Казанский государственный университет

Факультет вычислительной математики и кибернетики

Кафедра системного анализа и информационных технологий

Пшеничный П.В., Тагиров Р.Р.

Анализ и построение вычислительных алгоритмов (на примерах олимпиадных задач по программированию)

Методическое пособие

Казань, 2009 г

Данное методическое пособие предназначено для студентов, изучающих процесс разработки алгоритмов и программ для решения сложных задач по программированию.

Большинство задач предлагались участниками олимпиад фирмы ICL, студентам и школьникам г.Казани, а также взяты с соответствующих сайтов, посвященных олимпиадам по программированию. Тексты всех задач упрощены и исключены ограничения на память и время. Приведены идеи решения и примеры анализа нескольких конкретных задач из разных разделов, разработка нескольких вариантов алгоритмов для каждой из них, анализ их сложности. Для многих задач приведены тексты программ на языке Си++.

Все задачи разбиты на условные темы, хотя некоторые из них могут относиться к нескольким темам одновременно

Общие замечания

Они касаются определенных моментов (ситуаций) в решениях, которые могут привести к ошибкам или нарушениям ограничений.

1. Точность вычислений с вещественными аргументами может быть разной при использовании и вещественных и целых значений в одном арифметическом выражении – лучше сразу использовать целые, уже преобразованные в вещественные числа.

2. Размер вычисленных целых значений
Иногда число в результате не умещается в стандартный длинный целый тип (long int), хотя отдельные операнды выражения будут умещаться. Нужно анализировать (предварительно оценивать) возможные размеры результата. Может быть, достаточно хранить только существенную часть значения или в отдельных случаях вычислить результат по частям, с помощью 2-х или 3-х переменных (если число только чуть-чуть превышает стандартный максимум).

3. Длинная арифметика
Если получается один результат или массив результатов, каждый элемент которого может быть нестандартного размера, возможно, придется использовать длинную арифметику, т.е. хранить цифры числа в специальном массиве и использовать нестандартные операции над такими числами или функции для сложения, присваивания и т.д. Но предварительно придется оценить размеры нестандартности, т.е. сколько цифр хранить для каждого такого сверхдлинного числа.

4. Временная сложность задачи.
Часто существуют простые переборные варианты решений задач, поставленных в общем виде, но они могут потребовать значительного количества процессорного времени, что приведет к нарушению ограничений по времени выполнения программы. Придется искать варианты, уменьшающие перебор за счет использования каких-то дополнительных условий (ограничений) задачи.

5. Размеры массивов и ограничения по памяти.
Некоторые задачи имеют простые решения с использованием массивов, но размеры задачи, могут оказаться такими большими, что невозможно будет уложиться в

ограничения по памяти. В таких случаях придется, возможно, использовать вычисление значений и их хранение по частям в элементах массива меньшего размера или может вовсе обойтись без массивов. В некоторых переборных алгоритмах использование массивов умеренного размера может существенно ускорить процесс поиска, если в таком массиве хранить (накапливать) промежуточные результаты.

6. Варианты
В некоторых задачах наибольшую трудность может представлять не сложность самого алгоритма, а аккуратный перебор и учет всех возможных разных вариантов или допустимых ситуаций, например, комбинации в игре покер.

7. Крайние случаи
Для каждой задачи составляются тесты для проверки решений на крайних (редких) случаях исходных данных или возможных ограничений, а также для самых больших допустимых данных. Очень часто частные случаи не решаются общим методом, но их легко можно проанализировать отдельно.

8. Недопустимые варианты
Во многих задачах необходимо проверять получаемые решения на допустимость, т.е. что они удовлетворяют всем условиям задачи, если конечно, вы не уверены на 100% в том, что ваша программа генерирует только правильные решения. Это, обычно, касается задач с перебором вариантов.

9. Некоторые особенности задач, которые надо учитывать при решении
- в каких средах надо разрабатывать программы - Visual C++, Delphi, C#, Java.
- ограничения по времени (1-2 сек) и памяти (32-64Мбайт)
- исходные данные вводятся из текстового файла или с клавиатуры, а результаты выводятся в выходной файл или на экран;
- считается, что все данные заданы правильно
- максимальное значение целых и длинных целых чисел без знака = 2^32
- если возможно несколько решений, вывести любое из них.

Не всегда указывается формат входных и выходных файлов, т.к. не предусматривается автоматический анализ ответов с помощью contest-системы.

Краткий список тем задач

1 Графы, деревья
2 Лабиринты, матрицы
3 Оптимальные пути
4 Вычисления и оценки, выведение формул
5 Точность вычислений
6 Последовательные вычисления
7 Геометрия
8 Перестановки

Тема 1 Графы, деревья

S. Из школьного курса физики (Заочный ICL-2008)
Последовательно-параллельной называется цепь, построенная по одному из следующих правил (где R - сопротивление, а A1,A2...An - удовлетворяют определению последовательно-параллельной цепи):
1 сопротивление - это ППЦ
2 последовательное соединение нескольких ППЦ - это ППЦ
3 параллельное соединение нескольких ППЦ - это ППЦ
При этом эту цепь можно представить в виде графа, каждое ребро которого имеет некоторое сопротивление, вершины – это узлы, а входы схемы находятся в 1-й и K-ой вершинах. Требуется определить сопротивление такой цепи.
Во входном файле задано описание последовательно-параллельной цепи в следующем формате:
В начале файла записаны количество вершин в графе K и число ребер графа M (2 <= K <= 50, 1 <= M <= 10000).
Далее идет M строк, каждая из которых содержит три числа – описание ребра: номера вершин, которые оно соединяет и его сопротивление (положительное вещественное число).
В выходной файл вывести сопротивление цепи с точностью до 4-х знаков после запятой.

R. Логика решает все (Заочный ICL-2008)
Схемой из функциональных элементов (СФЭ) называется схема с несколькими входами, на которые подается либо 0, либо 1 и одним выходом, который принимает значение в зависимости от значений входов. СФЭ можно представить в виде ориентированного графа без циклов, каждой вершине которого приписано одно из значений:
1. Входная переменная. В такие вершины не входит ни одного ребра. На все ребра, выходящие из этой вершины, подается значение данной входной переменной.
2. Логическое И. В такие вершины входит по два ребра. На все ребра, выходящие из этой вершины, подается результат логического И входных значений.
3. Логическое ИЛИ (аналогично).
4. Логическое НЕ имеет один вход. На все выходы подается отрицание входа.
5. Выход схемы – во всей схеме только одна такая вершина. Она имеет одно входящее ребро и ни одного выходящего. Значение, которое подается на входящее в нее ребро, считается выходом всей схемы.
Ваша задача – определить значения выходов схемы для всех возможных значений входов
Во входном файле задано описание СФЭ.
Первая строка содержит два числа: N - количество вершин графа и M - количество входных переменных (2<= N<= 500,1<= M<= 15). Далее следует N строк, описывающих вершины: сначала идет число – тип вершины (см. список выше), а затем для вершин типа 2,3 два числа, для вершин типа 4,5 одно число – номера вершин, к выходам которых подключены входы данной вершины.
Вершины с номерами от 1 до M – это входные переменные, а описанная СФЭ корректна.
В выходной файл нужно вывести 2^M чисел – значения выхода схемы на всех наборах входных переменных.

Q. Любителям детективов (Заочный ICL-2007)
Локальная компьютерная сеть имеет структуру дерева и все узлы пронумерованы числами, начиная с 1 (корневой). Каждый узел, кроме корневого, является дочерним для некоторого узла с меньшим номером. Найти узел с наибольшим номером, который является общим родительским для двух заданных узлов.
Первая строка входного файла содержит натуральное число n (n < 30000) — количество узлов. Во второй строке записаны номера двух заданных узлов. В третьей строке находятся n —1 натуральных чисел, i-е из них не больше i и задает номер узла, к которому подсоединен узел i + 1.
В выходной файл вывести одно число — номер искомого узла.

B. Округлая планарность (Заочный ICL-2008)
Задан неориентированный граф. Потребуем, чтобы все его вершины располагались в точках некоторой окружности, а ребра соответствовали хордам окружности. Нужно проверить — возможно ли расположить вершины таким образом, чтобы ребра не пересекались.
В первой строке входного файла содержится число вершин графа N (1<= N<= 50) и число ребер M. В последующих M строках заданы ребра, каждое номерами двух вершин.
В первой строке выходного файла должно находиться слово Yes, если граф можно расположить заданным образом, или No, если нет.
Если граф можно расположить, то во второй строке выдать последовательность номеров вершин в порядке обхода по окружности.

K. Трудный выбор (Заочный ICL-2006)
Дана матрица размера N x N. Требуется выбрать N чисел из нее так, чтобы в каждой строке и в каждом столбце было выбрано ровно одно, и их сумма была минимальна.
Во входном файле записано N (0 < N < 401) и далее N*N целых чисел (составляющих матрицу), не превосходящие по модулю 20000.

