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 сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте.

Ограничение на размер исходного кода программы-решения

Во всех задачах размер файла с исходным кодом решения не должен превышать 256 КБ.

Процесс тестирования

Во всех задачах баллы за подзадачу начисляются только при прохождении всех тестов подзадачи. В таблице системы оценивания для каждой подзадачи указаны номера *необходимых подзадач*. Тесты для данной подзадачи запускаются только, если все необходимые подзадачи пройдены.

Ограничения

Задача	Ограничение по времени	Ограничение по памяти
1. Оборона крепости	2 секунды	512 МБ
2. Экспериментальная робототехника	2 секунды	512 МБ
3. Ловить или не ловить	1 секунда	512 МБ
4. Обитаемые горы	2 секунды	512 МБ

Замечания

В задачах с большим размером входных данных жюри рекомендует участникам, использующим языки группы `C/C++`, отправлять решения на проверку с использованием компилятора `LINUX GNU C++` и считывать данные, используя `scanf`.

Задача 1. Оборона крепости

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

Стена осаждённой крепости состоит из n участков, пронумерованных от 1 до n. Разведка доложила, что противник планирует отправить на штурм участка стены с номером i отряд из a_i нападающих. Для обороны крепости на участки стены будут направлены в общей сложности s защитников.

Участки стены различаются качеством укрепления, что приводит к различной эффективности обороны: на участке стены с номером i каждый защитник способен отразить атаку k_i нападающих.

Пусть на участок с номером i отправлено x_i защитников. Тогда если количество нападающих не превышает величину $x_i \cdot k_i$, то на этом участке ни один из нападающих не прорвётся в крепость. Иначе в крепость прорвутся $(a_i - x_i \cdot k_i)$ нападающих.

Требуется написать программу, распределяющую защитников по участкам стены так, чтобы их общее количество было равно s и в крепость прорвалось наименьшее количество нападающих.

Формат входных данных

Первая строка входных данных содержит целые числа n — количество участков стены и s — количество защитников крепости ($1 \leqslant n \leqslant 100\,000$; $1 \leqslant s \leqslant 10^9$).

Следующие n строк содержат по два целых числа a_i, k_i — общее количество нападающих на i-й участок стены и количество нападающих, атаку которых может отразить один защитник этого участка ($1 \leqslant a_i, k_i \leqslant 10^9$).

Формат выходных данных

Выходные данные должны содержать единственное целое число — минимальное количество нападающих, которые прорвутся в крепость.

Таблица системы оценивания

Номер подзадачи	Баллы	Ограничения				Необх. подзадачи
		n	s	a	k	
1	17	$1 \leqslant n \leqslant 100$	$1 \leqslant s \leqslant 10\,000$	$1 \leqslant a_i \leqslant 100$	$k_i = 1$	
2	21	$1 \leqslant n \leqslant 100$	$1 \leqslant s \leqslant 10\,000$	$1 \leqslant a_i \leqslant 100$	$1 \leqslant k_i \leqslant 2$	1
3	23	$1 \leqslant n \leqslant 100$	$1 \leqslant s \leqslant 10\,000$	$1 \leqslant a_i \leqslant 100$	$1 \leqslant k_i \leqslant 100$	1, 2
4	39	$1 \leqslant n \leqslant 100\,000$	$1 \leqslant s \leqslant 10^9$	$1 \leqslant a_i \leqslant 10^9$	$1 \leqslant k_i \leqslant 10^9$	1 – 3

Примеры

castle.in	castle.out
1 10 8 1	0
3 3 4 2 1 1 10 8	3

Пояснения к примерам

В первом тесте ни один из нападающих не прорвется в крепость, если поставить всех 10 защитников на единственный участок, так как они смогут отбить всех нападающих. Во втором примере можно, например, направить двух защитников на первый участок и одного — на третий.

