Задача A. ЗОРП 1

Имя входного файла:	`knapsack-1.in`
Имя выходного файла:	`knapsack-1.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Перед вами лежат n котиков. Каждый котик характеризуется своим весом w_i и своей мимимишностью c_i. Вы хотите выбрать некоторое число котиков суммарным весом не более чем S так, чтобы их суммарная мимимишность была максимально возможной.

Формат входных данных

В первой строке содержатся два целых числа n и S — число котиков и максимальный допустимый суммарный вес ($1 \leqslant n \leqslant 100$, $1 \leqslant S \leqslant 10^4$). Следующие n строк содержат по два целых числа w_i и c_i — вес и мимишность i-го котика ($1 \leqslant w_i \leqslant 10^4$, $0 \leqslant c_i \leqslant 10^7$).

Формат выходных данных

В первой строке выведите суммарную мимишность выбранных котиков. Во вторую строку выведите целое число k — количество выбранных котиков. В третьей строке выведите k чисел — номера выбранных котиков. Если оптимальных ответов несколько, то разрешается вывести любой из них.

Примеры

knapsack-1.in	knapsack-1.out
3 10 1 2 4 3 8 9	11 2 3 1

Задача B. ЗОРП 2

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

Перед вами лежат n котиков. Каждый котик характеризуется своим весом w_i и своей мимимишностью c_i. Вы хотите выбрать некоторое число котиков суммарным весом не более чем S так, чтобы их суммарная мимимишность была максимально возможной.

Формат входных данных

В первой строке содержатся два целых числа n и S — число котиков и максимальный допустимый суммарный вес ($1 \leqslant n \leqslant 100$, $1 \leqslant S \leqslant 10^9$). Следующие n строк содержат по два целых числа w_i и c_i — вес и мимишность i-го котика ($1 \leqslant w_i \leqslant 10^7$, $0 \leqslant c_i \leqslant 10^4$). Гарантируется, что сумма всех c_i не превосходит 10^4.

Формат выходных данных

В первой строке выведите суммарную мимишность выбранных котиков. Во вторую строку выведите целое число k — количество выбранных котиков. В третьей строке выведите k чисел — номера выбранных котиков. Если оптимальных ответов несколько, то разрешается вывести любой из них.

Примеры

knapsack-2.in	knapsack-2.out
3 10 1 2 4 3 8 9	11 2 3 1

Задача C. Программирование вслепую

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вам нужно решить *задачу о сумме подмножеств*, вариант задачи о рюкзаке. У вас есть n предметов, имеющих массу w_i грамм, и рюкзак вместимостью c грамм. Нужной найти подмножество предметов максимальной суммарной массы, не превосходящей c.

Слишком просто? Но это нужно делать вслепую!

В начале вам даются только два числа n и c — количество предметов и вместимость рюкзака соответственно. Тестирующей системе известны положительные веса w_i и две битовые строки A и B, каждая содержит n бит. Каждая строка представляет из себя возможное решение задачи: если i-й бит равен 1, тогда i-й предмет участвует в решении. A хранит предыдущий выбранный кандидат на решение, B — новый предложенный кандидат на решение, полученный инвертированием случайного бита в A. Вы можете одобрить или отклонить это предложение. Ваша задача — остановиться, когда B будет соответствовать оптимальному решению.

Изначально A инициализировано каким-то значением, которое вам не известно. На каждом шаге A копируется в B, и затем бит на случайной позиции в B инвертируется. Вам даётся масса, соответствующая B, равная $\sum_{i=1}^{n} w_i \cdot B(i)$, где $B(i)$ равен i-му биту в B.

В ответ, программа может выполнять одну из следующих трех действий:

- `stop` — финальное действие, после его выполнения ваша программа должна завершиться;
- `accept` — скопировать значение B в A;
- `decline` — проигнорировать B.

После каждого не финального действия начинается новый шаг. Вы можете выполнить не более 1000 действий.

Протокол взаимодействия

При запуске программы на вход подаются три целых числа n, c и масса, соответствующая B ($1 \leqslant n \leqslant 20, 0 \leqslant c \leqslant 10^9$).

Ваша программа должна вывести выполняемое действие в стандартный поток вывода и после ответа проверяющей системы считать массу, соответствующую B, и так далее.

Программа должна завершиться сразу после выполнения действия `stop`. После вывода каждой команды необходима выводить символ перевода строки и сбрасывать поток стандартного вывода.

Гарантируется, что $1 \leqslant w_i \leqslant 10^8$.

Примеры

Выполненная операция	Ответ тестирующей системы	Значение A	Значение B
Запуск	`2 5 3`	`00`	`01`
`decline`	`2`	`00`	`10`
`accept`	`5`	`10`	`11`
`stop`		`11`	

Веса предметов в примере — 2 и 3.

