Содержание

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

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

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

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

Must have

Задача 8A. Выбор вершин взвешенного дерева [0.1 sec, 256 mb]

Дан граф, являющийся деревом. В вершинах графа написаны целые числа. Множество вершин графа называется *допустимым*, если никакие две вершины этого множества не соединены ребром.

Рассмотрим все допустимые множества вершин графа. Для каждого такого множества вычислим сумму чисел, написанных в его вершинах. Какова максимальная из этих сумм?

Формат входных данных

Граф в этой задаче задан в виде *корневого дерева*. В графе выделена вершина — *корень дерева*. Для каждой вершины i, не являющейся корнем, задан номер вершины-предка p_i в корневом дереве. Дерево, заданное таким образом, состоит из рёбер $i - p_i$ для всех вершин i, кроме корня.

В первой строке входного файла записано целое число n — количество вершин в графе ($1 \leqslant n \leqslant 100$). В следующих n строках задан граф. В i-й из этих строк записаны через пробел два целых числа p_i и q_i; здесь p_i — номер вершины-предка i-ой вершины, а q_i — число, записанное в этой вершине. Для корня дерева $p_i = 0$; для всех остальных вершин $1 \leqslant p_i \leqslant n$. Числа q_i не превосходят по модулю 10 000.

Гарантируется, что заданный во входном файле граф является деревом.

Формат выходных данных

В первой строке выходного файла выведите одно число — максимальную сумму чисел в допустимом множестве.

Примеры

selectw.in	selectw.out
5 0 1 1 2 1 3 2 4 3 5	10
6 5 8 6 0 5 -1 1 1 0 3 1 2	8

Замечание

Чтобы хранить дерево, используйте `vector<int> c[n];`

Задача 8B. Строка Фибоначчи [0.1 sec, 256 mb]

Строка Фибоначчи — это строка из нулей и единиц, в которой не встречается двух идущих подряд единиц.

Даны числа n и k. Нужно вывести строку Фибоначчи, которая состоит из n цифр и является k-ой в лексикографическом порядке из таких строк.

Формат входных данных

В первой строке входного файла записаны целые числа n и k через пробел ($0 \leqslant n \leqslant 44$, $0 \leqslant k \leqslant 2 \cdot 10^9$). Гарантируется, что k-ая строка из n символов существует. Строки нумеруются с нуля.

Формат выходных данных

Выведите лексикографически k-ую строку Фибоначчи длины n.

Вывод *обязательно* завершать переводом строки.

Примеры

fibstr.in	fibstr.out
3 0	000
3 1	001
3 2	010
3 3	100
3 4	101

Замечание

В тему лекции.

Задача 8C. Отметки на подмножествах 2 [0.5 sec, 256 mb]

Рассмотрим множество $\mathcal{S}$, состоящее из n элементов — натуральных чисел 1, 2, ..., n.

Сперва отметим несколько подмножеств $\mathcal{S}$, а также все подмножества этих подмножеств.

Затем снимем все отметки, если они есть, с нескольких подмножеств $\mathcal{S}$, а также со всех их подмножеств.

Найдите количество отмеченных подмножеств после всех этих операций.

Формат входных данных

В первой строке входного файла заданы через пробел три целых числа n, x и y. Следующие x строк содержат описания подмножеств, отмеченных на первом шаге, по одному на строке; также были отмечены все подмножества этих подмножеств. Наконец, последние y строк входного файла содержат описания подмножеств, с которых сняли отметки на втором шаге, по одному на строке; также были сняты отметки со всех их подмножеств. Описание каждого подмножества имеет вид $k\ a_1\ a_2\ \ldots\ a_k$, где k — количество элементов данного подмножества ($0 \leqslant k \leqslant n$), а a_i — сами элементы (a_i попарно различны, $1 \leqslant a_i \leqslant n$). Элементы могут быть перечислены в любом порядке.

Формат выходных данных

В первой строке выходного файла выведите одно число — количество отмеченных подмножеств после всех описанных операций.

Ограничения

- $1 \leqslant n \leqslant 20$
- $0 \leqslant x, y \leqslant 1000$

Примеры

marked2.in	marked2.out
1 1 1 1 1 0	1
2 0 1 2 2 1	0
3 2 1 2 1 2 2 2 3 2 1 3	3

Пояснения к примерам

В первом примере на первом шаге ставится отметка на подмножество {1} и на пустое подмножество, на втором шаге с пустого подмножества снимается отметка.