Задача 2. Экспериментальная робототехника

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

Университеты Татарстана являются ведущими центрами по образовательной робототехнике. Для популяризации этого направления школьникам было предложено провести эксперимент по выживанию роботов в ограниченном пространстве.

Эксперимент проходит на прямоугольном поле размером $n \times m$ клеток. В начале эксперимента в заданных клетках размещается по одному неактивированному роботу. По команде «Старт» запускается таймер, который подаёт сигнал в начале каждой секунды. После каждого сигнала таймера, но не позднее чем через $T_{max} = 10^9$ секунд после команды «Старт», разрешается активировать некоторых роботов.

Каждая клетка поля закрашена в один из четырёх цветов, распознаваемых сенсором робота. Цвет обозначает направление движения из данной клетки в соседнюю: на север, юг, восток или запад. В момент сигнала таймера каждый активированный робот перемещается в соседнюю клетку в направлении, соответствующем цвету клетки, в которой он находится. Все активированные роботы перемещаются одновременно. Цвета клеток поля выбраны так, что при перемещении никакой робот не может выйти за пределы поля.

Во избежание повреждений запрещается активировать роботов таким образом, что это приведёт к их столкновению. Столкновением называется ситуация, когда два или более активированных роботов оказываются в одной клетке поля. Если происходит столкновение, то эксперимент считается неуспешным. При этом перемещение роботов из соседних клеток навстречу друг другу, в результате которого они меняются местами, к столкновению не приводит.

Эксперимент считается завершённым успешно, если все активированные роботы могут продолжать движение без столкновений по полю сколь угодно долго. Результатом эксперимента является количество активированных роботов.

Требуется написать программу, которая поможет школьникам по описанию поля и клеток, где исходно располагаются неактивированные роботы, определить максимально возможный результат эксперимента и, если потребуется, каких именно роботов и в какие моменты времени следует активировать для достижения такого результата.

Формат входных данных

В первой строке входных данных записаны целые числа n, m и g, где n и m — размеры поля с севера на юг и с запада на восток соответственно ($1 \leqslant n, m \leqslant 1000$), а g — признак, равный 1 или 0, обозначающий необходимость определения клеток и моментов времени для активации роботов.

Последующие n строк по m символов в каждой описывают цвета клеток поля и разрешение на активацию робота в них. Цвет поля задаётся английской буквой, соответствующей направлению движения: «`N`» или «`n`» — север, «`S`» или «`s`» — юг, «`E`» или «`e`» — восток и «`W`» или «`w`» — запад.

Клетки, в которых исходно размещены неактивированные роботы, обозначены заглавными буквами, а клетки, в которых исходно робота нет — строчными. Гарантируется, что на поле находится хотя бы один неактивированный робот.

Формат выходных данных

В первой строке выходных данных выведите одно число k — максимально возможный результат эксперимента. Для входных данных, в которых значение g равно 1, в каждой из последующих k строк выведите три целых числа r, c и t: номера строки и столбца, описывающие клетку для активации робота, и момент времени для его активации соответственно ($1 \leqslant r \leqslant n$; $1 \leqslant c \leqslant m$; $1 \leqslant t \leqslant 10^9$). Строки нумеруются от 1 до n сверху вниз (с севера на юг), а столбцы — от 1 до m слева направо (с запада на восток).

Если стратегий активации роботов, приводящих к максимальному результату несколько, то выведите любую из них.