L. Железная логика (Заочный ВМК-2007)
Имеется описание N фактов. Некоторые факты за определенное время позволяют установить какие-то другие факты. Определить список фактов, из которых за наименьшее время можно вывести все остальные факты.
Формат входных данных. В первой строке записано целое число N (0 < N <= 7500) – число фактов. Далее в каждой из N строк содержится описание факта. Описание начинается с целого числа Ai – времени на определение факта (0 <= Ai <= 100). Далее идет неотрицательное целое число Pi – число связей данного факта с другими. После этого следуют Pi различных целых чисел от 1 до N – номера фактов, которые определяются, если определён данный факт. Сумма всех Pi не превосходит 15000.
Формат выходных данных. В первой строке единственное число Q – наименьшее время, требуемое на определение всех фактов (если невозможно, то вывести -1). Если решение есть, то во второй строке вывести количество фактов, которые нужно определить пользователю, и в третьей строке – список номеров этих фактов через пробел. Если решений несколько, вывести любое из них.

Тема 2 Лабиринты, матрицы

C. Шарик в лабиринте (Заочный этап турнира ICL-2006)
В игре «Шарик в лабиринте» необходимо провести шарик по пластмассовому лабиринту. Для этого лабиринт можно наклонять, отчего шарик начинает катиться по проходу с ускорением g = 9,811 м/с^2. Шарик катится строго по прямой параллельно сторонам лабиринта. На поворотах шарик мгновенно и полностью останавливается. Лабиринт имеет размер N x N см (1 <= N <= 10) и высоту 1 см и состоит из клеток — кубиков 1 x 1 x 1 см. Каждый такой кубик является либо проходом, либо стенкой. Ровно один из кубиков помечен как выход из лабиринта. Требуется определить минимальное время, за которое шарик можно перекатить из левого верхнего угла лабиринта к выходу.
Примечание: путь, пройденный равноускоренно движущимся из состояния покоя телом за время t, равен g*t^2/2.
Описание лабиринта состоит из N строк. Каждая строка описания состоит из N цифр от 0 до 2, разделённых пробелами. Здесь 0 соответствует пустой клетке, 1 — стенке, 2 — выходу. При этом цифра 2 встречается во всем описании ровно один раз, а первая цифра первой строки не равна 1.

K. Кинг Конг (очный этап турнира ICL-2006)
До крыши Эмпайр Стейт Билдинг Джек должен преодолеть N-1 лестничных пролетов, причем известно, за какое время можно пробежать каждый из них. Также в здании находится лифт; для любого пролета между соседними этажами известно, какое время требуется лифту, чтобы проехать его. Чтобы войти в лифт, выйти из него или нажать кнопку требуется 0 секунд. Также за 0 секунд можно зайти в лифт, нажать кнопку нужного этажа, тут же выйти из него и побежать наверх. Можно останавливаться на этажах, чтобы подождать лифт. Изначально лифт находится на первом этаже, там же, где стоит Джек. Написать программу для нахождения наименьшего времени, которое нужно для подъема до самого верхнего этажа.
Первая строка входного файла содержит целое число N – количество этажей Эмпайр Стейт Билдинг (1 ≤ N ≤ 1000). Вторая строка содержит N-1 целых чисел x_i (1 ≤ x_i ≤ 1000); x_i – время в минутах, которое потребуется Джеку, чтобы преодолеть i-й пролет пешком. Третья строка содержит N-1 целых чисел y_i (1 ≤ y_i ≤ 1000); y_i – время в минутах, которое потребуется лифту, чтобы преодолеть i-й пролет

F. Три клетки (Заочный ICL-2006)
На клетчатом поле закрашены три клетки. Требуется закрасить дополнительно наименьшее количество клеток таким образом, чтобы все закрашенные клетки образовали 4-связную фигуру. Т. е. из каждой закрашенной клетки можно было бы добраться в любую другую, двигаясь только по закрашенным клеткам, соседним друг с другом по вертикали либо по горизонтали.
Входные данные содержат целые числа x1, y1, x2, y2, x3, y3 — координаты трёх различных закрашенных клеток, разделённые пробелами и/или символами перевода строки. Числа находятся в диапазоне от 0 до 100.

L. Лабиринты и монстры. (Заочный ICL-2007)
Два монстра помещаются в лабиринт размера NxN. Хороший монстр помещается в левый верхний угол лабиринта, а плохой - в правый нижний. Каждая клетка лабиринта содержит либо цифру 0 либо 1. Монстры могут ходить только по клеткам,

в которых записана цифра 1. Хороший монстр может двигаться только вправо или вниз. Плохой монстр может двигаться только влево или вверх. Каждый ход оба монстра обязаны сделать ровно 1 шаг, т.е. они не могут стоять на месте. Вам необходимо определить такой путь для каждого монстра, чтобы они встретились.
В первой строке файла записано число N (1<=N<=10) - размер лабиринта. В следующих N строках записан сам лабиринт, по N символов в каждой строке.

A. Кубики - не ролики... (Заочный ICL-2008)
В матрице N на M (1<= N,M<= 100000) в клетки записаны неотрицательные целые числа. Для каждого столбца и для каждой строки вычислены и запомнены максимумы. После этого матрица обнулена. Требуется определить максимальную и минимальную сумму всех чисел, которое могло быть в матрице.
В первой строке входного файла записаны числа N и M (1<= N,M<= 100000).
Во второй строке записано N чисел, где i-ое число обозначает максимальное из чисел, стоявших в i-ой строке матрицы.
В третьей строке записано M чисел, где j-ое число обозначает максимальное из чисел, стоявших в j-ом столбце матрицы. Числа во второй и третьей строках не превосходят 1000.
Если не существует конструкции, соответствующей входным данным, вывести в выходной файл строку ‘no solutions’. Иначе в первую строку вывести ‘OK’, во вторую — максимальное и минимальное число. Если N, M<= 100 то в последующих строках вывести пример матрицы для максимальной суммы и пример для минимальной суммы.

L. Ход конем (Заочный ICL-2008)
На шахматной доске стоит конь. На эту клетку положили 2^K золотых монет. На те клетки, на которые сможет дойти конь за 1 ход, положили 2^(K-1) золотых монет. И вообще, если с клетки, на которой стоял конь, можно дойти до некоторой клетки самое меньшее за P<= K ходов, то на нее положат 2^(K-P) золотых монет. Найти клетку для коня так, чтобы сумма всех монет была наибольшей и не превосходила M.
Примечание. Напоминаем, что шахматная доска имеет форму квадрата 8x8 клеток, столбцы называются латинскими буквами от a до h, а строки – цифрами от 1 до 8, клетка имеет название в виде пары буква-цифра, в зависимости от того, на пересечении какого столбца и какой строки она находится.
На первой строке находятся числа K(0<= K<= 25) и M (1<= M<= 10^9).
На первой строке выходного файла вывести число N - количество монет (или 0). Если N > 0 , на второй строке вывести в любом порядке, но без повторений, все возможные клетки, в которых мог стоять коня. Разделяйте имена клеток пробелами.

D. Еще одна игра (Заочный ICL-2008)
Для этой игры используется обычная шахматная доска 8x8. Играют двое. У каждого игрока от 1 до 12 фишек, у одного черного, а у другого белого цвета. Фишка игрока может ходить по вертикали или горизонтали на расстояние, равное числу фишек обоих игроков, стоящих на вертикали или горизонтали, по которой производится ход, включая фишку, делающую ход. Фишка при ходе может прыгать через фишки своего цвета, но не через фишки другого цвета. Приземляться фишка должна на свободное поле или на поле, занятое фишкой другого цвета (в этом случае фишка противника снимается с доски). Напишите программу, подсчитывающую число вариантов хода у каждого игрока в заданной позиции.

Во входном файле содержится 8 строк по 8 символов 'X', 'O' или '.'. Символом '.' (точка) обозначается свободная клетка, 'X' - клетка, занятая черной фишкой, а 'O' - клетка, занятая белой фишкой.
В первой строке выходного файла вывести два целых числа через пробел - количество вариантов хода у игрока, владеющего белыми фишками, и у игрока, владеющего черными фишками.

J. Развертка куба (Заочный ICL-2007)
Дан квадрат, расчерченный горизонтальными и вертикальными линиями так, что они делят большой квадрат на 36 маленьких. Ровно 6 из этих маленьких квадратов закрашены в черный цвет. Требуется определить, можно ли фигуру, образованную закрашенными квадратами, вырезать и сложить по прочерченным линиям так, чтобы она образовала куб в трехмерном пространстве, т.е. является ли эта фигура разверткой куба.
При этом большой квадрат является «замкнутым» либо по вертикали, либо по горизонтали. Например, в случае замкнутости по вертикали это означает, что каждый маленький квадрат в первом столбце является соседним для соответствующего ему квадрата в последнем столбце.
Входные данные состоят из шести строк, по шесть чисел 0 или 1 в каждой, обозначающих цвет соответствующего маленького квадрата (0 – белый, 1 – черный). Числа в строке разделены пробелами.
Если нарисованная фигура является разверткой куба, выведите единственное слово “YES” (без кавычек). В противоположном случае выведите слово “NO”.

Тема 3 Оптимальные пути

R. Игра (Заочный ICL-2007)
Герой прыгает по платформам, которые висят в воздухе. Он должен перебраться от одного края экрана до другого, при этом на то, чтобы перепрыгнуть с платформы на соседнюю, у Героя уходит $|y_2 — y_1|$ единиц Энергии, где y_1 и y_2 - высоты, на которых расположены эти платформы. Кроме того, у Героя есть Суперприем, который позволяет перескочить через платформу, но на это затрачивается $3 * |y_3 — y_1|$ единиц Энергии. Известны координаты всех платформ в порядке от левого края до правого. Найти, какое минимальное количество Энергии потребуется Герою, чтобы добраться с первой платформы до последней?
В первой строке входного файла записано количество платформ n (1 <= n <= 30000). Вторая строка содержит n натуральных чисел, не превосходящих 30000 — высоты, на которых располагаются платформы.