Задача D. Сокровища

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

Дочь короля Флатландии собирается выйти за прекрасного принца. Принц хочет подарить принцессе сокровища, но он не уверен какие именно бриллианты из своей коллекции выбрать.

В коллекции принца n бриллиантов, каждый характеризуется весом w_i и стоимостью v_i. Принц хочет подарить наиболее дорогие бриллианты, однако король умен и не примет бриллиантов суммарного веса больше R. С другой стороны, принц будет считать себя жадным всю оставшуюся жизнь, если подарит бриллиантов суммарным весом меньше L.

Помогите принцу выбрать набор бриллиантов наибольшей суммарной стоимости, чтобы суммарный вес был в отрезке $[L, R]$.

Формат входных данных

Первая строка содержит число n ($1 \leqslant n \leqslant 32$), L и R ($0 \leqslant L \leqslant R \leqslant 10^{18}$). Следующие n строк описывают бриллианты и содержит по два числа — вес и стоимость соответствующего бриллианта ($1 \leqslant w_i, v_i \leqslant 10^{15}$).

Формат выходных данных

Первая строка вывода должна содержать k — количество бриллиантов, которые нужно подарить принцессе. Вторая строка должна содержать номера даримых бриллиантов.

Бриллианты нумеруются от 1 до n в порядке появление во входных данных.

Если составить подарок принцессе невозможно, то выведите 0 в первой строке вывода.

Примеры

dowry.in	dowry.out
3 6 8 3 10 7 3 8 2	1 2

Задача E. Оппозиция снова с нами!

Имя входного файла:	`org.in`
Имя выходного файла:	`org.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В некоторой стране полиция выявила разветвленную сеть оппозиционной партии. Партия сильно законспирирована и состоит из рядовых членов и руководителей различных уровней. Во главе партии стоит один главный руководитель — лидер партии. До начала арестов приказ лидера может быть доведен до любого члена партии. Все члены партии пронумерованы от 1 до N.

Каждый член партии знает только своего вышестоящего руководителя (ровно одного) и своих непосредственных подчиненных (руководитель не знает подчиненных своего подчиненного и наоборот).

Естественно, что с началом арестов членов партии, она распадется на мелкие, не связанные друг с другом группы.

Полицмейстер уверяет, что группа, состоящая из менее, чем K членов партии, идеологически вырождается и не представляет угрозы для государства.

Стремясь не уронить престиж страны в глазах мирового общественного мнения, полицмейстер поставил задачу «устранить» некоторое количество членов партии так, чтобы от нее остались только идеологически вырождающиеся маленькие группы. При этом у каждого члена партии есть своя стоимость «устранения».

Требуется написать программу, которая бы по входным данным, описывающим структуру подпольной партии, находила минимальную сумму и номера членов партии, которых нужно арестовать.

Формат входных данных

Входной файл содержит четыре строки. В первой записано число K ($1 \leqslant K \leqslant 100$), во второй строке — число N ($1 \leqslant N \leqslant 1000$), определяющее количество членов партии. Третья строка содержит набор из $N - 1$ числа. В этой строке для каждого члена партии, кроме лидера, задается номер его непосредственного руководителя. Номер руководителя всегда меньше, чем номер подчиненного. При этом первое число задает номер руководителя второго члена партии, второе — третьего и так далее. Числа в строке разделяются одним пробелом. Четвертая строка состоит из N чисел — стоимостей «устранения» соответствующего члена партии (стоимости положительны и не превосходят 10000).

Формат выходных данных

В первую строку необходимо вывести необходимую сумму денег, а во вторую — номера членов партии, подлежащих аресту. Эти номера разделяются одним пробелом.

При наличии нескольких решений выведите одно из них.

Примеры

org.in	org.out
3 14 1 1 2 2 3 2 3 6 6 6 7 4 7 1 1 1 1 1 1 1 1 1 1 1 1 1	4 1 6 2 7

Задача F. Распродажа

Имя входного файла:	`sale.in`
Имя выходного файла:	`sale.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

В супермаркете «На троечку» часто происходят распродажи товаров, срок годности которых подходит к концу. Каждый товар привозят в магазин в определенное время, а через некоторое его вывозят из магазина, в связи с окончанием срока годности. Более формально, каждый товар имеет стоимость c_i, время его завоза в магазин a_i и время его вывоза из магазина b_i.