Во втором примере отметок нет.

В третьем примере на первом шаге отмеченными оказываются следующие шесть подмножеств: {}, {1}, {2}, {3}, {1, 2} и {2, 3}. На втором шаге снимаются отметки с трёх подмножеств {}, {1} и {3}.

Замечание

Предполагается решение за $\mathcal{O}(2^n n)$.

Если вы каждое из $x + y$ множеств обрабатываете за $\mathcal{O}(2^n)$, это слишком долго.

Представьте, что вы ставите пометку не во всех подмножествах множества a, а только в самом a. И так для каждого a. Это база динамики. Теперь нужно сделать переход, пометить ещё какие-то множества...

Задача 8D. Коммивояжёр возвращается! [2.5 sec, 256 mb]

Коммивояжёр возвращается в систему Альфы Центавра! Население системы с нетерпением ждёт его прибытия — каждый хочет приобрести что-нибудь с далёких планет!

Как обычно, коммивояжёр хочет минимизировать транспортные расходы. Он выбирает начальную планету, прилетает туда на межгалактическом корабле, после чего посещает все остальные планеты системы в порядке, минимизирующем суммарную стоимость посещения, и на другом межгалактическом корабле улетает обратно. Естественно, коммивояжёр не хочет летать ни на какую планету дважды.

Найдите оптимальный маршрут для коммивояжёра. Массы больше не могут ждать!

Формат входных данных

В системе Альфы Центавра n планет. Это число записано в первой строке входного файла ($1 \leqslant n \leqslant 19$). Следующие n строк содержат по n чисел каждая: j-ое число на i-ой из этих строк — стоимость перемещения a_{ij} от i-ой планеты до j-ой. Числа в каждой строке разделены пробелами. Числа a_{ii} не несут полезной информации. Все числа во входном файле положительны и не превосходят 10^8.

Формат выходных данных

В первой строке выходного файла выведите минимальную суммарную стоимость посещения всех планет. Во второй строке выведите n чисел через пробел — номера планет системы в порядке их посещения. Если оптимальных маршрутов несколько, можно вывести любой из них.

Пример

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

Замечание

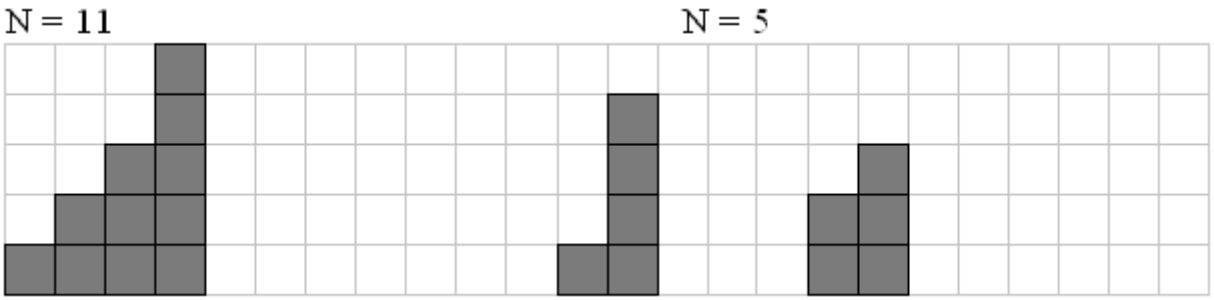
Решение за $\mathcal{O}(2^n n^2)$ без проблем должно получать OK.

Задача коммивояжера разобрана на лекции.

Обязательные задачи

Задача 8E. Лестницы [0.1 sec, 4 mb]

У маленького мальчика есть набор из n кубиков ($5 \leqslant n \leqslant 500$). Из этих кубиков можно сложить различные лестницы. Лестницы имеют ступени различного размера, следующие в порядке возрастания этого размера (обратите особое внимание на то, что лестница не может иметь две одинаковые ступени). Каждая лестница должна иметь минимум две ступени, и каждая ступень должна состоять минимум из одного кубика. На рисунке приведены примеры лестниц для $n = 11$ и $n = 5$:



Найдите число q различных лестниц, которые маленький мальчик может построить ровно из n кубиков.

Формат входных данных

Число n.

Формат выходных данных

Число q.