O. Перекати-кубик (Заочный ICL-2008)
На доске NxN в клетке (1,1), расположенной в юго-западном углу доски, находится игральный кубик с точками на каждой грани. На нижней грани кубика изображена одна точка, на верхней – шесть, на западной грани – пять точек, на восточной – две, на южной – четыре, на северной – три точки. За один ход разрешается перекантовать кубик на соседнюю (по стороне) клетку доски. При этом начисляется штрафной балл в размере числа точек на новой нижней грани кубика. Вам требуется доставить кубик в клетку с координатами (i,j), набрав наименьший штрафной балл.<\p>

Во входном файле записаны числа N, i и j. Размерность доски не превосходит 20.
В первую строку выходного файла вывести минимальный штрафной балл. Далее вывести путь, заданный направлением кантования кубика: N – на север, S – на юг, W – на запад, E – на восток.

Тема 4 Вычисления и оценки, выведение формул

C. Новый лицей (Заочный ICL-2008)
Нужно построить новое здание лицея в любой точке города так, чтобы суммарное расстояние, которое проезжают ученики от своих домов до лицея, было минимально. План города является целочисленной сеткой с единичным шагом. Школьник, добираясь до лицея, может двигаться только по сторонам этой сетки.
Написать программу, которая определит координаты постройки нового здания.
Первая строка входного файла содержит число N – количество учеников в лицее (1<= N<= 500). Каждая из последующих N строк содержит координаты дома школьника — два целых числа из диапазона [–32767, 32767], разделенных пробелом.
Программа должна вывести в выходной файл координаты места для постройки лицея — два целых числа, записанных через пробел.

G. Марсианин Василий (Заочный 2008)
Вася прибыл на Марс вечерним космолетом в 0 часов, 0 минут и 0 секунд по земному времени и именно в этот момент марсианское время было таким же! В марсианских сутках N часов, в каждом часе M минут, а в каждой минуте K секунд. Марсианские стрелочные часы выглядят точно как земные, только земная часовая стрелка делает за сутки два оборота в сутки, а марсианская часовая стрелка оборачивается один раз.
Через некоторое время (прошло не более суток) Вася посмотрел на свои земные часы – они показывали время в формате HH:MM:SS (часы:минуты:секунды). Вася решил немедленно установить марсианские часы на правильное время, но ему нужно помочь перевести текущее время с земного на марсианское.
Точно известно, что земная секунда равна марсианской.
В первой строке входного файла текущее земное время в формате HH:MM:SS
Во второй строке 3 натуральных числа N,M,K не превышающих 100.
В выходной файл необходимо вывести 3 числа a,b,c (0<=a,b,c<360) – углы отклонения часовой, минутной и секундной стрелок соответственно в градусах с точностью до пятого знака после десятичной точки. Отсчет углов ведется по часовой стрелке. Марсианское направление движения часовой стрелки совпадает с земным.

A. Скобки на посту (Заочный ICL-2007)
Даны вещественные числа a1..aN, между которыми расставлены знаки арифметических операций (+, -, *). Требуется расставить скобки, чтобы полученное значение полученного выражения было максимально.
Во входном файле задано исходное выражение – строка, не превышающая 255 символов (может содержать пробелы), числа - вещественные.

F. MechWarrior (Заочный ICL-2007)
На спутнике надо установить наиболее мощный комплект орудий, чтобы он имел максимальную силу выстрела в секунду (Дж/сек), но спутник не может поднять на борт более N кг вооружения, а его реактор производит не более M кВт энергии. Каждое орудие имеет следующие характеристики:

Вес (кг) , Потребляемая мощность (кВт), Скорострельность (выстрел/сек), Сила выстрела (Дж).
Первая строка входных данных содержит целые числа N и M, разделенные пробелом (1<=n,m<=100).
Вторая строка содержит целое число Z - количество моделей вооружения, причем можно установить несколько экземпляров одной модели (1<=Z<=1000).
Следующие Z строк содержат описания орудий, по одному на строке в следующем формате: W P A D, где
W - вес, целое число (0<=W<=100)
P - мощность, целое число (0<=P<=100)
A - скорострельность, действительное число (0<=A<=100)
D - Сила, действительное число (0<=D<=1000), причем гарантируется, что для любого орудия P+W>0.

H. Последовательности из 0 и 1. (Заочный ICL-2007)
Требуется подсчитать количество последовательностей длины N (1<=N<=45), состоящих из 0 и 1, в которых никакие две единицы не стоят рядом.

B. Банкноты (Пробный ICL-2006)
После того, как была отпечатана партия из N(1 <= N <= 1000000) банкнот достоинством в миллиард тугриков, выяснилось, что одна банкнота не напечаталась. Каждая из банкнот имеет уникальный номер от 1 до N. По заданному списку номеров отпечатавшихся банкнот необходимо выяснить номер недостающей.

K. Легонькая задачка (Заочный ICL-2007)
Найдите наиболее длинную арифметических прогрессию из простых чисел в заданном отрезке. Во входном файле записаны два целых числа 1 <= L <= R <= 65536 – границы отрезка.

D. Слишком вложенные скобки (Заочный ICL-2006)
Дана правильная последовательность из круглых скобок длиной от 2 до 10000 символов. Требуется удалить из неё все скобки, находящиеся на глубине вложенности N и более (1 <= N <= 5000). Например, в последовательности (()((()(()))))(((()))) выделены скобки, вложенные на 3 и более уровней.

G. Связка брусков (Заочный ICL-2006)
Два длинных бруска прямоугольного сечения обвязывают веревкой. Требуется расположить бруски так, чтобы длина веревки была минимальной. Вам даны длины сторон сечения каждого бруска a_1, b_1 и a_2, b_2. Требуется определить минимальную длину веревки.

J. Лесенки (Заочный ICL-2007)
Лесенкой называется набор кубиков, в котором каждый более верхний слой содержит кубиков меньше, чем предыдущий. Требуется подсчитать число лесенок, которое можно построить из N кубиков.
Во входном файле записано число N (1<=N<=100).

P. Шифровка. (Заочный ICL-2007)

Используется следующий шифр: буква 'A' будет записываться как 1, 'B' – 2, и так далее до 'Z' – 26. Но если зашифровать слово 'BEAN' (боб), оно будет закодировано как 25114. Но это сообщение можно расшифровать несколькими способами. Кроме 'BEAN' получаются слова 'BEAAD', 'YAAD', 'YAN', 'YKD' и 'BEKD'. Написать программу, которая подсчитывает число возможных расшифровок некоторого сообщения, закодированного этим шифром.
Входной файл содержит строку с шифровкой. Шифровка корректна, не начинаются с цифры 0 и может содержать до 1000 цифр.

Тема 5 Точность вычислений

D. Числа в вершинах (Заочный ICL-2007)
В неориентированном графе без кратных ребер и петель требуется расставить в вершинах числа так, чтобы если вершины соединены ребром, то числа имели общий делитель, а если нет — то общего делителя не было бы.
В первой строке записано число N (0<N<7) — количество вершин в графе. В последующих N строках записана матрица смежности.

C. Посчитай-ка (Заочный ICL-2007)
Во входном файле задано длинное число до 1000 знаков, требуется посчитать корень N степени (ближайшее целое, не превосходящее точного значения). N<=2000.

E. Простые факториалы (Заочный ICL-2007)
«Простой факториал» - произведение всех простых чисел, не превосходящих данное простое число. Дана разность двух простых факториалов (не более 5000 цифр). Написать программу, находящую сами два простых числа.

L. Странная игра (Заочный ICL-2007)
Дано целое число N (2 <= N <= 100000), a и b (0 <= a,b < N). Отправляясь от числа a, надо получить число b за наименьшее число ходов. За один ход разрешается от числа x перейти к одному из чисел (x + 1) mod N, (x^2+1) mod N или (x^3+1) mod N. Написать программу, которая определяет искомое наименьшее число ходов и последовательность чисел, которые при этом получаются (включая начальное a и конечное b).

H. Последние цифры (Заочный ICL-2006)
Дано N (1 <= N <= 20) целых чисел a_1, a_2, …, a_N в диапазоне от 1 до 10000. Требуется найти две последние цифры числа, определяющего количество натуральных делителей произведения $a_1 * a_2 * … * a_N$. Если число делителей меньше 10, то вывести это число без лидирующего нуля.

N. Тени исчезают в полдень (Заочный ICL-2008)
В центре парка в точке с координатами (0,0) поставили очень яркий фонарь, высота которого L, в результате чего деревья стали отбрасывать на землю тени. Требуется посчитать суммарную длину теней от всех деревьев.
Замечания:
Никакое дерево не растет в точке (0,0) и в тени другого. Все деревья растут в различных точках. Некоторые деревья могут иметь одинаковую высоту, но все они ниже фонаря.

