

Общая информация по задачам первого тура

Доступ к результатам проверки решений задач во время тура

Во всех задачах вы можете неограниченное число раз запрашивать результат окончательной проверки.

Для каждой подзадачи сведения о том, какая информация о результате окончательной проверки показывается при запросе, указаны в условиях задач. Возможные варианты приведены в следующей таблице.

Результаты	Пояснение
Потестовые	Для каждого теста указан результат запуска на этом тесте
Первая ошибка	Для подзадачи указаны баллы за эту подзадачу. Если баллы равны 0, указан минимальный номер теста, на котором произошла ошибка и тип ошибки
Баллы	Для подзадачи указаны баллы за эту подзадачу

Ограничения

Задача	Тип задачи	Огр. времени	Огр. памяти
1. Программирование квадрокоптеров	Интерактивная	2 секунды	512 МБ
2. Иллюзия сортировки	Стандартная	2 секунды	512 МБ
3. Тигры	Интерактивная	2 секунды	512 МБ
4. Путешествие в Метрополис	Стандартная	4 секунды	512 МБ

Система оценивания

Во всех задачах баллы за каждую подзадачу начисляются в случае успешного прохождения тестов для этой и всех необходимых для неё подзадач.

Замечания

Жюри обращает внимание участников, что производительность подсистем ввода-вывода под разными ОС для одного и того же языка программирования может существенно различаться. В связи с этим жюри рекомендует:

- на языке C++ в задачах с большим объёмом входных данных использовать компилятор GNU C++ под ОС Linux;
- на языке Паскаль в задачах с большим объёмом входных данных использовать компилятор Free Pascal под ОС Linux;
- на языке Java использовать компилятор Java под ОС Windows;
- в остальных случаях использовать ту же ОС, под которой работает участник.

Ввод и вывод

В стандартных задачах разрешается считывать данные как из файла, указанного в условии задачи, так и из стандартного потока ввода. Выходные данные можно выводить как в файл, указанный в условии задачи, так и в стандартный поток вывода.

Исключение: компиляторы C#, Free Pascal под Windows и Pascal ABC. Решения, использующие один из этих компиляторов, могут вводить данные либо из файла, либо из стандартного потока ввода, но обязательно должны выводить данные в стандартный поток вывода. Попытка вывести в файл приведёт к результату проверки «Runtime Error».

В интерактивных задачах при тестировании под ОС Linux не следует выводить в стандартный поток ошибок `stderr`.

Общие замечания по интерактивным задачам

После каждого действия вашей программы выводите перевод строки.

После каждого действия вашей программы делайте сброс потока вывода.

Если вы используете «`writeln`» в Free Pascal или PascalABC, «`cout << ... << endl`» в C++, «`System.out.println`» в Java, «`print`» в Python, «`Console.WriteLine`» в C#, то сброс потока вывода у вас происходит автоматически, дополнительно ничего делать не требуется. Если вы используете другой способ вывода, рекомендуется делать сброс потока вывода. Обратите внимание, что перевод строки надо выводить в любом случае. Для сброса потока вывода можно использовать «`fflush(stdout)`» в C и C++, «`flush(output)`» в Паскале, «`System.out.flush()`» в Java, «`sys.stdout.flush()`» в Python, «`Console.Out.Flush()`» в C#. Обратите внимание, что в Borland Delphi «`flush`» делать обязательно.

Типичные ошибки в интерактивных задачах:

- «**Wrong Answer**» обычно означает, что ваша программа соблюдала протокол, но ответ или промежуточные действия неверны, либо было превышено ограничение на количество запросов.
- «**Presentation Error**» обычно означает, что ваша программа нарушила протокол таким образом, что вывод вашей программы не может быть корректно проинтерпретирован программой жюри, с которой она взаимодействует.
- «**Idleness Limit Exceeded**» означает, что ваша программа ожидает ввода, но данных в стандартном потоке ввода нет. Например,
 - ваша программа ошибочно ожидает ввода, а она должна вывести информацию для программы жюри, либо завершиться;
 - ваша программа не вывела перевод строки или не произвела сброс потока вывода, программа жюри не получила вывод вашей программы и не может выполнить свои действия.
- «**Runtime Error**» редко означает проблемы с интерактивностью и чаще возникает из-за обычных ошибок в программе. Хотя ничего нельзя исключить.

Задача 1. Программирование квадрокоптеров

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Школьники готовятся к участию в соревновании по программированию квадрокоптеров. Квадрокоптер, который используется в соревновании, может выполнять две команды: подняться вверх на 1 метр и опуститься вниз на 1 метр. Команда подъёма обозначается символом «(», а команда спуска — символом «)».

Программа для квадрокоптера представляет собой последовательность команд. Программа считается *корректной*, если, начав её исполнение на уровне земли и выполнив последовательно все команды, квадрокоптер снова оказывается на уровне земли. При этом в процессе выполнения программы квадрокоптер не должен пытаться опуститься ниже уровня земли.

Например, следующие программы являются корректными: «() ()», «((()))». Программа «((((» не является корректной, поскольку квадрокоптер завершает её выполнение на высоте 3 метра над уровнем земли, программа «()) (» также не является корректной, поскольку при выполнении третьей команды квадрокоптер пытается опуститься ниже уровня земли.

Участник соревнования написал корректную программу для квадрокоптера, состоящую из n команд, пронумерованных от 1 до n . Он загрузил её в память квадрокоптера для демонстрации во время соревнования. К сожалению, после загрузки программы в память квадрокоптера участник случайно удалил её на своём компьютере, а квадрокоптер не позволяет выгрузить программу из своей памяти.

К счастью, квадрокоптер поддерживает специальный режим отладки программы. В этом режиме квадрокоптер с загруженной в него программой может отвечать на специальные запросы. Каждый запрос представляет собой два целых числа: l и r , $1 \leq l \leq r \leq n$. В ответ на запрос квадрокоптер сообщает, является ли фрагмент загруженной в него программы, состоящий из команд с l -й по r -ю включительно, корректной программой для квадрокоптера, либо нет. Участник хочет с помощью режима отладки восстановить загруженную в квадрокоптер программу.

Требуется написать программу-решение, которая взаимодействует с программой жюри, моделирующей режим отладки квадрокоптера, и в итоге восстанавливает загруженную в квадрокоптер программу.