Таблица системы оценивания

Номер подзадачи	Баллы	Ограничения			Необх. подзадачи
		n, m	g	Дополнительно	
1	11	$1 \leqslant n, m \leqslant 10$	$g = 0$	в каждой клетке исходно находится робот	
2	13	$1 \leqslant n, m \leqslant 100$	$g = 0$	в каждой клетке исходно находится робот	1
3	13	$1 \leqslant n, m \leqslant 100$	$g = 0$	—	1, 2
4	23	$1 \leqslant n, m \leqslant 100$	$g = 1$	—	1–3
5	17	$1 \leqslant n, m \leqslant 1000$	$g = 0$	—	1–3
6	23	$1 \leqslant n, m \leqslant 1000$	$g = 1$	—	1–5

Примеры

robots.in	robots.out
3 4 0 SSSS EESW ENWW	4
3 4 1 SSSS EeSW ENwW	4 2 3 2 3 2 1 3 4 1 2 1 2
4 4 1 essS Eess Snww EeWN	5 1 4 9 2 1 4 4 3 3 4 1 2 4 4 7

Пояснение к примеру

В первом примере можно активировать любых четырёх роботов, выбрав для этого подходящие моменты времени. Например, роботы, находящиеся в клетках (1, 1) и (3, 1) не могут быть активированы одновременно. Указывать, какие именно роботы должны быть активированы и в какие моменты в времени, в данном тесте не требуется.

Во втором примере приведённый ответ не является единственным.

В третьем примере активированные в клетках (4, 1) и (4, 3) роботы могут перемещаться между клетками (4, 2) и (4, 3) сколько угодно раз, не сталкиваясь при этом.

Задача 3. Ловить или не ловить

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

Владельцы рыболовецкого судна, ведущего промысел на реке Кама, решили в летнем сезоне оптимизировать свой бизнес.

Они получили сезонное разрешение на лов рыбы в n точках русла реки на расстояниях $x_1, x_2, \ldots, x_n$ километров от устья. При этом в точке с номером i разрешается выловить не более a_i тонн рыбы.

Выловленную рыбу можно продавать на m оптовых базах, расположенных вдоль берега реки в точках на расстояниях $y_1, y_2, \ldots, y_m$ километров от устья. При этом база в точке номер j готова в этом сезоне закупить не более b_j тонн рыбы по цене c_j рублей за тонну.

Расстояния от устья до точек вылова и оптовых баз измеряются вдоль русла реки.

Судно отправляется на лов из устья реки и должно вернуться туда же после окончания сезона. В течение сезона судно может произвольным образом плавать вверх и вниз по реке, останавливаясь для лова или продажи рыбы. Грузоподъёмность судна достаточна для перевозки любого количества выловленной рыбы. При удалении от устья судно движется против течения, расходуя на один километр пути топливо стоимостью p рублей. При перемещении в сторону устья судно движется по течению и поэтому не расходует топлива.

По итогам сезона прибыль за улов будет равна суммарной стоимости проданной рыбы за вычетом суммарной стоимости затраченного топлива.

Требуется написать программу, определяющую максимальную прибыль, которую можно получить за сезон.

Формат входных данных

Первая строка входных данных содержит три целых числа n, m и p — количество точек лова рыбы, количество оптовых баз и стоимость топлива ($1 \leqslant n, m \leqslant 500\,000$; $0 \leqslant p \leqslant 10^9$).

Следующие n строк содержат по два целых числа x_i и a_i — расстояние от устья и максимальный улов для каждой точки лова рыбы ($0 < x_1 < x_2 < \ldots < x_n \leqslant 10^9$; $0 < a_i \leqslant 10^6$).

Следующие m строк содержат по три целых числа y_j, b_j, c_j — расстояние от устья, максимальное закупаемое количество тонн рыбы и цена закупки за тонну для каждой оптовой базы ($0 < y_1 < y_2 < \ldots < y_m \leqslant 10^9$; $0 < b_j, c_j \leqslant 10^6$).

Формат выходных данных

Выходные данные должны содержать единственное целое число — максимальную возможную прибыль.