Во входном файле записано натуральное число N (1<= N<= 1000) — количество деревьев и целое число L — высота фонаря (2<= L<= 10^6). Далее идет N троек целых чисел x, y и z, задающих соответственно координаты и высоту деревьев (–10^5<= x<= 10^5, –10^5<= y<= 10^5, 1<= z<= L/2).
В выходной файл нужно поместить одно число – суммарную длину теней с точностью 3 знака после десятичной точки.

Тема 6 Последовательные вычисления

A. Сокращение одночленов (Заочный ICL-2006)
Одночлен — это выражение, состоящее из однобуквенных переменных с операциями умножения и возведения в целочисленную степень. Именами переменных являются малые латинские буквы. Умножение обозначается символом '*' (ASCII 42), возведение в степень — символом '^' (ASCII 94). Показатель степени состоит из одной десятичной цифры от 1 до 9. Примеры одночленов: t, a*b*c^2, y*d^1*y^9.
Требуется по данным N + M одночленам построить дробь, равную произведению первых N одночленов, делённому на произведение оставшихся M одночленов. При этом:
Числитель и знаменатель дроби должны быть одночленами.
Переменная не должна встречаться в дроби более одного раза (т. е. дробь необходимо сократить).
Степени переменных должны быть целыми числами, большими или равными 2.
В числителе и знаменателе переменные должны быть отсортированы по алфавиту.
В первой строке содержатся числа N и M (1 <= N, M <= 999), разделённые пробелами.
Следующие N + M строк содержат по одному одночлену каждая. Длина каждой строки не превосходит 100 символов.

E. Школьный субботник (Заочный ICL-2006)
В субботнике участвуют школьники всех S школ города, в каждой из которых обучалось по 50 классов, состоящих из 30 учеников каждый.
Для уборки U улиц были сформированы ровно U бригад, каждая из которых изначально состояла из одного класса и имела порядковый номер от 1 до 50*S. Впоследствии часть бригад была разделена на бригады меньшего размера, а часть — наоборот, объединена в более крупные.
Деление бригад на более мелкие производилось примерно поровну, т. е. в случае деления бригады с порядковым номером B из C человек на P частей, число школьников в i-й части определялось по формуле: $[\frac{i \cdot C}{P}] - [\frac{(i-1) \cdot C}{P}]$. (Здесь квадратные скобки обозначают округление до ближайшего целого вниз). При этом новым бригадам присваивались номера от B до B + P – 1, а номера бригад большие B увеличивались на P – 1.
Для уборки длинных улиц формировались более крупные бригады, объединявшие в своем составе несколько исходных бригад. Новая бригада получала наименьший из номеров объединенных бригад, после чего номера остальных бригад уменьшались

так, чтобы в итоге все бригады были пронумерованы последовательно, а порядок предшествования при этом не изменился.
После окончания субботника оказалось, что лучше всех убрана N-я улица, и требуется определить, сколько было школьников в этой бригаде.
В первой строке находятся числа S, K и N, разделённые пробелами. При этом S (1 <= S <= 100) — количество школ, K (1 <= K <= 5000) — количество операций над бригадами, N (1 <= N <= U) — номер самой чистой улицы. В следующих K описываются операции над бригадами в следующем формате:
0 P B — бригада с номером B была разделена на P бригад (2 <= P <= 10)
или
1 Q B_1 B_2 … B_Q — из Q бригад с номерами B_1 B_2 … B_Q сформирована новая бригада; $B_i < B_j$ если i < j.
На каждом этапе число групп не превышает 10000.

C. Привал (Тренировочный ICL-2005)
Путник двигался t_1 часов со скоростью v_1, t_2 часов со скоростью v_2, ..., t_n часов со скоростью v_n. За какое время время он одолел первую половину пути (после чего запланировал привал)?
Первая строка содержит единственное число N - количество участков пути. Следующие N строк содержат по два числа t_i и v_i, разделенных пробелом. Все числа в файле натуральные и не превышают 100.

M. Перенаправление номеров (Заочный ICL-2007)
Необходимо определить, куда в итоге дозвонится человек, позвонивший по определенному телефону, если разрешается перенаправлять звонки на другие номера.
В первой строке входного файла записано количество клиентов телефонной компании N (0<=N<=1000) В последующих N строках для каждого клиента записан номер клиента, на которого перенаправляются его звонки. Если номер, куда перенаправляется звонок равен номеру клиента, это значит, что звонок для этого клиента не перенаправляется. В N+2-й строке записано число M (0<=M<=1000) - количество входящих звонков Следующие M строк содержат номера клиентов, к которым поступает звонок.

I. Голевой момент (Заочный ICL-2006)
Необходимо определить, попадет ли мяч после удара в ворота (если ему не преградить путь), возможно, с отскоком от штанги.
Штанги ворот расположены на оси Ox, левая штанга находится в начале координат, правая – в точке (x, 0), где x > 0. Штанги имеют радиус r_1.
Поле лежит в верхней полуплоскости, мяч находится в поле (не на линии ворот) в точке с координатами (x_0, y_0). В момент времени t=0 мячу сообщается постоянная скорость, определяемая вектором w = (u, v).
Считается, что мяч попал в ворота, если он полностью находится за линией ворот, и имеет с ней не более одной общей точки. Радиус мяча известен и равен r_2.
Соударения мяча со штангами происходят по следующим законам:
- угол падения равен углу отражения;
- при соударениях модуль вектора скорости w не меняется.

Единственная строка содержит семь вещественных чисел x, r_1, r_2, x0, y0, u, v, каждое с тремя знаками после десятичной точки. Все числа не превосходят по модулю 100. Числа разделены пробелами.
Гарантируется, что для входных данных выполняются следующие условия:
$1 <= r_1 <= 25$, $12 <= x <= 100$, $1 <= r_2 <= 9.8$, $x - 2*r_1 - 5*2*r_2 >= 0$

Q. Бочки (Турнир ICL-2004)
N одинаковых бочек, открытых сверху, заполнены водой. Уровень воды в бочках может различаться. Бочки соединены трубками: первая со второй, вторая с третьей, третья с четвертой и так далее. Последняя бочка соединена с первой. Вода может свободно протекать по трубкам в обоих направлениях. Все бочки расположены на одинаковой высоте. Концы трубок закреплены в бочках высоте 0 от пола. На каждой трубе расположен один кран. Все краны первоначально закрыты. После открытия крана на какой-либо трубе вода начинает свободно перетекать между бочками по закону сообщающихся сосудов до тех пор, пока не установится на одинаковом уровне. Требуется найти минимальное число кранов, которое нужно открыть, чтобы вода в бочках установилась на одном уровне.
Первая строка содержит целое число N ($3<=N<=1000$) – количество бочек. Во второй строке записано N целых чисел X_i, разделенных пробелами ($1<=X_i<=1000$) – количество литров воды в бочках.

I. Выборы антиподов (Заочный ICL-2008)
Австралийские бюллетени требуют, чтобы избиратели расположили всех кандидатов в порядке предпочтения. Первоначально учитывается только первый кандидат из получившегося списка, и если один из кандидатов набрал более 50% процентов голосов, то он считается избранным. Тем не менее, если ни один из кандидатов не набрал более 50% голосов, то все кандидаты с наименьшим количеством голосов выбывают. Бюллетени, засчитанные в пользу этих кандидатов, засчитываются в пользу следующего невыбывшего кандидата в порядке предпочтения. Этот процесс исключения самых слабых кандидатов и пересчет их бюллетеней в пользу следующего по порядку предпочтения продолжается до тех пор, пока один из кандидатов не наберет 50% голосов или пока у всех кандидатов не окажется одинаковое количество голосов.
Входной файл начинается со строки, содержащей одно целое положительное число $N <= 20$, означающее число кандидатов. Следующие N строк содержат имена кандидатов, каждое до 10 символов длиной. Имя может содержать произвольные печатаемые символы. Далее следуют до 1000 строк, каждая из которых содержит описание бюллетеня. Каждый бюллетень содержит числа от 1 до N в произвольном порядке. Первое число означает предпочтительного кандидата, второе - второго по предпочтимости и т.д.
Выходной файл должен содержать одну строку с именем победившего кандидата, либо несколько строк в произвольном порядке с именами кандидатов, набравших одинаковое количество голосов.

B. Просто спорт (Заочный ВМК-2007)
Написать программу, которая по заданному числу определит наибольшее четное количество простых чисел, сумма которых не превосходит заданное.
Исходные данные - натуральное число A, $1<=A<=100000$.
Результат - количество простых чисел или 0, если задача решений не имеет.

Тема 7 Геометрия

B. Нечетный N-угольник (Заочный ICL-2006)
Выпуклый N-угольник P преобразуется в N-угольник Q путём замены середин сторон исходного многоугольника P на вершины многоугольника Q. Требуется по выпуклому N-угольнику Q, заданному координатами вершин, восстановить координаты вершин исходного N-угольника P.
Входные данные содержат нечётное число вершин N (3 <= N <= 999), за которым следуют целочисленные координаты x_i, y_i вершин многоугольника Q, перечисленные в порядке обхода по часовой стрелке. Значения координат находятся в диапазоне от –20000 до 20000.
Все числа целые и разделены произвольным количеством пробелов и/или символов перевода строки.

C. Пересечение (Пробный тур ICL-2006)
Требуется определить, имеют ли отрезок и прямоугольник общие точки. Во входном файле заданы координаты вершин прямоугольника (в порядке обхода по или против часовой стрелки), затем координаты концов отрезка. Все числа во входном файле вещественные, не превосходят 10000 по модулю.