Протокол взаимодействия

Это интерактивная задача.

Сначала на вход подаётся целое число n — количество команд в программе квадрокоптера ($2 \leq n \leq 50\,000$).

Для каждого теста жюри зафиксировано число k — максимальное количество запросов. Гарантируется, что k запросов достаточно, чтобы решить задачу. Это число не сообщается программе-решению. Ограничения k в различных подзадачах приведены в таблице системы оценивания. Если программа-решение делает более k запросов к программе жюри, то на этом тесте она получает в качестве результата тестирования «Неверный ответ».

Чтобы сделать запрос, программа-решение должна вывести строку вида «? l r », где l и r — целые положительные числа, задающие фрагмент программы квадрокоптера ($1 \leq l \leq r \leq n$).

В ответ на запрос программы-решения программа жюри подаёт ей на вход либо строку «Yes», либо строку «No», в зависимости от того, является ли запрошенный фрагмент программы квадрокоптера корректной программой.

Если программа-решение определила ответ на задачу, то она должна вывести строку «! $c_1 c_2 \dots c_n$ », где символ c_i задаёт i -ю команду в программе квадрокоптера и равен либо «(», либо «)».

После этого программа-решение должна завершиться.

Гарантируется, что в каждом тесте программа в памяти квадрокоптера является фиксированной корректной программой, которая не меняется в зависимости от запросов, произведённых программой-решением.

Примеры

стандартный ввод	стандартный вывод
4	? 1 4
Yes	? 1 3
No	? 1 2
Yes	? 3 4
Yes	! () ()
6	? 3 4
Yes	? 1 2
No	? 2 5
Yes	! ((()))

Пояснения к примерам

Приведённые примеры иллюстрируют взаимодействие программы-решения с программой жюри «по шагам», для чего в них добавлены дополнительные пустые строки. При реальном тестировании лишние пустые строки вводиться не будут, выводить пустые строки также не требуется.

В первом примере $n = 4$. Единственная возможная корректная программа из двух команд это «()», поэтому из результатов третьего и четвёртого запросов можно сделать вывод, что программа в памяти квадрокоптера — «() ()». Поэтому, в частности, ответ на второй запрос действительно «No», так как фрагмент программы «() (» не является корректной программой: если квадрокоптер исполнит именно эти три команды, он останется на уровне одного метра над землёй.

В втором примере $n = 6$, и произведённых запросов достаточно, чтобы однозначно определить, что программа в памяти квадрокоптера — «((()))».

В тестах из условия $k = 150$, то есть, разрешается произвести не более 150 запросов.

Система оценивания

Подзадача	Баллы	Ограничения		Необходимые подзадачи	Результаты во время тура
		n	k		
1	21	$2 \leq n \leq 16$	$k = 150$		Потестовые
2	28	$2 \leq n \leq 100$	$k = 10\,000$	1	Потестовые
3	26	$2 \leq n \leq 1000$	$k = 10\,000$	1, 2	Потестовые
4	25	$2 \leq n \leq 50\,000$	$k = 100\,000$	1 – 3	Потестовые

Задача 2. Иллюзия сортировки

Имя входного файла: `order.in`
Имя выходного файла: `order.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 512 мегабайт

Известный программист пробует себя в роли иллюзиониста. Его коронный фокус состоит в следующем.

Для заданного массива из n целых неотрицательных чисел a_1, a_2, \dots, a_n он быстро подбирает *магическое* число b . Целое неотрицательное число b называется магическим для массива, если применение операции побитового *исключающего ИЛИ* с этим числом к каждому элементу массива превращает его в отсортированный массив. Иначе говоря,

$$(a_1 \oplus b) \leq (a_2 \oplus b) \leq \dots \leq (a_n \oplus b),$$

где \oplus — операция побитового исключающего ИЛИ.

Чтобы фокус был более эффектным, после предъявления магического числа для заданного массива иллюзионист q раз выполняет следующее действие. Он предлагает зрителям изменить один из элементов массива и после этого снова пытается предъявить магическое число. При этом программист настолько отточил свое мастерство иллюзиониста, что каждый раз предъявляет зрителям *минимальное возможное* магическое число. Иногда фокус не удаётся, так как для полученного массива невозможно подобрать магическое число.

Требуется написать программу, которая по заданному исходно массиву, а также после каждого изменения элемента, вычисляет минимальное магическое число для полученного массива, либо определяет, что такого числа нет.

Замечание

Исключающее ИЛИ — это логическая операция, обозначаемая знаком \oplus , которая задаётся следующей таблицей истинности:

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Определим побитовое исключающее ИЛИ для двух неотрицательных целых чисел x и y . Запишем каждое из целых чисел x и y в двоичной системе счисления, дополнив при необходимости более короткое из чисел ведущими нулями до равной длины. Побитовое исключающее ИЛИ двух целых чисел x и y , обозначаемое также как $x \oplus y$, это целое неотрицательное число, каждый разряд которого в двоичной системе счисления является исключающим ИЛИ соответствующих разрядов чисел x и y . Например, $5 \oplus 22 = 101_2 \oplus 10110_2 = 10011_2 = 19$.

Среди предложенных на олимпиаде языков программирования в языке Паскаль для обозначения исключающего ИЛИ используется оператор «xor», в остальных языках программирования используется оператор «^».

Формат входных данных

Первая строка входных данных содержит целое число n — количество чисел в массиве ($1 \leq n \leq 10^6$).

Вторая строка содержит n целых чисел a_1, a_2, \dots, a_n — элементы массива ($0 \leq a_i < 2^{30}$).

Третья строка содержит целое число q — число изменений элемента массива ($0 \leq q \leq 10^6$).

Следующие q строк содержат по два целых числа p_i и v_i , где p_i — номер элемента массива, который следует заменить ($1 \leq p_i \leq n$), а v_i — новое значение этого элемента ($0 \leq v_i < 2^{30}$).

Формат выходных данных

Выходные данные должны содержать $(q + 1)$ целых чисел b_0, b_1, \dots, b_q , по одному в строке.

Значение b_0 — либо минимальное возможное магическое число для исходного массива, либо -1 , если такого числа не существует.

