Задача A. Гиперкуб

Имя входного файла:	`cube.in`
Имя выходного файла:	`cube.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Гиперкуб — это обобщение понятия трёхмерного куба на N измерений. Нульмерным 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 входного файла

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 выходного файла

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 B. Сумма «случайных» чисел

Имя входного файла:	`modsum.in`
Имя выходного файла:	`modsum.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

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 входного файла

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 выходного файла

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

Ограничения

- $2 \leqslant n \leqslant 10$
- $1 \leqslant s_i \leqslant 1000$

Примеры

modsum.in	modsum.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:

$0 = 3 \bmod 1$	$1 = 3 \bmod 2$	$0 = 3 \bmod (1+2)$
$1 = 1 \bmod 3$	$1 = 1 \bmod 2$	$1 = 1 \bmod (3+2)$
$2 = 2 \bmod 3$	$0 = 2 \bmod 1$	$2 = 2 \bmod (3+1)$
$0 = (3+1) \bmod 2$	$0 = (3+2) \bmod 1$	$0 = (1+2) \bmod 3$

Задача C. Сумма «случайных» чисел 2

Имя входного файла:	modsum2.in
Имя выходного файла:	modsum2.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Условие такое же, как у задачи modsum.

Ограничения

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

Задача D. Сумма «случайных» чисел 3

Имя входного файла:	modsum3.in
Имя выходного файла:	modsum3.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Условие такое же, как у задачи modsum.

Ограничения

- $2 \leqslant n \leqslant 18$
- $1 \leqslant s_i \leqslant 10$
- Сумма всех s_i не превосходит 50.

Задача E. Отметки на подмножествах

Имя входного файла:	marked.in
Имя выходного файла:	marked.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

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 входного файла

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 ... 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 выходного файла

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

Ограничения

- $1 \leqslant n \leqslant 10$
- $0 \leqslant x, y \leqslant 1000$

Примеры

marked.in	marked.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}.

Задача F. Отметки на подмножествах 2

Имя входного файла:	`marked2.in`
Имя выходного файла:	`marked2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Условие такое же, как у задачи marked.

Ограничения

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

Задача G. Отметки на подмножествах 3

Имя входного файла:	`marked3.in`
Имя выходного файла:	`marked3.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Условие такое же, как у задачи marked.

Ограничения

- $1 \leqslant n \leqslant 26$
- $0 \leqslant x, y \leqslant 1000$

Задача H. Коммивояжёр возвращается!

Имя входного файла:	`salesman.in`
Имя выходного файла:	`salesman.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	64 мегабайта

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 входного файла

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 выходного файла

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

Задача I. Самый длинный путь

Имя входного файла:	`path.in`
Имя выходного файла:	`path.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

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

Формат входного файла

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 выходного файла

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