Примеры

stairs.in	stairs.out
212	995645335

Задача 8F. Сумма «случайных» чисел 2 [0.8 sec, 256 mb]

На столе лежат n шариков, на каждом шарике написано натуральное число. Валя и Витя используют эти шарики, чтобы получать «случайные» числа. Процедура получения «случайного» числа следующая. Сначала Валя берёт со стола некоторое непустое подмножество шариков; при этом необходимо, чтобы на столе остался хотя бы один шарик. Затем Витя также берёт какое-то непустое подмножество шариков, оставшихся на столе; после этого шариков на столе может не остаться. Наконец, Валя и Витя вычисляют «случайное» число $r = a \bmod b$, где a — это сумма чисел на шариках Вали, а b — сумма чисел на шариках Вити; $a \bmod b$ — это остаток от деления a на b. После этого все шарики возвращаются на стол.

Предположим, что Валя выбрала каждое допустимое подмножество шариков, и для каждого из них Витя выбрал по одному разу все допустимые подмножества оставшихся шариков. Найдите сумму всех «случайных» чисел, которые получились при этом.

Формат входных данных

В первой строке входного файла задано целое число n. Вторая строка содержит n целых чисел s_1, s_2, …, s_n через пробел; s_i — это число, написанное на i-м шарике.

Формат выходных данных

В первой строке выходного файла выведите одно число — сумму всех получившихся у Вали и Вити «случайных» чисел.

Ограничения

- $2 \leqslant n \leqslant 16$
- $1 \leqslant s_i \leqslant 1\,000\,000$

Примеры

modsum2.in	modsum2.out
2 1 1	0
3 1 1 1	3
3 3 1 2	8

Пояснения к примерам

В первом примере у Вали и Вити получаются числа $a = b = 1$, а $1 \bmod 1 = 0$.

Во втором примере числа, отличные от нуля, получаются, только когда Валя берёт один любой шарик, а Витя — оба оставшихся.

В третьем примере суммируются следующие числа:

$$\begin{array}{lll} 0 = 3 \bmod 1 & 1 = 3 \bmod 2 & 0 = 3 \bmod (1+2) \\ 1 = 1 \bmod 3 & 1 = 1 \bmod 2 & 1 = 1 \bmod (3+2) \\ 2 = 2 \bmod 3 & 0 = 2 \bmod 1 & 2 = 2 \bmod (3+1) \\ 0 = (3+1) \bmod 2 & 0 = (3+2) \bmod 1 & 0 = (1+2) \bmod 3 \end{array}$$

Замечание

Предполагается решение за $\mathcal{O}(3^n)$.

Задача 8G. Гиперкуб [0.1 sec, 256 mb]

Гиперкуб — это обобщение понятия трёхмерного куба на N измерений. Нуль-мерным гиперкубом является точка, одномерным — отрезок, двумерным — квадрат. В общем же случае N-мерный гиперкуб — это правильный N-мерный многогранник, каждая из $2 \cdot N$ граней которого является $(N-1)$-мерным гиперкубом. Например, для $N = 2$ квадрат — это правильный многоугольник, каждая из $2 \cdot 2 = 4$ сторон которого — отрезок, то есть одномерный гиперкуб. Отметим, что N-мерный гиперкуб имеет 2^N вершин.

Старшеклассник Петя долго разбирался, что же такое гиперкуб, но наконец понял, как этот объект устроен, и ему настолько понравилось, что он даже придумал свою собственную игру на гиперкубе. Игра заключается в следующем.

Рассмотрим N-мерный единичный гиперкуб. Расположим его таким образом, чтобы одна из вершин находилась в начале координат — точке $(0, 0, \ldots, 0)$ в N-мерном пространстве, а для любой из остальных вершин каждая координата равнялась бы нулю или единице. В каждой из 2^N вершин запишем по целому неотрицательному числу. Игрок начинает свой путь в начале координат. За один ход можно переместиться из текущей вершины по любому ребру при условии, что сумма координат новой вершины строго больше суммы координат старой. Игра заканчивается, когда игрок попадает в вершину $(1, 1, \ldots, 1)$, имеющую максимальную сумму координат — N. Результат игры — сумма чисел во всех посещённых игроком вершинах. Цель игры — пройти по гиперкубу таким образом, чтобы эта сумма (количество очков за игру) оказалась как можно больше.

Петя довольно быстро понял, что между двумя вершинами гиперкуба A и B ребро есть тогда и только тогда, когда все координаты этих вершин $(A_1, A_2, \ldots, A_N)$ и $(B_1, B_2, \ldots, B_N)$ совпадают, кроме одной, которая равна нулю у одной из вершин (скажем, A) и единице у другой (B). Поскольку при этом $A_1 + A_2 + \ldots + A_N + 1 = B_1 + B_2 + \ldots + B_N$, то по такому ребру можно перемещаться из A в B, но не наоборот. Однако, сыграв в свою игру, Петя не может с уверенностью сказать, является ли полученная им сумма максимальной или можно на данном гиперкубе сыграть по-другому и набрать больше очков.

