.

efimovgk 23 августа 2013 в 10:53

С реальными бизнес-задачами — да, наверно. Но эти «кубики» кто-то должен написать. Так что тут смотря с какой стороны посмотреть.

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь

Mrrl 27 августа 2013 в 09:32

Всегда есть опасность, что какого-нибудь кубика не хватит. Или он будет делать не совсем то, что надо. Или потребует очень громоздких подпорок. И тогда придётся или его реализовывать, или оправдываться перед начальством тем, что «гугл такого не знает».

XAMelleOH 23 августа 2013 в 11:29 +

Геннадию можно заочно давать первое место и не дразнить других участников.

А вообще, парень в 18 лет не хило бабло поднимает на проге — каждое соревнование это $10-20к.

iCune 23 августа 2013 в 14:00

Объясните пожалуйста, почему |f(x, S)| может быть вычислено за время O(|S| + 3^B)? Откуда 3^B?

Zalina 23 августа 2013 в 20:59 +

Передаю вам ответ от Лёши Толстикова:

> Каждое число *y* нам нужно отметить в ячейках *x* таких, что *x* & *y* = *x* (т.е. биты *x* — это подмножество битов *y*). Различных *y* у нас не более 2^B, а суммарно количество подходящих *x* не более 3^B (об этом написано здесь: e-maxx.ru/algo/all_submasks).

petrovnn 23 августа 2013 в 19:32 +

надо было назвать статью «как Яндекс ищет новых струдников»

AKuvichko 25 августа 2013 в 13:53

Возможно, где-то упустил по тексту, но на каких языках могли участники писать программы?

EgorK 25 августа 2013 в 15:54 *(комментарий был изменён)* +

Java (6/7), gcc (c/c++/c++0x, x32/x64), python (2/3), Delphi/fp

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.

САМОЕ ЧИТАЕМОЕ Разра

Сейчас | Сутки | Неделя | Месяц

Анализ шифровальщика Wana Decrypt0r 2.0

+96 37,3k 124 225

О том, как в Instagram отключили сборщик мусора Python и начали жить

+26 6,4k 59 3

Как защищаться от атаки вируса-шифровальщика «WannaCry»

+2 9,9k 38 22

Создание JPEG из ниоткуда

+35 7,6k 30 2

Познакомимся с WannaCry поближе

+45 38,2k 87 61

ИНТЕРЕСНЫЕ ПУБЛИКАЦИИ

Гигер и фантастические технологии мира Чужих GT

1,5k 6 2

Колония. Глава 11: Рассказ доктора GT

333 0 3

Физики впервые установили контрфактическое квантовое соединение GT

1,7k 8 6

Джефф Безос: будущее бизнеса — машинное обучение, концентрация на интересах клиентов и быстрое принятие решений GT

742 2 0

Да будет фильм с Xamarin.Forms

731 10 0