Для i от 1 до q значение b_i — либо минимальное возможное магическое число для массива после первых i изменений, либо -1 , если такого числа не существует.

Пример

order.in	order.out
3	0
0 1 4	2
3	-1
2 7	4
3 3	
1 4	

Система оценивания

Подзадача	Баллы	Ограничения			Необх. подзадачи	Результаты во время тура
		n	q	a_i, v_i		
1	30	$1 \leq n \leq 500$	$0 \leq q \leq 500$	$0 \leq a_i, v_i < 2^9$		Потестовые
2	29	$1 \leq n \leq 1000$	$0 \leq q \leq 1000$	$0 \leq a_i, v_i < 2^{30}$	1	Потестовые
3	21	$1 \leq n \leq 10^5$	$0 \leq q \leq 10^5$	$0 \leq a_i, v_i < 2^{30}$	1, 2	Баллы
4	20	$1 \leq n \leq 10^6$	$0 \leq q \leq 10^6$	$0 \leq a_i, v_i < 2^{30}$	1 – 3	Баллы

Задача 3. Тигры

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

В заповеднике живут q тигров. Чтобы следить за положением тигров на территории заповедника, используются ошейники с радиомаяком. Ошейник у каждого тигра имеет радиомаяк с уникальным сигналом. Система обнаружения настраивается на приём сигнала радиомаяка от i -го тигра последовательно для i от 1 до q .

Для приёма сигнала на территории заповедника установлено n приёмников в точках с координатами $(x_1, y_1), \dots, (x_n, y_n)$. Система обнаружения позволяет сотруднику заповедника за один запрос выбрать любые m ($3 \leq m \leq n$) приёмников. Выбранные приёмники должны являться вершинами выпуклого многоугольника. Система определяет, находится ли радиомаяк i -го тигра внутри этого многоугольника.

Сотрудник заповедника должен *локализовать* положение каждого тигра. Положение i -го тигра считается *локализованным*, если удалось определить такое множество приёмников, являющихся вершинами выпуклого многоугольника, что внутри этого многоугольника находится тигр, но нет других приёмников.

Для того, чтобы локализовать положение каждого из тигров, сотруднику разрешается сделать не более k запросов.

После того как положение i -го тигра локализовано, система автоматически переходит к приёму сигналов от следующего тигра, пока положение всех q тигров не будет локализовано.

Гарантируется, что никакие три приёмника не лежат на одной прямой, и ни один тигр не находится на прямой, проходящей через два приёмника. Гарантируется, что существует хотя бы один выпуклый многоугольник с вершинами в приёмниках, внутри которого находится тигр.

Требуется написать программу, которая взаимодействует с программой жюри и локализует положение каждого тигра.

Протокол взаимодействия

Это интерактивная задача.

Сначала на вход подаётся информация об установленных в заповеднике приёмниках и количестве тигров.

Первая строка входных данных содержит целое число n — количество приёмников ($3 \leq n \leq 5000$). Последующие n строк описывают координаты приёмников, j -я из этих строк содержит два целых числа x_j и y_j — координаты j -го приёмника ($-10^9 \leq x_j, y_j \leq 10^9$). Следующая строка содержит число целое число q — количество тигров ($1 \leq q \leq 2000$).

Для локализации положения тигров необходимо выполнять запросы к системе обнаружения, роль которой выполняет программа жюри.

Для каждого теста зафиксировано число k — максимальное количество запросов к системе обнаружения для локализации положения одного тигра. Гарантируется, что k запросов достаточно, чтобы решить задачу для соответствующих данных. Это число не сообщается программе-решению, но ограничения на него в различных подзадачах приведены в таблице системы оценивания. Если программа-решение делает более k запросов для определения местоположения одного из тигров, на этом тесте она получает в качестве результата тестирования «Неверный ответ».

Запрос к системе обнаружения начинается с символа «?», за которым следует целое число m — количество выбранных в запросе приёмников ($3 \leq m \leq n$), и m различных целых чисел p_i — номера приёмников, перечисленные в порядке обхода многоугольника по или против часовой стрелки ($1 \leq p_i \leq n$).

В ответ программа получает строку «Yes», если тигр находится внутри многоугольника, образованного приёмниками с номерами p_1, \dots, p_m , и строку «No» в противном случае.

После того, как положение тигра локализовано, программа-решение должна вывести строку, начинающуюся с символа «!», за которым следует целое число m — количество выбранных приёмников ($3 \leq m \leq n$), и m различных целых чисел p_i — номера приёмников, перечисленные в порядке

обхода многоугольника по или против часовой стрелки ($1 \leq p_i \leq n$). Эта строка означает, что внутри выпуклого многоугольника, образованного приёмниками с номерами p_1, \dots, p_m , находится тигр и нет других приёмников.

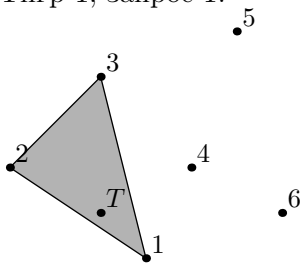
Ответное сообщение от программы жюри отсутствует, и программа-решение должна немедленно приступить к поиску следующего тигра. Локализовав положение тигра с номером q , программа-решение должна завершить работу.

Тигры не перемещаются во время работы системы обнаружения. Координаты тигров в каждом тесте фиксированы и не меняются в процессе тестирования.

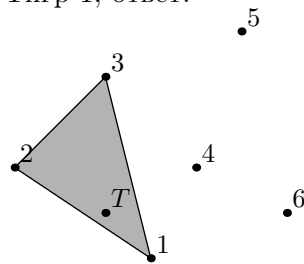
Если существует несколько правильных многоугольников, локализующих положение тигра, можно вывести любой из них.

На рисунке продемонстрирована процедура локализации положения каждого из тигров из приведенного ниже примера.