Напишите программу, которая по данному гиперкубу находит максимальную сумму, которую можно получить, сыграв в эту игру.

Формат входных данных

В первой строке входного файла записано число N ($1 \leqslant N \leqslant 10$) — размерность гиперкуба. В следующих 2^N строках содержится по одному числу в каждой; в $(k+2)$-ой строке записано C_k ($0 \leqslant C_k \leqslant 1000$) — число в вершине с номером k.

Номер вершины вычисляется так: вершина A с координатами $(A_1, A_2, \ldots, A_N)$ имеет номер, равный $A_1 \cdot 2^{N-1} + A_2 \cdot 2^{N-2} + \ldots + A_{N-1} \cdot 2 + A_N$, то есть координаты просто интерпретируются как двоичная запись номера вершины. В этой нумерации начальная вершина имеет номер 0, а конечная — номер $2^N - 1$.

Формат выходных данных

В выходной файл выведите одно число — максимальную сумму, которую можно получить при игре на данном гиперкубе.

Пример

cube.in	cube.out
3 1 2 3 4 5 6 7 8	21

Пояснение к примеру

Наш маршрут таков:

- вершина 0 (число 1, координаты (0, 0, 0)) — начальная
- вершина 4 (число 5, координаты (1, 0, 0))
- вершина 6 (число 7, координаты (1, 1, 0))
- вершина 7 (число 8, координаты (1, 1, 1)) — конечная

Наше количество очков: $1 + 5 + 7 + 8 = 21$.
Любой другой маршрут с соблюдением правил игры даёт меньшее количество очков.

Задача 8H. Самый длинный путь 22 [1 sec, 256 mb]

В данном ориентированном графе найдите самый длинный путь такой, что каждая вершина графа встречается в нём не более одного раза.

Формат входных данных

В первой строке входного файла заданы через пробел два целых числа n и m ($1 \leqslant n \leqslant 22$, $0 \leqslant m \leqslant 1000$). В следующих m строках заданы рёбра графа в формате u_i v_i — номера начальной и конечной вершин ребра i, соответственно. Граф может содержать петли и кратные рёбра.

Формат выходных данных

В первой строке выходного файла выведите длину искомого пути l. Во второй строке выведите $l + 1$ число через пробел — вершины пути в порядке обхода. Если оптимальных ответов несколько, можно вывести любой из них.

Примеры

path.in	path.out
3 3 1 2 2 3 3 1	2 1 2 3
4 6 1 2 2 1 2 3 2 4 3 2 4 2	2 1 2 4
5 3 3 2 2 2 1 5	1 3 2

Замечание

Предполагается решение за $\mathcal{O}(2^n n)$. Разобрана на лекции.

Любители перебора с отсечениями могут потренироваться и получить OK ;-)

Дополнительные задачи

Задача 8I. Три типа скобок [0.1 sec, 256 mb]

Определим по индукции множество $\mathcal{T}$ *правильных скобочных последовательностей из трёх типов скобок*:

- $\varepsilon \in \mathcal{T}$ (пустая строка)
- $A \in \mathcal{T} \Rightarrow (A) \in \mathcal{T}$
- $A \in \mathcal{T} \Rightarrow [A] \in \mathcal{T}$
- $A \in \mathcal{T} \Rightarrow \{A\} \in \mathcal{T}$
- $A \in \mathcal{T}$, $B \in \mathcal{T} \Rightarrow AB \in \mathcal{T}$

Пусть теперь $\mathcal{T}_n$ — это множество правильных скобочных последовательностей из $2n$ символов — n открывающих и n закрывающих скобок.

Упорядочим элементы множества $\mathcal{T}_n$ лексикографически с некоторым порядком символов '`(`', '`)`', '`[`', '`]`', '`{`' и '`}`'.

По данным числам n и p, а также порядку, заданному на скобках, найдите p-ый в этом порядке элемент множества $\mathcal{T}_n$.

Формат входных данных

В первой строке входного файла заданы через пробел два целых числа n и p ($0 \leqslant n \leqslant 20$, $0 \leqslant p \leqslant 9 \cdot 10^{18}$). Скобочные последовательности нумеруются с нуля.