A. Метро в Казани (Очный тур ICL-2005)
Две точки расположены на границах двух выпуклых многоугольников. Написать программу, определяющую минимально возможную длину пути между двумя точками, не пересекающего эти многоугольники.
Первая строка вводного файла содержит единственное число N ($3 \le N \le 20$)- количество вершин первого многоугольника. В последующих N строках записано по паре целых чисел X, Y ($0 \le X,Y \le 10005$), разделенных пробелом – координаты вершин первого многоугольника, перечисленных в порядке обхода по часовой или против часовой стрелки.
В следующей строке два вещественных числа X_{c1} Y_{c1} ($0 \le X_{c1},Y_{c1} \le 10005$) – координаты первой точки.
В следующей строке число M ($3 \le M \le 20$)- количество вершин второго многоугольника. В последующих M строках записано по паре целых чисел X Y ($0 \le X,Y \le 10005$) – координаты вершин второго многоугольника, перечисленных в порядке обхода по часовой или против часовой стрелки.
В последней строке два вещественных числа X_{c2} Y_{c2} ($0 \le X_{c2},Y_{c2} \le 10005$) - координаты второй точки. Пары чисел разделены пробелом.

A. Квадрат (Тренировочный тур ICL-2005)
Квадрат ABCD задается на плоскости координатами своих вершин. Известны координаты XA, YA, XC, YC диагонально расположенных вершин A и C. Требуется вычислить координаты вершин B и D. Вершины A, B, C и D перечисляются в порядке обхода по часовой стрелке.

Единственная строка содержит четыре вещественных числа XA, YA, XC, YC – координаты вершин A и C квадрата. Координаты не превосходят по модулю 200. Числа разделены пробелами.

B. Охотники за привидениями (Заочный ICL-2007)
На плоскости имеется N красных точек и N синих. Разбить все точки на пары, которые бы соединялись не пересекающимися отрезками. Никакие три точки не лежат на одной прямой.
Составить программу, которая:
Входной файл содержит набор исходных данных следующего вида:
в первой строке набора находится число N - количество красных (N<=150), в следующих 2*N строках заданы позиции на плоскости N красных и N синих точек; каждая позиция занимает отдельную строку и представляет собой пару целых чисел, разделенных пробелами, - координаты позиции на плоскости (абсолютные значения координат не превосходят 32000).

G. Mission Impossible (Заочный ICL-2007)
На полигоне установлены радары с радиусом действия R (0<R<=100000) метров. Полигон представляет собой квадрат размером NxN метров (0<N<=100000 м, целое), юго-западный угол полигона имеет координаты (0; 0), северо-восточный - (N; N). Необходимо найти точку с целыми координатами, так чтобы не попасть под действие радара.
Первая строка входного файла содержит числа N и R, разделенные пробелом.
Вторая строка содержит целое число Z (количество радаров, 0<=Z<=10).
Следующие Z строк содержат пары целых чисел X и Y, разделенные пробелом - координаты радаров (0<=X,Y<=N).

Тема 8 Перестановки

E. По порядку становись (Заочный ICL 2008)
Возьмем все непустые различные подмножества из некоторого набора букв и упорядочим их в алфавитном порядке: сначала буквы внутри подмножеств, а затем сами подмножества. Например, из набора букв AABC получаются следующие подмножества после записи их в алфавитном порядке A, AA, AAB, AABC, AAC, AB, ABC, AC, B, BC, C. По заданной последовательности букв латинского алфавита (до 30) и номеру подмножества в упорядоченном списке требуется найти это подмножество.
Первая строка входного файла содержит последовательность заглавных букв латинского алфавита, из которых формируются подмножества. Вторая строка содержит целое положительное число - номер искомого подмножества в упорядоченной последовательности. Гарантируется, что подмножество с таким номером всегда существует.
Единственная строка выходного файла должна содержать строку, описывающую искомое подмножество.

A. Максимальная тройка (Пробный ICL-2006)
В данном двумерном целочисленном массиве a размером N x N (2 ≤ N ≤ 100) требуется найти три элемента, сумма которых максимальна. При этом первый элемент должен быть соседним по горизонтали или вертикали со вторым, а второй – с третьим. Гарантируется, что для любых трех чисел в матрице их сумма не превосходит 109 (все числа неотрицательны).

Тема Дополнения

E. Лесной Интернет (Заочный ВМК-2007)
Упорядочить по неубыванию список IP-адресов.
Первая строка содержит число N (1<=N<=1000000) жителей Двоичного Леса.
Следующие N строк содержат IP-адреса жителей, по одному в строке.
Отсортированный по неубыванию список IP-адресов, по одному IP-адресу в строке

G. Перепись бобров (Заочный ВМК-2007)
Из двух букв С и К строятся строки одинаковой длины. Они должны быть различны и не должны распадаться на несколько одинаковых подстрок (так, строка СКССКС не допускается, а СКСККК - допускается).
Входные данные - натуральное число N (1<=N<=2500) – длина каждой строки.
Результат – количество различных строк.

D. Шведские карандаши (Заочный ВМК-2007)
Все дома пронумерованы числами от 1 до N. Написать программу для выполнения запросов.
В первой строке два натуральных числа, N (1<=N<50000) – число домов и L (1<=L<=100000) - количество запросов к системе. Далее в L строчках идут запросы следующего вида:
0 A K - в дом с номером A принесли K карандашей;
1 A K - из дома с номером A выкинули K карандашей;
2 A B K - из дома с номером A в дом с номером B перенесли K карандашей;
3 A B - подан запрос на вычисление суммарного количества карандашей в домах, нумерованных от A до B (включая концы)
Первоначально предполагается, что в домах нет ни одного карандаша (и в каждом доме может присутствовать не более 25000 карандашей).

C. Кроличий блокнот (Заочный ВМК-2007)
Телефонным номером называется любая строка из ровно десяти цифр, не прерываемая никакими другими знаками (например, "1111111111" - номер, а "12345678912" и "111-111-111-1" - не номера).
Входная информация - текст, размером не более 100 кб, содержащий не менее одного телефонного номера, оканчивающийся символом конца файла.
Результат - список всех телефонных номеров в том порядке, в каком они встречаются в тексте, причем каждый номер должен находиться в отдельной строке.

Если в тексте телефонный номер входит как подстрока в состав более длинной строки, состоящей только из цифр, то такие номера выписывать не надо.

A. Волки и медведи (Заочный ВМК-2007)
Строятся две геометрические прогрессии - B, 4B, 16B,… из L элементов и A, 2A, 4A,… из K элементов.
Написать программу, которая определит, сколько элементов надо взять, чтобы сумма чисел в первой прогрессии стала по крайней мере вдвое больше суммы чисел во второй.
Исходные данные - числа в следующем порядке : A,K,B,L (0<=A,B<=1000, 1<=K,L<=7), разделенные пробелами.
Результат - единственное число - сколько элементов надо взять или -1, если невозможно.

O. Аккуратная морковка (Заочный ВМК-2007)
Морковка имеет форму прямого кругового конуса высоты H и радиусом основания R.
Написать программу вычисления объемов частей, на которые делится конус плоскостями, параллельными основанию, и проходящими через точки деления высоты конуса на N равных частей.
Входные данные - вещественные числа R, H (1 <= R <= 1000, 1 <= H <= 1000) и целое число N (1 <= N <= 1000).
Результаты - N вещественных чисел - объёмы кусочков, на которые разрезается конус. Объёмы считать и выводить с точностью до пятого знака после десятичной точки.

P. Кубический компас (Заочный ВМК-2007)
Даны коэффициенты некоторого кубического многочлена, такого, что старший коэффициент его равен единице. Если у кубического многочлена найдутся два таких корня, что их полусумма равна третьему корню, то следует искать к северо-западу; в противном случае - к северо-востоку.
Исходные данные - Коэффициенты кубического многочлена a2, a1, a0 (0 <= a0, a1, a2 <= 1 000 000) - сперва коэффициент при второй степени, потом при первой, и, наконец, свободный член.
Результат - единственная строка "NORTHWEST", если нужно идти на северо-запад, и "NORTHEAST" в противном случае (кавычки выводить не следует).

F. Крольчата (Заочный ВМК-2007)
Своё исходное терпение Крольчиха оценивает как S. Далее, она составила потактовый график своей работы. На каждом такте происходит какое-то из двух событий.
Первое событие - 'a'. В этот такт родители приводят в садик ещё одного крольчонка с активностью A^i.
Второе тип событий - 's' - это совершение невинных шалостей. Из-за каждой невинной шалости Крольчиха теряет часть терпения, а именно, B^i.
Если в какой-то такт суммарная активность крольчат станет больше, чем терпение Крольчихи, то ситуация перестанет быть контролируемой. При этом, если дать крольчонку Морковку, то его активность можно считать нулевой (Морковку можно дать сразу же, как только родители привели крольчонка в садик).