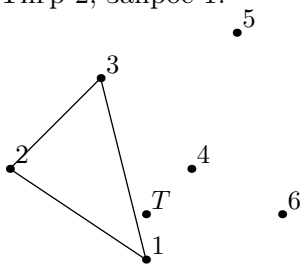
Тигр 1, запрос 1:



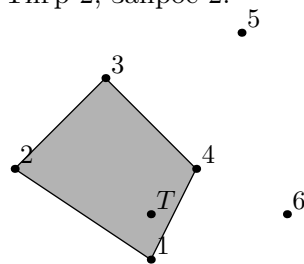
Тигр 1, ответ:



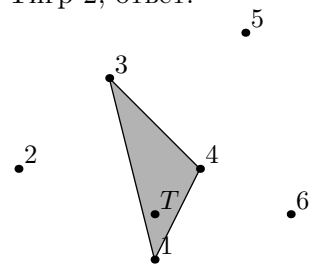
Тигр 2, запрос 1:



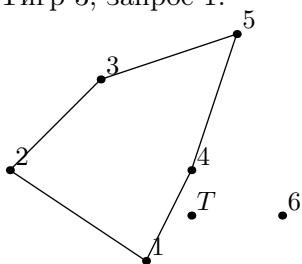
Тигр 2, запрос 2:



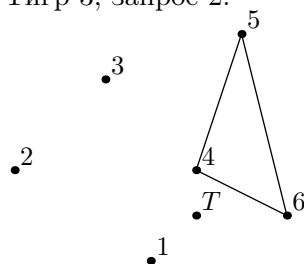
Тигр 2, ответ:



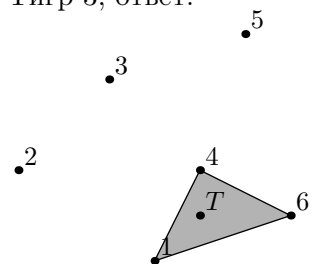
Тигр 3, запрос 1:



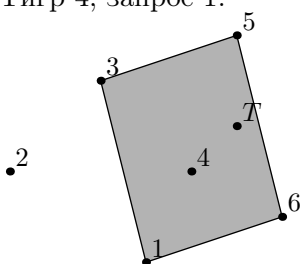
Тигр 3, запрос 2:



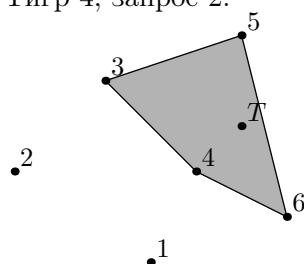
Тигр 3, ответ:



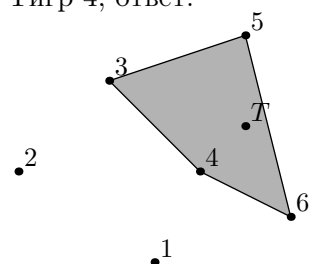
Тигр 4, запрос 1:



Тигр 4, запрос 2:



Тигр 4, ответ:



Пример

стандартный ввод	стандартный вывод
6	
3 0	
0 2	
2 4	
4 2	
5 5	
6 1	
4	
Yes	? 3 1 2 3
	! 3 1 2 3
	? 3 1 2 3
No	
	? 4 1 2 3 4
Yes	
	! 3 4 3 1
	? 5 2 3 5 4 1
No	
	? 3 4 5 6
No	
	! 3 1 4 6
	? 4 1 3 5 6
Yes	
	? 4 4 3 5 6
Yes	
	! 4 4 3 5 6

Замечание

Приведённые примеры иллюстрируют взаимодействие программы-решения с программой жюри «по шагам», для чего в них добавлены дополнительные пустые строки. При реальном тестировании лишние пустые строки вводиться не будут, выводить пустые строки также не требуется.

Система оценивания

Подзадача	Баллы	Ограничения			Необходимые подзадачи	Результаты во время тура
		n	q	k		
1	15	$3 \leq n \leq 6$	$1 \leq q \leq 50$	$k = 4000$		Потестовые
2	17	$3 \leq n \leq 20$	$1 \leq q \leq 50$	$k = 4000$	1	Потестовые
3	9	$3 \leq n \leq 60$	$1 \leq q \leq 400$	$k = 4000$	1, 2	Потестовые
4	9	$3 \leq n \leq 300$	$1 \leq q \leq 1000$	$k = 600$	1 – 3	Потестовые
5	10	$3 \leq n \leq 5000$	$1 \leq q \leq 10$	$k = 10\,000$		Потестовые
6	10	$3 \leq n \leq 300$	$1 \leq q \leq 1000$	$k = 250$	1 – 4	Потестовые
7	10	$3 \leq n \leq 1000$	$1 \leq q \leq 1000$	$k = 200$	1 – 4, 6	Потестовые
8	10	$3 \leq n \leq 1000$	$1 \leq q \leq 2000$	$k = 60$	1 – 4, 6, 7	Потестовые
9	10	$3 \leq n \leq 2500$	$1 \leq q \leq 2000$	$k = 40$	1 – 4, 6 – 8	Потестовые

Задача 4. Путешествие в Метрополис

Имя входного файла:	<code>trains.in</code>
Имя выходного файла:	<code>trains.out</code>
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мегабайт

Путешествие по стране никогда не бывает простым, особенно когда не существует прямого сообщения между городами. Группа туристов хочет добраться в город Метрополис, используя сеть железных дорог, которая соединяет n городов, пронумерованных от 1 до n . Город, из которого выезжает группа, имеет номер 1, Метрополис имеет номер n .

На железной дороге постоянно функционируют m маршрутов поездов. Каждый маршрут определяется последовательностью городов, перечисленных в том порядке, в каком их проезжает поезд, обслуживающий этот маршрут. В каждом маршруте для каждой пары соседних городов задано время, за которое поезд этого маршрута проезжает перегон между этими городами. При этом поезда разных маршрутов могут проезжать один и тот же перегон за разное время.

По пути в Метрополис группа может садиться на поезд и сходить с поезда в любом городе маршрута, не обязательно в начальном или конечном. При этом, можно сойти с поезда маршрута i , пересесть на поезд маршрута j , возможно сделать еще несколько пересадок, а потом вновь сесть в поезд того же маршрута i .

Туристы предъявляют высокие требования к выбору способа проезда в Метрополис.

Во-первых, суммарное время, проведенное в поездах, должно быть минимальным.

Во-вторых, среди всех способов с минимальным временем нахождения в поездах предпочтительным является тот способ, для которого сумма квадратов промежутков времени, непрерывно проведенных в поезде между двумя пересадками, максимальна. Назовём эту сумму *качеством путешествия*.