Таблица системы оценивания

Номер подзадачи	Баллы	Ограничения		Необх. подзадачи
		n, m	Дополнительные	
1	16	$1 \leqslant n, m \leqslant 50\,000$	$p = 0$	
2	9	$1 \leqslant n, m \leqslant 50\,000$	$y_m < x_1$	1
3	16	$1 \leqslant n, m \leqslant 50\,000$	$x_n < y_1$	1
4	11	$1 \leqslant n, m \leqslant 1\,000$	—	
5	9	$1 \leqslant n, m \leqslant 8\,500$	—	4
6	20	$1 \leqslant n, m \leqslant 50\,000$	—	1–5
7	6	$1 \leqslant n, m \leqslant 200\,000$	—	1–6
8	7	$1 \leqslant n, m \leqslant 320\,000$	—	1–7
9	6	$1 \leqslant n, m \leqslant 500\,000$	—	1–8

Примеры

fisher.in	fisher.out
3 2 0 1 5 2 3 4 5 2 2 10 3 6 5	50
2 1 100 6 5 100 4 5 100 2000	9400
3 3 10 1 1 10 100 20 10 2 1000 1 11 50 50 17 50 2	2441

Пояснения к примерам

Во втором примере оптимальными будут следующие действия. Следует доплыть до точки на расстоянии 6 километров от устья, потратив 600 рублей на топливо, и выловить в ней 5 тонн рыбы. После этого следует спуститься на 1 километр по реке к базе на расстоянии 5 километров от устья и продать выловленную рыбу по цене 2000 рублей за тонну. Затем следует вернуться в устье реки. Суммарная прибыль составит 9400 рублей.

Задача 4. Обитаемые горы

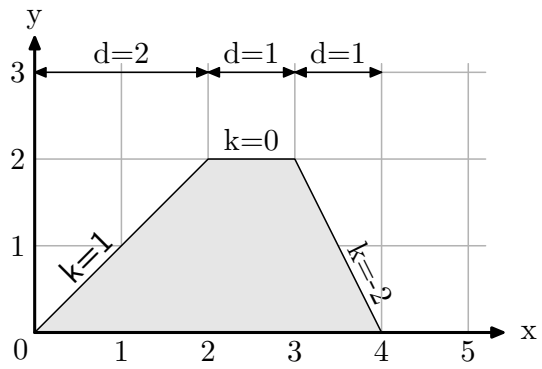
Имя входного файла:	`mountain.in`
Имя выходного файла:	`mountain.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

По мотивам фантастической повести
А. и Б. Стругацких «Обитаемый остров»

Правительство страны Неизвестных Отцов планирует построить в гористом районе на границе с Хонтией башню противобаллистической защиты.

Участок горной цепи в этом районе представлен ломаной, состоящей из n звеньев, последовательно соединяющих $n+1$ вершину. Вершины пронумерованы числами от 0 до n в порядке возрастания координаты x. Звенья пронумерованы числами от 1 до n, при этом i-е звено соединяет вершины с номерами $i-1$ и i.

Вершина с номером 0 находится в точке $(0,0)$. Звено с номером i задано двумя числами d_i — длиной проекции на ось x и k_i — угловым коэффициентом. Таким образом, если вершина с номером $i-1$ имеет координаты (x_{i-1}, y_{i-1}), то координаты i-й вершины можно вычислить как $(x_{i-1}+d_i, y_{i-1}+k_i \cdot d_i)$. Последняя вершина лежит на оси x, то есть $y_n = 0$.



Точка $A\ (x_A, y_A)$ находится в *прямой видимости* из точки $B\ (x_B, y_B)$, если ни одна точка отрезка AB не находится **строго** под ломаной.

Башня представлена вертикальным отрезком ненулевой длины, нижняя точка которого расположена на ломаной. Гражданин страны Неизвестных Отцов чувствует себя в безопасности, если верхняя точка башни находится в его прямой видимости.

Пусть верхняя точка башни имеет координаты (x, y). Два разведчика выбегают из нижней точки башни соответственно на запад (в направлении уменьшения координаты x) и на восток (в направлении увеличения координаты x). Каждый разведчик бежит по поверхности горной цепи до тех пор, пока дальнейшее перемещение не выводит верхнюю точку башни из его прямой видимости или до границы горной цепи.