Итак, Крольчихе нужно продумать, сколько взять с собой Морковок, кому и сколько их дать.
Входные данные
Первая строка входных данных содержит число S (100 <= S <= 10^8). Далее следует K строк (1 <= K <= 100000), содержащих расписание. Каждая строка имеет один из двух форматов:
a A^i (100 <= A^i <= 1000)
s B^i (1 <= B^i <= 100)
Результат. В первой строке должно быть одно число - минимально необходимое количество Морковок, которые нужно дать крольчатам, чтобы ситуация была контролируемой весь день.
Вторая строка должна содержать K чисел. i-е число равно 0, если на i-м такте никому давать морковку не нужно, либо - если морковку дать нужно - порядковому номеру крольчонка (по появлению в расписании), которому нужно её дать.

J. Лесоповал (Заочный ВМК-2007)
Даны два луча на плоскости, пущенные из начала координат. Необходимо начертить наименьшую дугу окружности, соединяющую эти два луча. Каждый луч характеризуется углом, под которым он пущен. Угол задается в градусах. Угол — целое число; угол изменяется от 0 до 359 градусов.
Во входном файле на одной строке через пробел заданы два числа — углы для этих лучей.
В выходной файл необходимо вывести дугу в формате A B, где A — угол, с которого начинать рисовать дугу, B — угловая длина дуги. 0 <= A < 360, 0 <= B <= 180.

N. Фатальная загадка
Даны натуральные числа N, K, L, M (1 <= K <= N <= 1000,1 <= L <= M <= 1000)
Результат - Наименьшее общее кратное чисел сочетаний из N по K и из L по M.

I. Конструктивная логика
Зафиксируем на множестве переменных некоторый линейный порядок (и запишем его как X1<X2<...<Xk). Занумеруем также биты в двоичном представлении числа следующим образом: первый бит - это крайний справа бит, а номера возрастают справа налево. Тогда каждому набору булевских значений соответствует некоторое K-значное двоичное число, двоичная запись которого означает следующее - i-ый бит равен 1, если Xi - истина, и 0 в противном случае. Назовем это число номером двоичного набора. Таблицей истинности назовем последовательность из 2^K нулей и единиц, где в i-ом символе находится значение логической функции на наборе с номером i-1. Так, для AND таблица истинности - 0001. Стандартным мономом назовем конъюнкцию некоторого числа различных переменных, выписанных по неубыванию (в соответствии с заданным линейным порядком). Номером стандартного монома назовем число, двоичная запись которого по заданному стандартному моному формируется так: в i-ом бите находится 1, если переменная Xi присутствует в стандартном мономе, и 0 в противном случае. Полиномом Жегалкина назовем сумму по модулю 2 некоторого количества различных стандартных мономов. Записью полинома Жегалкина назовем последовательность из 2^K единиц и нулей, где в i-ой позиции стоит 1, если стандартный моном с номером i входит в полином Жегалкина, и 0 в противном случае. Например, для OR (дизъюнкции двух переменных) полином Жегалкина выглядит как xy+x+y, а запись его - как 0111.

Входные данные - Таблица истинности - последовательность нулей и единиц длиной 2^K.
Результат - Запись полинома Жегалкина, такого, что его значение на любом булевом наборе совпадает со значением функции, заданной входной таблицей истинности.

H. Многословный кузен
Есть словарь некоторого языка. и также есть фрагмент текста, который надо расшифровать. Назовем интерпретацией текста его разбиение на слова из заданного языка; назовем длиной интерпретации количество слов (не обязательно различных) в интерпретации. Если для текста не существует интерпретации, назовем его неинтерпретируемым.
Найти максимальную длину возможной интерпретации или убедиться, что текст неинтерпретируем.
Входные данные - Первая строка, длиной не более 500 символов - интерпретируемый текст (состоящий только из строчных букв латинского алфавита). Во второй строке содержится число Q - количество слов в словаре. В следующих Q строках содержатся слова языка (также только из строчных букв латинского алфавита), гарантировано, что ни одно из них не превосходит по длине интерпретируемого текста.
Результат - максимально возможная длина интерпретации или 0, если текст неинтерпретируем.

K. Шифровка
Текст, который нужно зашифровать (далее - открытый текст), кодируется в последовательность цифр 0 и 1 следующим образом: каждому символу соответствует свой код, символы кодируются независимо друг от друга, т.е. если тексту X соответствует код Y, а символу A - код B, то тексту XA соответствует код YB.
Символу " " (пробел) соответствует код 00000
Символу a - 00001
Символу b - 00010
Символу c - 00011
...
Символу z - 11011
Выбирается ключ шифрования, одинаковый для всех блоков. Ключом является пара чисел от 0 до 65536, далее обозначаемых как K_1 и K_2.
Код открытого текста делится на блоки по 3 символа. Каждый блок рассматривается как двоичная запись числа от 0 до 32767. Далее каждый блок зашифровывается независимо от других.
Пусть мы зашифровываем блок P (т.е. три кода символа образовали двоичную запись числа P). После зашифровки блок P будет числом C, которое получается по следующему закону: $C = K_1 * P + K_2$. Здесь все умножения и сложения производятся по модулю 65537.
Это число и записывается в зашифрованную строку.
Содержимое входного файла, зашифрованного так, как указано в задаче. Зашифрованные данные записаны в одну строку блоками ровно по 5 цифр, разделенных пробелами. Если блок состоит менее, чем из 5 цифр, он дополняется ведущими нулями. Гарантируется, что после расшифровки вложение состоит только из строчных латинских букв a - z и пробелов. Размер вложения после расшифровки не превышает 1000 символов и кратен 3.

Выходные данные - в первой строке записан ключ в виде трех чисел от 0 до 65536, дополненных нулями до длины 5 символов. Числа разделены пробелами. Во второй строке записано расшифрованное вложение, записанное в одну строчку. Расшифрованный текст не должен содержать символов, отличных от строчных латинских букв и пробелов. Расшифрованный текст должен содержать подстроку "spherical liion" (без кавычек).

M. Кролик в виртуальности
В игре есть N различных локаций. Для перехода между локациями используют ключи. В начале игрок находится в первой локации и имеет ровно M ключей. Каждый i-й ключ открывает портал между локациями A_i и B_i и используется только один раз. При неправильном использовании ключ просто пропадает. Определить, в каких локациях может оказаться игрок после использования всех ключей.
Входные данные - первая строка содержит числа N и M (2<= N <= 1000, 1 <= M <= 1000). Каждая из следующих M строк содержит числа A_i и B_i – номера локаций, между которыми свиток открывает портал.
Выходные данные - список номеров локаций, в которых может оказаться игрок, в порядке возрастания.

J. Крестословица (Заочный ICL-2008)
Задана сетка размером m X n, каждая клетка которой содержит заглавную или прописную латинскую букву. Для каждого слова из заданного списка требуется определить позицию в сетке, в которой находится это слово.
Слово в сетке может располагаться только по прямой непрерывной линии букв. Регистр букв при совпадениях не учитывается. Слово может располагаться в любом из восьми диагональных, горизонтальных или вертикальных направлений.
Первая строка входного файла содержит целые положительные числа m и n, разделенные пробелом (1 <= m, n <= 50). Следующие m строк содержат ровно по n букв каждая; они представляют собой сетку букв, в которой необходимо искать слова. Следующая строка содержит одно целое число k (1 <= k <= 20). Следующие k строк входных данных содержат список слов для поиска, по одному в строке. Слова состоят только из прописных и строчных латинских букв.
Для каждого слова из списка вывести два целых числа, разделив их пробелом, представляющие собой его положение в сетке. Первое число - это номер строки, где расположена первая буква слова (строки во входном файле нумеруются сверху вниз). Второе число в строке обозначает столбец, где находится первая буква слова (столбцы нумеруются слева направо). Если слово в сетке встречается более одного раза, выведите расположение самого верхнего варианта. Если под это условие подходит несколько случаев расположения слова, выведите самый левый из этих случаев. Все слова встречаются в сетке по меньшей мере один раз.

M. Спички детям не игрушка (Заочный ICL-2008)
Петя придумал новую игру. На стол кладется кучка из N спичек, и затем Петя с Ваней по очереди берут спички из кучки. Первым берет Петя, ему разрешается взять от 1 до K спичек. Затем игрок может взять любое количество спичек, не более чем на 1 превышающее то количество, которое взял игрок перед ним (можно взять меньше или столько же, но обязательно хотя бы одну). Например, если N=10, K=5, то на первом ходу Петя может взять 1, 2, 3, 4 или 5 спичек, если Петя возьмет 3, то на

следующем ходу Ваня может взять 1, 2, 3 или 4, и если Ваня возьмет 1, то Петя затем может взять 1 или 2, и т. д. Проигрывает тот, кто возьмет последнюю спичку. Теперь Петя хочет рассчитать какое количество спичек он должен взять на первом ходу, чтобы выиграть при любой игре Вани. Помогите ему.
На первой строке входного файла находятся числа N и K, разделенные пробелом. (1 <= K <= N <= 200).
Вывести в выходной файл все такие X, что, взяв на первом ходу X спичек, Петя выиграет. Если таких X не существует, выведите в выходной файл единственное число - 0. Числа следует разделять пробелами.

F. День выборов (Заочный ICL-2008)
В президентских выборах во второй тур прошли два кандидата – Лапкин и Палкин, причем Палкин в первом туре набрал больший процент голосов равный N. Во втором туре выигрывает тот кандидат, который выиграл в первом туре, но при условии, что он наберет ровно M процентов голосов, в противном случае выигрывает его противник по второму туру. M вычисляется следующим образом $M=(N!+100)_2*10 + (N!+100)_3$, где X_i – i-тая цифра числа в десятичной записи. Сколько процентов голосов надо набрать Лапкину, чтобы стать президентом деревни. Известно, что сумма процентов голосов отданных за обоих кандидатов всегда равна 100, но при этом никому никогда не удавалось и не удастся набрать 100% голосов.
Входные данные - натуральное число N (0<N<100)
Выходные данные - натуральное число - ответ на поставленную задачу.