Время, проведенное вне поездов, не учитывается.

Требуется написать программу, которая по описаниям имеющихся маршрутов поездов определит минимальное время, которое группе туристов придется провести в поездах, а также максимальное качество путешествия с таким временем.

Формат входных данных

В первой строке входных данных заданы два целых числа ($2 \leq n \leq 10^6$, $1 \leq m \leq 10^6$) — количество городов и количество маршрутов соответственно.

Далее в m строках содержится описание маршрутов.

Описание каждого маршрута начинается с целого числа s_i — количество перегонов в маршруте с номером i ($1 \leq s_i \leq 10^6$). Далее следуют $(2s_i + 1)$ целых чисел, описывающих города маршрута и время проезда перегона между соседними городами маршрута, в следующем порядке: $v_{i,1}, t_{i,1}, v_{i,2}, t_{i,2}, v_{i,3}, \dots, t_{i,s_i}, v_{i,s_i+1}$, где $v_{i,j}$ — номер j -го города маршрута, $t_{i,j}$ — время проезда перегона между j -м и $(j + 1)$ -м городом ($1 \leq v_{i,j} \leq n$, $1 \leq t_{i,j} \leq 1000$).

Гарантируется, что $s_1 + s_2 + \dots + s_m \leq 10^6$. Никакие два города в описании маршрута не совпадают. Гарантируется, что с помощью имеющихся маршрутов можно добраться из города с номером 1 в город с номером n .

Формат выходных данных

Выходные данные должны содержать два целых числа — минимальное суммарное время, которое придется провести в поездах, и максимальное качество пути с таким временем.

Примеры

trains.in	trains.out
2 1 1 1 3 2	3 9
5 2 4 1 3 2 3 3 5 5 10 4 3 4 2 2 1 3 4 1	9 35
5 2 3 1 1 2 2 3 3 4 3 2 2 3 3 4 4 5	10 82

Замечание

В первом примере группа туристов отправится прямым маршрутом в Метрополис.

Во втором примере не оптимально проехать напрямую по первому маршруту, так как время в поезде при этом не будет минимальным возможным. Поэтому они отправятся на поезде по маршруту 1 из города 1 в город 2, затем на поезде по маршруту 2 из города 2 в город 3, а затем снова на поезде по маршруту 1 из города 3 в город 5. При этом сумма квадратов промежутков времени, проведенных в поездах между пересадками, равна $3^2 + 1^2 + 5^2 = 35$.

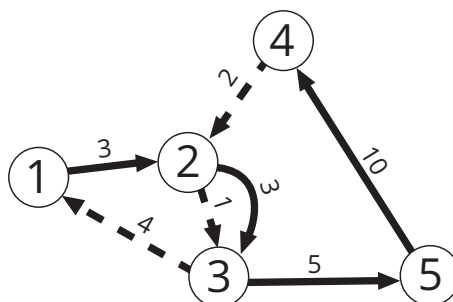


Иллюстрация ко второму примеру.

В третьем примере добраться из города 1 в город 4 за минимальное время можно, пересаживаясь с маршрута 1 на маршрут 2 в любом из городов 2, 3 или 4. Максимальное качество путешествия достигается при пересадке в городе 2: $1^2 + 9^2 = 82$.

Обратите внимание, что второй и третий примеры не удовлетворяют ограничениям первой и второй подзадачи, решение будет протестировано на этих подзадачах, если оно пройдет первый тест из примера. Все тесты из примера подходят под ограничения подзадач 3 – 7, решение будет проверяться на тестах этих подзадач только в случае прохождения всех тестов из примера.

Система оценивания

Подзадача	Баллы	Ограничения		Необходимые подзадачи	Результаты во время тура
		n	s_i		
1	10	$n \leq 10$	$\sum s_i \leq 20, s_i = 1$		Потестовые
2	10	$n \leq 1000$	$\sum s_i \leq 1000, s_i = 1$	1	Потестовые
Подзадачи, начиная с 3, тестируются только при прохождении <i>всех</i> тестов из примера					
3	17	$n \leq 1000$	$\sum s_i \leq 1000$	1, 2	Потестовые
4	17	$n \leq 1000$	$\sum s_i \leq 100000$	1 – 3	Потестовые
5	19	$n \leq 10000$	$\sum s_i \leq 200000$	1 – 4	Баллы
6	19	$n \leq 200000$	$\sum s_i \leq 200000$	1 – 5	Баллы
7	8	$n \leq 10^6$	$\sum s_i \leq 10^6$	1 – 6	Баллы

Общая информация по задачам второго тура

Доступ к результатам проверки решений задач во время тура

Во всех задачах вы можете неограниченное число раз запрашивать результат окончательной проверки.

Для каждой подзадачи сведения о том, какая информация о результате окончательной проверки показывается при запросе, указаны в условиях задач. Возможные варианты приведены в следующей таблице.

Результаты	Пояснение
Потестовые	Для каждого теста указан результат запуска на этом тесте
Первая ошибка	Для подзадачи указаны баллы за эту подзадачу. Если баллы равны 0, указан минимальный номер теста, на котором произошла ошибка и тип ошибки
Баллы	Для подзадачи указаны баллы за эту подзадачу

Ограничения

Обратите внимание, что ограничение по памяти в задаче 7 составляет **128 МБ**.

Задача	Тип задачи	Огр. времени	Огр. памяти
5. Накопитель	Стандартная	2 секунды	512 МБ
6. Серверы на Меркурии	Стандартная	1 секунда	512 МБ
7. Антивещество	Стандартная	2 секунды	128 МБ
8. Траектория обучения	Стандартная	2 секунды	512 МБ

Система оценки

Во всех задачах баллы за каждую подзадачу начисляются в случае успешного прохождения тестов для этой и всех необходимых для неё подзадач.

Замечания

Жюри обращает внимание участников, что производительность подсистем ввода-вывода под разными ОС для одного и того же языка программирования может существенно различаться. В связи с этим жюри рекомендует:

- на языке C++ в задачах с большим объёмом входных данных использовать компилятор GNU C++ под ОС Linux;
- на языке Паскаль в задачах с большим объёмом входных данных использовать компилятор Free Pascal под ОС Linux;
- на языке Java использовать компилятор Java под ОС Windows;
- в остальных случаях использовать ту же ОС, под которой работает участник.