Правительство подготовило q вариантов расположения башни, каждый из которых характеризуется двумя целыми числами (u_j, v_j) — координатами верхней точки башни. Требуется написать программу, которая для каждого варианта определяет x-координаты двух точек, до которых добегут разведчики.

Формат входных данных

Первая строка входных данных содержит два числа n и q $(1 \leqslant n, q \leqslant 400\,000)$ — количество звеньев ломаной и количество вариантов расположения башни.

Ограничения на величины последующих входных данных зависят от значения константы C, которая может быть равна 10^4 или 10^9 в зависимости от подзадачи (см. таблицу системы оценивания).

Каждая из последующих n строк содержит по два целых числа d_i, k_i $(1 \leqslant d_i \leqslant C;\ -C \leqslant k_i \leqslant C)$ — проекцию на ось x и угловой коэффициент i-го звена ломаной $(0 = x_0 < x_1 < \cdots < x_i < \cdots < x_n \leqslant C;\ y_0 = y_n = 0;\ -C \leqslant y_i \leqslant C)$.

Каждая из последующих q строк содержит по два целых числа u_j, v_j ($0 \leqslant u_j \leqslant C$, $-C \leqslant v_j \leqslant C$) — координаты верхней точки башни в j-м варианте расположения.

Формат выходных данных

Выходные данные должны содержать q строк, в каждой из которых находятся по два целых числа l_j и r_j — x-координаты двух точек, до которых добегут разведчики, направляющиеся на запад и на восток соответственно в j-м варианте расположения башни. Гарантируется, что числа l_j и r_j являются целыми.

Примеры

mountain.in	mountain.out
6 1 3 1 2 -1 1 1 1 -1 1 1 2 -1 5 3	3 8
5 3 1 1 1 -2 2 0 2 1 1 -1 3 0 3 5 3 3	1 6 0 7 0 6
6 4 1 2 2 -2 1 1 1 -2 4 1 1 -1 1 4 3 4 10 4 7 4	0 4 1 9 4 10 1 10
8 4 1 -3 2 0 1 1 2 0 1 -3 1 3 1 2 1 0 2 -2 6 -1 6 4 7 -4	0 6 4 9 0 10 6 9

Таблица системы оценивания

Номер подзадачи	Баллы	Ограничения			Необх. подзадачи
		n, q	C	Дополнительные ограничения	
1	9	$1 \leqslant n, q \leqslant 100$	$C = 10^4$	$k_i = \pm 1$	
2	9	$1 \leqslant n, q \leqslant 100$	$C = 10^4$		1
3	10	$1 \leqslant n, q \leqslant 3000$	$C = 10^9$		1, 2
4	11	$1 \leqslant n, q \leqslant 100\,000$	$C = 10^9$	$k_i = \pm 1$	1
5	11	$1 \leqslant n, q \leqslant 100\,000$	$C = 10^9$	нижние точки всех башен совпадают	
6	12	$1 \leqslant n, q \leqslant 100\,000$	$C = 10^9$	верхние точки всех башен находятся на одной горизонтальной прямой	
7	21	$1 \leqslant n, q \leqslant 100\,000$	$C = 10^9$		1–6
8	17	$1 \leqslant n, q \leqslant 400\,000$	$C = 10^9$		1–7

Замечание

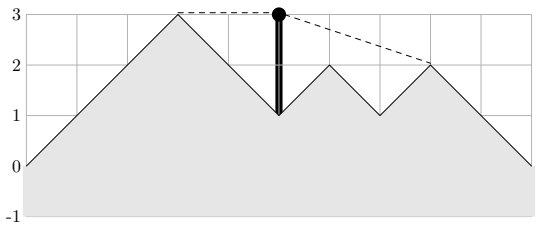


Рис. 1: Первый пример

Обратите внимание, что все точки отрезка между $(6, 2)$ и $(7, 1)$ находятся в прямой видимости из верхней точки башни согласно определению в условии.