Литература

1. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М., «Мир», 1979 г.
2. Гасфилд Д. Строки, деревья и последовательности в алгоритмах. СПб., «Невский диалект». 2003г.
3. Долинский М.С. Алгоритмизация и программирование на TURBO PASCAL. От простых до олимпиадных задач. Учебное пособие. СПб., «Питер», 2005 г.
4. Долинский М.С. Решение сложных и олимпиадных задач по программированию. Учебное пособие. СПб., «Питер», 2006 г.
5. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 2-е изд., М., Вильямс, 2007 г.
6. Липский В.В. Комбинаторика для программистов. М., «Мир», 1988 г.
7. Меньшиков Ф. Олимпиадные задачи по программированию, СПб., Питер, 2006, 2007 г.
8. Московские олимпиады по информатике. Под ред. Андреевой Е.В., Гуровица В.М., Матюхина В.А. М., «МЦНМО», 2006 г.
9. Новиков Ф.А. Дискретная математика для программистов. Учебник для ВУЗов. 2-е изд. СПб., «Питер», 2004 г.

10. Окулов С. Программирование в алгоритмах. М., «Бином. Лаборатория знаний», 2004 г.
11. Порублёв И.Н., Ставровский А.Б. Алгоритмы и программы. Решение олимпиадных задач. М., СПб, К., «Диалектика», 2007 г.
12. Потопахин В.В. TURBO PASCAL: Решение сложных задач. СПб., «БХВ-Петербург», 2006 г.
13. Рейнгольд Э. Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. М., «Мир», 1980 г.
14. Стивен С. Скиелла, Мигель А. Ревилла Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям. М., «Кудиц-образ», 2005 г.
15. Шень А. Программирование. 2-е изд. Теоремы и задачи. М., МЦНМО, 2004 г.

Сайты

acm.sgu.ru acm.timus.ru acm.uva.es
algolist.manual.ru alglib.manual.ru dl.gsu.unibel.by
www.icl.kazan.ru\turnir neerc.ifmo.ru/school rain.ifmo.ru/~malyshev
olympiads.ru pco.iis.nsk.su/~dyatlov informaics.ru
uoi.kiev.ua byoi.narod.ru contest.ur.ru
sunschool.math.rsu.ru castle.ssu.samara.ru/olymp/ ips.ifmo.ru
borlpasc.narod.ru pascal.hop.ru pms.ru cmc-online.ru
school6.tgl.ru www.petropavl.kz/whouse www.olympiads.ru
www.informatics.ru olympiads.win.tue.nl/ioi www.usaco.org
olympiads.port5.com cpc.baylor.edu neerc.ifmo.ru/online
dl.gsu.unibel.by shade.msu.ru/~mab g6prog.narod.ru

Идеи. Алгоритмы. Решения.

1 S. Из школьного курса физики (Заочный ICL-2008) SPHYSICS.CPP

Граф будем хранить в виде квадратной матрицы смежности, где в каждом элементе <I, J> будем хранить суммарное сопротивление между узлами I и J (если 2 вершины не имеют соединения, то в матрице на соответствующее место ставим 0). При вводе данных будем избавляться от параллельных ребер, вычисляя по соответствующим правилам это суммарное сопротивление. Далее будем многократно проходить по графу, анализируя матрицу смежности или используя операции над матрицами, и искать узлы, которые имеют ровно 2 смежных с ними узла, т.е. кусок с последовательным соединением и заменять эту пару ребер одним (вместо пары ребер <A, B> и <B, C> будет одно ребро <A, C> с суммарным сопротивлением). При этом на отдельных участках могут вновь возникать простейшие параллельные соединения, т.е. параллельные ребра, которые удаляются и суммируются по соответствующим правилам. В конце останется только одно ребро между начальным и конечным узлами с общим сопротивлением.

1 R. Логика решает все (Заочный ICL-2008) RLOGIC.CPP

Сначала нужно определить порядок вычисления операций (например, по заданной схеме легко выписать обратную польскую запись), т.е. при выполнении операции в некоторой вершине в тех вершинах, из которых поступают ее аргументы, должны быть уже вычисленные значения. Для этого через очередь пропускаем все вершины и новый порядок номеров вершин схемы запоминаем в специальном массиве. Дальше перебираем все наборы входных переменных и вычисляем результат в конечной вершине.

1 Q. Любителям детективов (Заочный ICL-2007) QDETECT.CPP

Компьютерная сеть представляет собой дерево, где узлы – это отделы и каждый отдел присоединен к отделу с меньшим номером. Надо найти узел, который является корнем минимального поддерева, куда входят оба подозрительных отдела. Этот узел должен в дереве находиться как можно на более низком уровне, чтобы охватывать как можно меньше лишних отделов. Двигаясь параллельно по двум путям - из двух подозрительных узлов (отделов) вверх по дереву надо достичь общего родительского узла. Движение надо организовать так, что на каждом шаге двигаться по одному пути - там, где номер текущего отдела пока больше номера текущего отдела в другом пути.

1 K. Трудный выбор (Заочный ICL-2006)

Ссылка на литературу по венгерскому методу и двудольным графам – см. по методам оптимизации

2 C. Шарик в лабиринте (Заочный ICL-2006) CBALLF.CPP
Можно организовать перебор вариантов с возвратом, начиная с исходной клетки и до конечной. В каждой клетке двумерного массива будет храниться наименьшее время, которое нужно, чтобы добраться отсюда до конечной клетки. Для ускорения процесса поиска можно заранее просчитать время до конечной клетки из некоторого множества клеток лабиринта, откуда можно попасть на выход напрямую или повернув один раз. Это множество можно легко построить в два этапа. На первом – выбрать точки, откуда напрямую можно попасть в конечную клетку. На втором – выбрать точки, из которых напрямую можно попасть в точки первого типа. Тогда из начальной клетки достаточно будет оптимальным образом добраться только до любой клетки из этого множества.

2 L. Ход конем (Заочный ICL-2008) LKNIGHT.CPP

Задача решается обычным перебором с возвратами всех клеток шахматной доски, достижимых из заданной без повторений. При этом ведется подсчет требуемой суммы. Если она превышает заданное ограничение, то исходная клетка не является решением и продолжается перебор. Если сумма вычислена нормально, то сравниваем с неким максимумом для нахождения самой большой суммы.

2 F. Три клетки (Заочный ICL-2006) FCELL3.CPP

В задаче можно выбрать много кратчайших путей, но во всех случаях наименьшее число клеток на кратчайшем пути равно сумме клеток от самой левой клетки до самой правой и от самой верхней до самой нижней.

2 A. Кубики - не ролики... (Заочный ICL-2008) AMATRIX.CPP

При решении задачи для небольших N параллельно заполняются две матрицы и ведется подсчет двух сумм – для минимальной и максимальной. Заранее упорядочим заданные значения - максимумы по строкам и столбцам. При перестановке строк и столбцов суммы в результате не изменяются, поэтому в примере даны уже упорядоченные вводимые значения. Дальнейшие действия ведутся последовательно от больших заданных значений (максимумов) к меньшим – их надо разместить в матрицах. Возможны два варианта. Очередной максимум встречается в нескольких строках и столбцах. Размещаем это значение в обеих результирующих матрицах по главной диагонали с продолжением по длинной стороне. Если очередной максимум встречается только в строках или столбцах, то , иначе не удастся восстановить матрицу.
Для максимальной матрицы заполнить заданным значением всю часть матрицы, которая еще не заполнена большими числами левее и выше текущей точки.
Для минимальной матрицы все еще незаполненные элементы заполняются нулями.
Для матриц больших размеров придется воспользоваться длинной арифметикой, т.к. суммы не умещаются в переменных стандартных типов. В следующем примере приведены промежуточный и 2 окончательных варианта восстановленной матрицы.

```
 7 7 5 5 5 5 5 3 1 1
 ------------------        MAX                MIN
7 7 . . . . . .    7 7 5 5 5 5 5 3 1 1    7 0 0 0 0 0 0 0 0 0
7 . 7 . . . . . .  7 7 5 5 5 5 5 3 1 1    0 7 0 0 0 0 0 0 0 0
5 . . 5 . . . . .  5 5 5 5 5 5 5 3 1 1    0 0 5 0 0 0 0 0 0 0
5 . . . 5 . . . .  5 5 5 5 5 5 5 3 1 1    0 0 0 5 0 0 0 0 0 0
5 . . . . 5 5 5 . . .  5 5 5 5 5 5 5 3 1 1    0 0 0 0 5 5 5 0 0 0
3 . . . . . . . 3 1 1  3 3 3 3 3 3 3 3 1 1    0 0 0 0 0 0 0 3 1 1
```

2 B. Охотники за привидениями (Заочный ICL-2007)

ссылка на книгу [Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К., стр. 1081]

2 L. Лабиринты и монстры. (Заочный ICL-2007) LABIRINT.CPP

Будем записывать в каждую клетку такое целое число, за какое количество ходов до нее может добраться один монстр. С левого верхнего угла и до побочной диагонали включительно будем записывать ходы хорошего монстра, а с правого нижнего навстречу ему ходы плохого монстра. Те клетки на побочной диагонали, в которые можно попасть обоим монстрам, являются возможными местами их встречи.