Ввод и вывод

В стандартных задачах разрешается считывать данные как из файла, указанного в условии задачи, так и из стандартного потока ввода. Выходные данные можно выводить как в файл, указанный в условии задачи, так и в стандартный поток вывода.

Исключение: компиляторы C#, Free Pascal под Windows и Pascal ABC. Решения, использующие один из этих компиляторов, могут вводить данные либо из файла, либо из стандартного потока ввода, но обязательно должны выводить данные в стандартный поток вывода. Попытка вывести в файл приведёт к результату проверки «Runtime Error».

Задача 5. Накопитель

Имя входного файла: `storage.in`
 Имя выходного файла: `storage.out`
 Ограничение по времени: 2 секунды
 Ограничение по памяти: 512 мегабайт

Исследуется новое цифровое устройство для хранения информации. Информация на устройстве хранится в виде последовательности *ячеек*, каждая из которых находится в одном из двух состояний, обозначаемых символами «+» и «-», и, таким образом, хранит один бит информации.

Назовём *фрагментом* группу соседних ячеек с одинаковым состоянием, слева от которой либо нет ячеек, либо находится ячейка в противоположном состоянии, и справа — либо нет ячеек, либо находится ячейка в противоположном состоянии.

Операция записи позволяет выбрать любую пару соседних фрагментов разной длины и изменить состояние всех ячеек более короткого фрагмента на противоположное, объединяя таким образом два или три соседних фрагмента в один.

Требуется написать программу, которая по заданной исходной и итоговой последовательностям состояний ячеек определяет, можно ли из исходной последовательности получить итоговую с помощью последовательных операций записи.

Формат входных данных

Первая строка входных данных содержит целое положительное число q — количество тестов.

Каждая из следующих q строк содержит s_i, t_i — непустые последовательности символов «+» и «-» одинаковой длины, разделённые одним пробелом. Эта строка означает, что в тесте номер i из исходной последовательности состояний ячеек s_i требуется получить итоговую последовательность t_i .

Формат выходных данных

Выходные данные должны содержать q строк, где i -я строка равна «Yes», если из исходной последовательности состояний ячеек s_i можно получить итоговую последовательность t_i , или «No» в противном случае.

Примеры

storage.in	storage.out
3 +- + +-+ + +-+--+	Yes No Yes
3 +-+--+ + +-+--+ + -+- -+-	Yes No Yes

Система оценки

Подзадача	Баллы	Ограничения		Необх. подзадачи	Результаты во время тура
		Сумма длин s_i	t_i		
1	20	$\sum s_i \leq 16$	t_i состоит из символов «+»	–	Потестовые
2	30	$\sum s_i \leq 1000$	t_i состоит из символов «+»	1	Потестовые
3	20	$\sum s_i \leq 10^6$	t_i состоит из символов «+»	1, 2	Потестовые
4	20	$\sum s_i \leq 1000$		1, 2	Потестовые
5	10	$\sum s_i \leq 10^6$		1 – 4	Потестовые

Задача 6. Серверы на Меркурии

Имя входного файла:	<code>servers.in</code>
Имя выходного файла:	<code>servers.out</code>
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Компьютерная система управления станциями на Меркурии состоит из n серверов, пронумерованных от 1 до n . Серверы соединены $(n - 1)$ двусторонними каналами связи, i -й из которых соединяет i -й и $(i + 1)$ -й серверы.

С Земли необходимо передать *пакет обновления* программного обеспечения для компьютерной системы управления. Пакет необходимо установить на каждый сервер. Стоимость передачи пакета обновления с Земли на Меркурий очень высока, поэтому с Земли пакет обновления передаётся только на один сервер. Затем пакет необходимо передать на все остальные серверы по каналам связи, возможно, через другие серверы.

Из-за высокой солнечной радиации на Меркурии передавать пакет обновления по каналам связи можно только в некоторые промежутки времени. Для i -го канала связи известен промежуток времени $[l_i, r_i]$, во время которого возможна передача пакета по этому каналу. Пакет передаётся по любому каналу связи мгновенно.

Пакет обновления, переданный на j -й сервер, немедленно устанавливается и помещается в специальный буфер памяти, из которого он может быть передан на другие серверы. Пакет находится в буфере памяти j -го сервера в течение t_j секунд с момента его получения. Если в момент нахождения пакета в буфере памяти сервера появляется возможность передать его по каналу связи на соседний сервер, на котором пакет обновления пока не установлен, то он немедленно передаётся по этому каналу связи.

Поскольку пакет содержит важные обновления, требуется начать его распространение как можно раньше.

Требуется написать программу, которая для всех i от 1 до n определяет, возможно ли установить пакет обновления на все серверы, передав его с Земли на i -й сервер. Если это возможно, то необходимо определить, в какой минимальный неотрицательный момент времени можно установить пакет на этот сервер, чтобы в результате обновление оказалось установлено на всех серверах.

Формат входных данных

Первая строка входных данных содержит n — количество серверов ($1 \leq n \leq 200\,000$).

Вторая строка содержит n целых чисел t_1, t_2, \dots, t_n , где t_j — время нахождения пакета в буфере памяти j -го сервера ($0 \leq t_j \leq 10^9$).

Следующие $(n - 1)$ строк описывают каналы связи. Для описания i -го канала задаются два целых числа l_i и r_i — границы промежутка времени, на протяжении которого возможна передача пакета по этому каналу ($0 \leq l_i \leq r_i \leq 10^9$).

Формат выходных данных

Выходные данные должны содержать n целых чисел a_1, a_2, \dots, a_n .

Число a_i должно быть равно такому минимальному неотрицательному моменту времени, что при установке пакета обновления на i -й сервер в момент a_i , пакет будет в итоге установлен на всех серверах. Если такого момента времени для i -го сервера не существует, необходимо вывести $a_i = -1$.

Примеры

servers.in	servers.out
1 10	0
2 3 5 6 8	3 1
3 1 2 4 7 10 3 5	-1 5 5
4 1 0 3 2 4 6 5 5 7 10	5 5 4 -1

Замечание

В первом примере имеется всего один сервер, минимальное подходящее время, в которое можно установить на него обновление — 0.

