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	$\sum \lvert s_i \rvert \leqslant 16$	t_i состоит из символов «+»	–	Потестовые
2	30	$\sum \lvert s_i \rvert \leqslant 1000$	t_i состоит из символов «+»	1	Потестовые
3	20	$\sum \lvert s_i \rvert \leqslant 10^6$	t_i состоит из символов «+»	1, 2	Потестовые
4	20	$\sum \lvert s_i \rvert \leqslant 1000$		1, 2	Потестовые
5	10	$\sum \lvert s_i \rvert \leqslant 10^6$		1 – 4	Потестовые

Задача 6. Серверы на Меркурии

Имя входного файла:	`servers.in`
Имя выходного файла:	`servers.out`
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 ($1 \leqslant n \leqslant 200\,000$).

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

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 ($0 \leqslant l_i \leqslant r_i \leqslant 10^9$).

Формат выходных данных

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

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 получат 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	$1 \leqslant n \leqslant 500$	$0 \leqslant t_i \leqslant 500$	$0 \leqslant r_i \leqslant 500$	–	Первая ошибка
2	10	$1 \leqslant n \leqslant 5000$	$t_i = 5000$	$0 \leqslant r_i \leqslant 5000$	–	Первая ошибка
3	10	$1 \leqslant n \leqslant 5000$	$0 \leqslant t_i \leqslant 5000$	$r_i = 5000$	–	Первая ошибка
4	10	$1 \leqslant n \leqslant 5000$	$0 \leqslant t_i \leqslant 5000$	$0 \leqslant r_i \leqslant 5000$	1–3	Первая ошибка
5	15	$1 \leqslant n \leqslant 200\,000$	$t_i = 10^9$	$0 \leqslant r_i \leqslant 10^9$	2	Первая ошибка
6	15	$1 \leqslant n \leqslant 200\,000$	$0 \leqslant t_i \leqslant 10^9$	$r_i = 10^9$	3	Первая ошибка
7	20	$1 \leqslant n \leqslant 200\,000$	$0 \leqslant t_i \leqslant 10^9$	$0 \leqslant r_i \leqslant 10^9$	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 ($1 \leqslant n \leqslant 100$, $1 \leqslant a \leqslant 2\,000\,000$).

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 ($1 \leqslant l_i \leqslant r_i \leqslant a$, $1 \leqslant c_i \leqslant 100$).

Формат выходных данных

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

Примеры

anti.in	anti.out
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	$n = 1$	$1 \leqslant a \leqslant 1\,000$			Потестовые
2	10	$1 \leqslant n \leqslant 10$	$1 \leqslant a \leqslant 1\,000$	$l_i = r_i$		Потестовые
3	20	$1 \leqslant n \leqslant 10$	$1 \leqslant a \leqslant 1\,000$		1, 2	Потестовые
4	20	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 50\,000$		1 – 3	Потестовые
5	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 100\,000$		1 – 4	Баллы
6	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 200\,000$		1 – 5	Баллы
7	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 300\,000$		1 – 6	Баллы
8	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 400\,000$		1 – 7	Баллы
9	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 500\,000$		1 – 8	Баллы
10	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 800\,000$		1 – 9	Баллы
11	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 1\,100\,000$		1 – 10	Баллы
12	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 1\,400\,000$		1 – 11	Баллы
13	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 1\,700\,000$		1 – 12	Баллы
14	4	$1 \leqslant n \leqslant 100$	$1 \leqslant a \leqslant 2\,000\,000$		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 $a_1, a_2, \ldots, a_n$, имеющих рейтинги $x_1, x_2, \ldots, x_n$ соответственно. Программа обучения второго университета состоит из последовательности перечисленных в хронологическом порядке m различных дисциплин $b_1, b_2, \ldots, b_m$, имеющих рейтинги $y_1, y_2, \ldots, y_m$ 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 ($1 \leqslant l_a \leqslant r_a \leqslant n$), 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 ($1 \leqslant l_b \leqslant r_b \leqslant m$), 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 ($1 \leqslant n, m \leqslant 500\,000$).

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

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 ($1 \leqslant x_i \leqslant 10^9$).

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

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 ($1 \leqslant y_i \leqslant 10^9$).

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	$1 \leqslant n, m \leqslant 50$	–	Потестовые
2	10	$1 \leqslant n, m \leqslant 100$	1	Потестовые
3	10	$1 \leqslant n, m \leqslant 300$	1, 2	Потестовые
4	10	$1 \leqslant n, m \leqslant 500$	1 – 3	Потестовые
5	10	$1 \leqslant n, m \leqslant 2000$	1 – 4	Потестовые
6	5	$1 \leqslant n, m \leqslant 5000$	1 – 5	Первая ошибка
7	5	$1 \leqslant n, m \leqslant 10\,000$	1 – 6	Первая ошибка
8	10	$1 \leqslant n, m \leqslant 30\,000$	1 – 7	Первая ошибка
9	10	$1 \leqslant n, m \leqslant 100\,000$	1 – 8	Первая ошибка
10	10	$1 \leqslant n, m \leqslant 250\,000$	1 – 9	Первая ошибка
11	10	$1 \leqslant n, m \leqslant 500\,000$	1 – 10	Первая ошибка