2 D. Еще одна игра (Заочный ICL-2008) DGAME.CPP

Для каждой строки и каждого столбца нужно посчитать число фишек на этой линии. Организуем двойной цикл по всем клеткам - если есть фишка, смотрим 4 варианта хода - вверх, вниз, влево, вправо и проверяем возможность хода.

2 J. Развертка куба (Заочный ICL-2006) JJCUBE.CPP

Надо аккуратно перебрать все варианты развертки куба, которые могут разместиться на клетчатом поле. Особое внимание уделить размещению развертки на краях доски.

3 R. Игра (Заочный ICL-2007) RGAME.CPP

Задача решается методом оптимальных хвостов – выбором на каждом шаге движения оптимального из двух возможных вариантов. В каждой точке запоминаем цену оптимального пути, по которому дошли до этой точки.

4 C. Новый лицей (Заочный ICL-2008) CLICEUM.CPP

Заметим, что обе координаты не зависят друг от друга, поэтому лучшее место для лицея будем искать по каждой координате одинаково, но отдельно. Если координату X всех точек (местожительства детей) упорядочить по возрастанию (неубыванию), то от той точки, которая окажется в середине упорядоченного массива, до всех остальных точек суммарное расстояние будет самым минимальным. Если брать точки левее или правее, то легко проверить, что суммарное расстояние может только возрасти или остаться таким же, т.к. с одной стороны от центра число точек обязательно будет больше и их суммарное расстояние будет расти быстрее, чем убывать суммарное расстояние с другой стороны, где число точек меньше.

4 D. Слишком вложенные скобки (Заочный ICL-2006) DPARENT.CPP

За один проход по последовательности ведем подсчет глубины вложения очередных скобок. При встреч открывающей скобки счетчик увеличиваем на 1, а при закрывающей – уменьшаем. Если скобки не достаточно глубоко вложены (счетчик не превышает заданного значений), то выводим их в результат.

4 H. Последовательности из 0 и 1. (Заочный ICL-2007) H101.CPP

Простая проверка для небольших значений длины последовательности приводит к формуле для чисел Фибоначчи. Надо только учесть, что последние числа большие и их лучше вычислить отдельно (заранее) с помощью калькулятора.

4 K. Легонькая задачка (Заочный ICL-2007) KLIGHT.CPP

Предварительно подготовим массив простых чисел, например, с помощью решета Эратосфена. Дальше организуем перебор прогрессий, начиная с разных первых элементов и задавая различные значения разности прогрессии.

4 J. Лесенки (Заочный ICL-2007) JTRAP.CPP

Искомое количество легко подсчитывается по рекуррентной формуле, которая выводится для малых значений общего числа кубиков.

4 G. Марсианин Василий (Заочный ICL-2008) GMARS.CPP
4 G. Связка брусков (Заочный ICL-2006) G2BARS.CPP
4 N. Тени исчезают в полдень (Заочный ICL-2008) NSHADOW.CPP

Три задачи на аккуратное вычисление суммы вещественных чисел по простым формулам.

4 P. Шифровка. (Заочный ICL-2007) PSHIFR.CPP
Задача на использование перебора с возвратами. На каждом этапе выбираем букву с наименьшим возможным кодом и идем дальше. Несколько вариантов может быть для цифр 1 и 2, потому, что это могут быть одиночные самостоятельные коды, а могут быть первые цифры двузначных кодов 11, 12 и т.д. Если дошли до конца, то получили еще один вариант расшифровки и возврат на шаг назад. Возврат выполняется также в случаях, когда очередная цифра больше 2-х и она уже была рассмотрена.

4 B. Банкноты (Пробный ICL-2006) BANKNOTE.CPP

Заводим массив из нулей, где каждый элемент соответствует по номеру одной банкноте. Каждую банкноту отмечаем в массиве единичкой. В конце ищем элемент равный 0 – он соответствует отсутствующей банкноте.

5 D. Числа в вершинах (Заочный ICL-2007) DGRAPH.CPP

Можно каждой вершине графа приписать число 1, а потом каждую пару вершин, имеющих общее ребро, умножить на один и тоже простой множитель, но для разных ребер использовать разные числа. При таком подходе в вершинах графа с большим числом ребер окажутся огромные числа. Можно оптимизировать этот процесс за счет предварительного перебора всех троек вершин, где каждая пара связана ребром. Для этого потребуется обычный тройной цикл. Тогда на всю тройку этих вершин будет использоваться одно и тоже простое число. Дальнейшую оптимизацию (перебор четверок, пятерок и т.д.) можно не выполнять – это уже лишнее.

5 H. Последние цифры (Заочный ICL-2006) H2DIGITS.CPP

Каждое число можно разложить на простые множители единственным образом, т.е. простой делитель P_1 входит в это разложение с кратностью K_1 и т.д. Тогда общее число делителей исходного числа равно произведению (K_1+1) $(K_2+1)\ldots$ (K_n+1), где n – количество разных простых делителей. При вычислении этого числа надо иметь в виде, что оно может получиться большим, и все его цифры нам не нужны. Поэтому при вычислениях всегда достаточно хранить только последние две цифры этого произведения.

5 E. Простые факториалы (Заочный ICL-2007) ESIMPFAC.CPP

Сначала выделим простые или невозможные случаи. Делим исходное число на первые простые числа до тех пор, пока оно делится, затем к частному прибавляем 1 и опять продолжаем делить на следующие простые числа. Если в конце будет получена единица, то мы имеем два числа в ответе, иначе – исходные данные неправильные.

5 L. Странная игра (Заочный ICL-2007) LGAME.CPP

Используем массив для хранения промежуточных результатов – в нем строим по порядку все числа, получающиеся из исходного числа A применением 3-х правил. Для каждого числа, получающегося по одной из трех заданных формул, отмечаем в очередном по номеру элементе массива из какого числа оно получилось и за сколько шагов. Далее переходим к следующему элементу строящейся цепочки и к нему применяем эти 3 правила. Когда в конце будет получено заключительное число B, в обратном порядке получаем всю цепочку преобразований.

6 A. Сокращение одночленов (Заочный ICL-2006) ACASTING.CPP

Аккуратно сосчитать количество вхождений (степень) каждой переменной в числителе и знаменателе и потом сформировать результат

6 E. Школьный субботник (Заочный ICL-2006) ESABBATH.CPP

Аккуратно выполнять все действия, закодированные во входном файле.

6 M. Перенаправление номеров (Заочный ICL-2007) MNUMBER.CPP

Заводим массив для всех номеров абонентов, в котором будут записаны номера перенаправлений каждого абонента или -1, если какой-то номер не перенаправляется на другой. Далее для каждого номера идем по цепочке всех перенаправлений, пока не дойдем до -1 – это и есть конечный абонент.

7 B. Нечетный N-угольник (Заочный ICL-2006) BPOLYGON.CPP

Координаты $<A_i, B_i>$ точки на середине отрезка между точками с номерами i и i+1 вычисляются как средние арифметические координат концов отрезков, т.е. $A_i=(X_i+X_{i+1})/2$, $B_i=(Y_i+Y_{i+1})/2$. Суммируя все эти равенства находим, что сумма всех A_i равна сумме всех X_i. Отсюда, группируя слагаемые парами, находим формулу для вычисления $X_1=2(A_1+A_2+\ldots+A_n)-A_2-A_4-\ldots-A_{n-1}$. Потом последовательно находим координаты остальных точек по другой простой формуле $X_i=2A_{i-1}-X_{i-1}$, для i=2,…n. Вторую координату каждой вершины вычисляем по аналогичным формулам.

7 G. Mission Impossible (Заочный ICL-2007) GMISSION.CPP

Если есть такая точка, то где она может находиться? Полный перебор и проверка всех точек полигона требует большого количества времени (порядка N^2Z проверок квадратного неравенства), хотя это не сложно организовать. Если есть точка внутри полигона, которая находится «достаточно далеко» от всех радаров, то

должна быть точка, которая находится очень близко к зоне покрытия какого-то радара, но не покрываемая им. Поэтому, можно организовать для каждого круга вокруг радаров проверку точек, находящихся рядом с окружностью с внешней стороны и которые находятся внутри полигона. Таких точек будет примерно $2\pi RZ$.

8 E. По порядку становись (Заочный ICL-2008) EORDER.CPP

Задачу можно решать просто методом перебора с возвратом последовательно строя одну цепочку букв за другой в требуемом порядке. Исходную строку букв упорядочить и начать с пустой строки
// цикл по шагам до нужного номера
// если идем вперед, то
// если еще есть, то добавляем очередную букву из исходных данных (++) и идем вперед
// иначе - убираем последнюю букву и переходим к предпоследней - не идем вперед
// иначе (если не идем вперед)
// если можно, последнюю букву заменяем на большую по алфавиту (++) и идем вперед
// иначе - убираем последнюю букву и переходим к предпоследней - не идем вперед

8 A. Максимальная тройка (Пробный ICL-2006) ATRIPLE.CPP

Рассматриваем все варианты расположения троек – в линию и углом.