Во втором примере есть два сервера, передать обновление между которыми можно в промежуток от 6 до 8. Первый сервер хранит обновление в буфере 3 единицы времени, а второй — 5 единиц времени. Если отправить обновление первому серверу в момент 3, то он передаст его второму серверу в момент 6. Аналогично если отправить обновление второму серверу в момент 1, то он передаст его первому серверу в момент 6.

В третьем примере нельзя передать обновление первому серверу так, чтобы оно передалось третьему серверу, так как канал 2–3 закрывается до того, как открывается канал 1–2. Можно отправить обновление второму или третьему серверу в момент 5. В этот момент канал 2–3 открыт, поэтому его сразу получают второй и третий серверы. В момент 7, когда откроется канал 1–2 обновление ещё будет находиться в буфере второго сервера, и передастся первому серверу.

В четвёртом примере второй сервер хранит пакет 0 единиц времени, а канал 2–3 открыт в промежуток 5–5. Чтобы передать обновление через второй сервер к третьему серверу, оно должно попасть ко второму серверу в момент 5. Если же мы хотим отправить обновление третьему серверу, то это можно сделать в момент 4, при этом оно будет храниться до момента 7 и будет в итоге установлено на все серверы.

Система оценки

Подзадача	Баллы	Ограничения			Необх. подзадачи	Результаты во время тура
		n	t_i	r_i		
1	20	$1 \leq n \leq 500$	$0 \leq t_i \leq 500$	$0 \leq r_i \leq 500$	–	Первая ошибка
2	10	$1 \leq n \leq 5000$	$t_i = 5000$	$0 \leq r_i \leq 5000$	–	Первая ошибка
3	10	$1 \leq n \leq 5000$	$0 \leq t_i \leq 5000$	$r_i = 5000$	–	Первая ошибка
4	10	$1 \leq n \leq 5000$	$0 \leq t_i \leq 5000$	$0 \leq r_i \leq 5000$	1–3	Первая ошибка
5	15	$1 \leq n \leq 200\,000$	$t_i = 10^9$	$0 \leq r_i \leq 10^9$	2	Первая ошибка
6	15	$1 \leq n \leq 200\,000$	$0 \leq t_i \leq 10^9$	$r_i = 10^9$	3	Первая ошибка
7	20	$1 \leq n \leq 200\,000$	$0 \leq t_i \leq 10^9$	$0 \leq r_i \leq 10^9$	1–6	Первая ошибка

Задача 7. Антивещество

Имя входного файла: `anti.in`
Имя выходного файла: `anti.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 128 мегабайт

Компания тестирует технологию получения антивещества, используемого в качестве топлива в межпланетном звездолёте. Антивещество получается в результате специальных экспериментов в реакторе.

Известно n типов экспериментов, приводящих к получению антивещества. В результате проведения эксперимента i -го типа в выходной контейнер реактора добавляется от l_i до r_i граммов антивещества. Из соображений безопасности запрещается накапливать в контейнере более a граммов антивещества.

Затраты на проведение эксперимента i -го типа составляют c_i , а стоимость одного грамма полученного антивещества составляет 10^9 .

Если после проведения экспериментов в контейнере образовалось t граммов антивещества, а суммарные затраты на проведение экспериментов в реакторе составили s , то прибыль определяется по формуле $(t \cdot 10^9 - s)$. Компании необходимо разработать стратегию проведения экспериментов, позволяющую максимизировать прибыль, которую можно гарантированно получить.

В зависимости от результатов предыдущих экспериментов стратегия определяет, эксперимент какого типа следует провести, или решает прекратить дальнейшее выполнение экспериментов. Стратегия позволяет *гарантированно получить* прибыль x , если при любых результатах проведения экспериментов: во-первых, в контейнере реактора оказывается не более a граммов антивещества, во-вторых, прибыль составит не менее x .

Например, пусть возможен только один тип эксперимента, порождающий от 4 до 6 граммов антивещества, затраты на его проведение равны 10, а вместимость контейнера составляет 17 граммов. Тогда после двукратного проведения эксперимента в контейнере может оказаться от 8 до 12 граммов антивещества. Если получилось 12 граммов, то больше проводить эксперимент нельзя, так как в случае получения 6 граммов антивещества контейнер может переполниться. В остальных случаях можно провести эксперимент в третий раз и получить от 12 до 17 граммов антивещества. В худшем случае придётся провести эксперимент трижды, затратив в сумме 30, прибыль составит $(12 \cdot 10^9 - 30) = 11\,999\,999\,970$.

Требуется написать программу, которая определяет максимальную прибыль x , которую гарантированно можно получить.

Формат входных данных

Первая строка входных данных содержит два целых числа: n — количество типов экспериментов и a — максимально допустимое количество антивещества в контейнере ($1 \leq n \leq 100$, $1 \leq a \leq 2\,000\,000$).

Следующие n строк содержат по три целых числа l_i , r_i и c_i — минимальное и максимальное количество антивещества, получаемое в результате эксперимента типа i , и затраты на эксперимент этого типа, соответственно ($1 \leq l_i \leq r_i \leq a$, $1 \leq c_i \leq 100$).

Формат выходных данных

Выходные данные должны содержать одно целое число — максимальную прибыль x , которую гарантированно можно получить.

Примеры

<code>anti.in</code>	<code>anti.out</code>
1 17 4 6 10	11999999970
2 11 2 2 100 3 5 5	9999999890