Во второй строке записаны шесть символов — '`(`', '`)`', '`[`', '`]`', '`{`' и '`}`' — в некотором порядке. Их порядок задаёт лексикографический порядок на множестве $\mathcal{T}_n$.

Формат выходных данных

В первой строке выходного файла выведите $2n$ символов без пробелов — p-ю правильную скобочную последовательность длины $2n$ из трёх типов скобок.

Если для данного n не существует p-я правильная скобочная последовательность, выведите в первой строке "`N/A`".

Примеры

parens3.in	parens3.out
1 0 ()}[{]	()
1 1 ()}[{]	[]
1 2 ()}[{]	{}
1 3 ()}[{]	N/A

Задача 8J. Справедливый дележ [0.5 sec, 256 mb]

— Я хотел честно, — сказал Балаганов, собирая деньги с кровати, — по справедливости.

В коробке от сигарет «Кавказ», отнятой у Корейко, было всего $1 \le N \le 15$ купюр, каждая номиналом $1 \leqslant a_i \leqslant 10^3$. Разделить надо было на $1 \le K \le 100$ частей, причём известно, что общая сумма делится на K. Метод деления «по справедливости» следующий: если разделить поровну не получается, то делить следует так, чтобы среднеквадратичное отклонение было минимальным.

Среднеквадратичное отклонение для данного способа дележа определяется следующим образом. Пусть i-й человек получил сумму денег, равную S_i. Обозначим за M среднее арифметическое сумм денег, полученных каждым: $M = (\sum_{i=1}^{K} S_i)/K$. Тогда среднеквадратичное отклонение можно вычислить так: $\sigma = \sqrt{(\sum_{i=1}^{K}(S_i - M)^2)/K}$.

— И как? — поинтересовался Остап.

— Сложно... — вздохнул Шура.

Попробуйте и вы разделить деньги «по справедливости».

Формат входных данных

В первой строке входного файла задано N — количество купюр и K — количество участников дележа. Далее, в следующей строке, заданы N целых чисел a_i — номиналы купюр.

Формат выходных данных

В первой строке выведите искомое среднеквадратичное отклонение с точностью до шести знаков после десятичной точки, в следующей строке выведите N чисел от 1 до K — номер участника дележа, которому досталась i-я купюра. Если оптимальных решений несколько, разрешается выводить любое.

Пример

fair.in	fair.out
4 3 1 2 3 6	1.414214 2 2 1 3

Замечание

Существует решение за $\mathcal{O}(3^n)$, но за $\mathcal{O}(3^n n)$ тоже можно запихать.

Задача 8K. Covering Points [1 sec, 256 mb]

Сегодня, Вася отправляется в научно-исследовательский институт сфер и кругов (RISC). Этот институт изучает все задачи, связанные со сферами и кругами, и теперь Васю просят что-то сделать с кругами.

Даны N точек на плоскости, необходимо покрыть их все K кругами минимально возможного радиуса. Кругам разрешено касаться и пересекаться друг с другом. Не могли бы вы помочь Васе справиться с заданием?

Формат входных данных

Входной файл состоит из одного или нескольких тестов. Каждый тест начинается со строки, содержащей два целых числа N и K ($1 \leqslant N \leqslant 15$, $1 \leqslant K \leqslant N$), а затем N строк, каждая из которых описывает одну точку её целыми координатами x_i и y_i. Точки могут совпадать. Все координаты не превышают 1000 по абсолютной величине. Ввод завершается тестом $N = K = 0$, который учитывать не нужно. Общая сумма чисел N по всем тестам в одном входном файле не превышает 150.

Формат выходных данных

Для каждого теста сначала выведете минимальный радиус, затем приведите пример покрытия. Если есть несколько решений — выводите любое. Радиусы нужно вывести с точностью не менее 6 знаков после запятой. Чекер будет осуществлять проверку "попала ли точка в круг" с точностью 10^{-6}. Мы рекомендуем выводить числа с максимальной точностью.

Заметьте, что круг с нулевым радиусом — точка.

Примеры

cover.in

```
3 2
0 0
0 1
0 2
3 1
0 0
0 1
1 0
0 0
```

cover.out

```
Case 1: The minimal possible radius is 0.5
  circle 1 at (0.0, 0.5)
  circle 2 at (0.0, 2.0)

Case 2: The minimal possible radius is 0.7071067811865476
  circle 1 at (0.5, 0.5)
```