У Иннокентия есть хитрый план похода в магазин. Даже несколько. Каждый план похода в магазин выглядит так: Иннокентий выбирает какое-то время, когда он появится в магазине m_j, время s_j, которое он проведет в магазине среди огромных стеллажей товаров, и сумму денег k_j, которую он рассчитывает потратить. Для каждого плана он хочет узнать, сможет ли он осуществить его, т. е. верно ли, что он сможет во время своего пребывания в магазине купить несколько товаров суммарной стоимостью **ровно** k_j, при этом все выбранные товары должны быть в магазине на протяжении всего пребывания Иннокентия в магазине.

Помогите Иннокентию определить, какие из его планов можно выполнить.

Формат входных данных

В первой строке входных данных содержится число N — общее количество товаров в магазине ($1 \leqslant N \leqslant 800$). Далее содержатся описания товаров, каждый товар описывается тремя целыми числами c_i, a_i, b_i, обозначающими стоимость товара, время его завоза и время его вывоза из магазина ($1 \leqslant c_i \leqslant 1\,000, 1 \leqslant a_i < b_i \leqslant 10^9$).

Далее содержится число M — количество планов Иннокентия ($1 \leqslant M \leqslant 800\,000$). Каждый план описывается тремя целыми числами m_j, k_j, s_j, обозначающими время прихода Иннокентия в магазин, сумму денег, которую он готов потратить в этом плане и длительность его пребывания в магазине ($1 \leqslant m_j \leqslant 10^9, 1 \leqslant k_j \leqslant 100\,000, 0 \leqslant s_j \leqslant 10^9$).

Помните, что это только планы, т. е. ситуация в магазине не меняется вне зависимости от того, может ли Иннокентий осуществить план или нет.

Формат выходных данных

Для каждого плана в отдельной строке выведите «`YES`», если Иннокентий может его осуществить, и «`NO`» в противном случае.

Примеры

sale.in	sale.out
5 6 2 7 5 4 9 1 2 4 2 5 8 1 3 9 5 2 7 1 2 7 2 3 2 0 5 7 2 4 1 5	YES NO YES YES NO

Задача G. Рюкзак-матроид

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

Одна из версий задачи о рюкзаке формулируется следующим образом: дано n предметов, i-й из них имеет вес w_i и стоимость v_i, требуется выбрать множество предметов с суммарным весом, не превышающим c (вместимости рюкзака) и максимальной возможной суммарной стоимостью. Известно, что задача о рюкзаке является NP-полной. Для нее существуют решения с помощью динамического программирования, но они обычно имеют время работы, линейное относительно суммы весов предметов и вместимости рюкзака, которое, таким образом, не является полиномиальным относительно размера входных данных.

Однако бывают случаи, когда задачу о рюкзаке можно так или иначе решить жадно. Один важный пример такого класса задач — если набор весов таков, что множество решений задачи образует матроид.

Матроидом называется пара $\langle X, \mathcal{I} \rangle$, где X — конечное множество, а $\mathcal{I}$ семейство помножеств X, которые называют независимыми. При этом $\mathcal{I}$ должно удовлетворять следующим трем свойствам:

1. $\mathcal{I} \neq \varnothing$;
2. Если $A \in \mathcal{I}$ и $B \subset A$, то $B \in \mathcal{I}$;
3. Если $A, B \in \mathcal{I}$ и $|A| > |B|$, то найдется такой $x \in A \setminus B$, что $B \cup \{x\} \in \mathcal{I}$.

Например, ребра неориентированного графа и семейство их ациклических подмножеств образуют матроид.

Рассмотрим предметы с весами $w_1, w_2, \ldots, w_n$. Пусть X представляет собой множество целых чисел от 1 до n. Будем называть подмножество $\{i_1, i_2, \ldots, i_k\}$ независимым, если $w_{i_1} + w_{i_2} + \ldots + w_{i_k} \leqslant c$. Проверьте, образует ли получившяся пара матроид.

Формат входных данных

Первая строка входного файла содержит число n ($1 \leqslant n \leqslant 50$). Вторая строка содержит n целых чисел $w_1, w_2, \ldots, w_n$ ($1 \leqslant w_i \leqslant 100$). Третья строка содержит число c ($\min w_i \leqslant c \leqslant \sum w_i$).

Формат выходных данных

Выведите «`YES`», если множество решений задачи о рюкзаке образует матроид. В противном случае выведите «`NO`».

Во втором случает выведите на второй строке число 2 или 3 — номер свойства, которое нарушается. Следующие две строки должны содержать контрпример к указанному свойству. Первая из строк должна описывать множество A, а вторая — B. Описание множества должно состоять из числа элементов в множестве и затем списка входящих в него предметов.

Примеры

knapsack.in	knapsack.out
3 1 2 3 4	YES
3 3 4 5 7	NO 3 2 1 2 1 3