Система оценки

Подзадача	Баллы	Ограничения			Необх. подзадачи	Результаты во время тура
		n	a	Доп. ограничения		
1	10	$n = 1$	$1 \leq a \leq 1\,000$			Потестовые
2	10	$1 \leq n \leq 10$	$1 \leq a \leq 1\,000$	$l_i = r_i$		Потестовые
3	20	$1 \leq n \leq 10$	$1 \leq a \leq 1\,000$		1, 2	Потестовые
4	20	$1 \leq n \leq 100$	$1 \leq a \leq 50\,000$		1 – 3	Потестовые
5	4	$1 \leq n \leq 100$	$1 \leq a \leq 100\,000$		1 – 4	Баллы
6	4	$1 \leq n \leq 100$	$1 \leq a \leq 200\,000$		1 – 5	Баллы
7	4	$1 \leq n \leq 100$	$1 \leq a \leq 300\,000$		1 – 6	Баллы
8	4	$1 \leq n \leq 100$	$1 \leq a \leq 400\,000$		1 – 7	Баллы
9	4	$1 \leq n \leq 100$	$1 \leq a \leq 500\,000$		1 – 8	Баллы
10	4	$1 \leq n \leq 100$	$1 \leq a \leq 800\,000$		1 – 9	Баллы
11	4	$1 \leq n \leq 100$	$1 \leq a \leq 1\,100\,000$		1 – 10	Баллы
12	4	$1 \leq n \leq 100$	$1 \leq a \leq 1\,400\,000$		1 – 11	Баллы
13	4	$1 \leq n \leq 100$	$1 \leq a \leq 1\,700\,000$		1 – 12	Баллы
14	4	$1 \leq n \leq 100$	$1 \leq a \leq 2\,000\,000$		1 – 13	Баллы

Задача 8. Траектория обучения

Имя входного файла:	internship.in
Имя выходного файла:	internship.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Победитель студенческой олимпиады получил предложения о стажировке от двух университетов. При подготовке планов обучения он узнал рейтинг качества преподавания каждой дисциплины в этих университетах.

Программа обучения первого университета состоит из последовательности перечисленных в хронологическом порядке n различных дисциплин a_1, a_2, \dots, a_n , имеющих рейтинги x_1, x_2, \dots, x_n соответственно. Программа обучения второго университета состоит из последовательности перечисленных в хронологическом порядке m различных дисциплин b_1, b_2, \dots, b_m , имеющих рейтинги y_1, y_2, \dots, y_m соответственно.

Студент имеет возможность составить план обучения в первом университете таким образом, чтобы изучить дисциплины на позициях учебной программы с l_a по r_a включительно ($1 \leq l_a \leq r_a \leq n$), либо отказаться от стажировки в первом университете. Аналогично он может составить план обучения во втором университете таким образом, чтобы изучить дисциплины на позициях учебной программы с l_b по r_b включительно ($1 \leq l_b \leq r_b \leq m$), либо отказаться от стажировки во втором университете.

Изучать одну и ту же дисциплину дважды в разных университетах не имеет смысла, поэтому все дисциплины в двух выбранных планах обучения должны быть различны.

Требуется написать программу, которая определит планы обучения студента таким образом, чтобы получить наибольшую возможную сумму рейтингов изучаемых дисциплин.

Формат входных данных

Первая строка входных данных содержит целые числа n и m — количество дисциплин в программах обучения первого и второго университетов ($1 \leq n, m \leq 500\,000$).

Вторая строка входных данных содержит n целых чисел a_i — дисциплины, входящие в программу обучения первого университета, перечисленные в хронологическом порядке ($1 \leq a_i \leq n + m$).

Третья строка входных данных содержит n целых чисел x_i — рейтинги дисциплин, входящих в программу обучения первого университета, перечисленные том же порядке, что и дисциплины a_i ($1 \leq x_i \leq 10^9$).

Четвёртая строка входных данных содержит m целых чисел b_i — дисциплины, входящие в программу обучения второго университета, перечисленные в хронологическом порядке ($1 \leq b_i \leq n + m$).

Пятая строка входных данных содержит m целых чисел y_i — рейтинги дисциплин, входящих в программу обучения второго университета, перечисленные том же порядке, что и дисциплины b_i ($1 \leq y_i \leq 10^9$).

Формат выходных данных

Первая строка выходных данных должна содержать целое число r — наибольшую возможную сумму рейтингов дисциплин.

Вторая строка выходных данных должна содержать целые числа l_a, r_a — позиции в учебной программе первой и последней дисциплин, входящих в план обучения в первом университете, либо «0 0», если студент отказался от стажировки в первом университете.

Третья строка выходных данных должна содержать целые числа l_b, r_b — позиции в учебной программе первой и последней дисциплин, входящих в план обучения во втором университете, либо «0 0», если студент отказался от стажировки во втором университете.

Если возможных правильных ответов несколько, разрешается вывести любой из них.

Примеры

internship.in	internship.out
7 5 3 1 4 8 6 9 2 2 7 4 10 1 5 3 9 2 11 3 8 3 5 3 4 12	39 2 6 2 4
2 3 1 2 1 4 2 3 1 17 2 15	34 0 0 1 3
3 3 4 2 1 10 1 2 5 4 2 1 2 9	19 1 1 3 3

Замечание

В первом тесте из условия приведённые планы обучения в университетах приводят к суммарному рейтингу дисциплин $(7 + 4 + 10 + 1 + 5) + (5 + 3 + 4) = 27 + 12 = 39$. Если бы студент выбрал только вторую и третью дисциплины в первом университете и весь курс обучения во втором университете, суммарный рейтинг дисциплин был бы $(7 + 4) + (3 + 5 + 3 + 4 + 12) = 11 + 27 = 38$.

Во втором тесте из условия первая и третья дисциплины во втором университете имеют настолько высокий рейтинг по сравнению с соответствующими дисциплинами первого университета, что наиболее выгодный вариант — пройти целиком стажировку во втором университете и отказаться от стажировки в первом университете.

Система оценки

Подзадача	Баллы	Ограничения	Необходимые подзадачи	Результаты во время тура
		n, m		
1	10	$1 \leq n, m \leq 50$	–	Потестовые
2	10	$1 \leq n, m \leq 100$	1	Потестовые
3	10	$1 \leq n, m \leq 300$	1, 2	Потестовые
4	10	$1 \leq n, m \leq 500$	1 – 3	Потестовые
5	10	$1 \leq n, m \leq 2000$	1 – 4	Потестовые
6	5	$1 \leq n, m \leq 5000$	1 – 5	Первая ошибка
7	5	$1 \leq n, m \leq 10\,000$	1 – 6	Первая ошибка
8	10	$1 \leq n, m \leq 30\,000$	1 – 7	Первая ошибка
9	10	$1 \leq n, m \leq 100\,000$	1 – 8	Первая ошибка
10	10	$1 \leq n, m \leq 250\,000$	1 – 9	Первая ошибка
11	10	$1 \leq n, m \leq 500\,000$	1 – 10	Первая ошибка