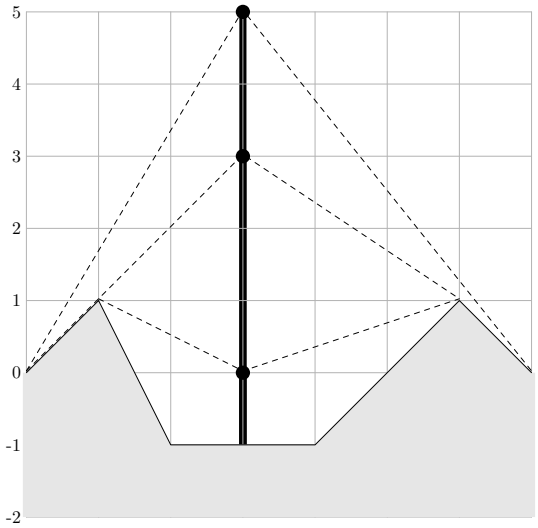


Рис. 2: Второй пример

В данном тесте нижние точки всех вариантов башен совпадают.

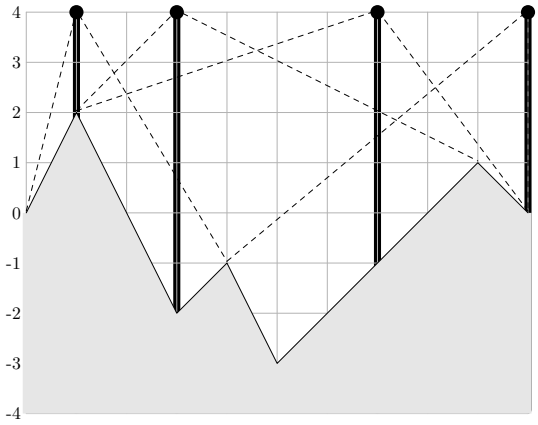


Рис. 3: Третий пример

В данном тесте верхние точки башен всех вариантов находятся на одной горизонтальной прямой. Обратите внимание, что нижняя точка башни может совпадать с одним из концов горной цепи.

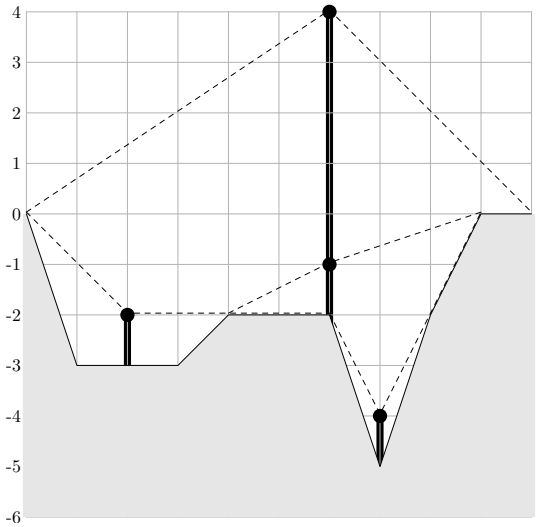


Рис. 4: Четвертый пример

В четвёртом тесте показывается, что в стране Неизвестных Отцов горная цепь может целиком находиться ниже уровня её концов.

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 сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте.

Ограничение на размер исходного кода программы-решения

Во всех задачах размер файла с исходным кодом решения не должен превышать 256 КБ.

Процесс тестирования

Во всех задачах баллы за подзадачу начисляются только при прохождении всех тестов подзадачи. В таблице системы оценивания для каждой подзадачи указаны номера *необходимых подзадач*. Тесты для данной подзадачи запускаются только, если все необходимые подзадачи пройдены.

Ограничения

Задача	Ограничение по времени	Ограничение по памяти
5. Управление видеонаблюдением	1 секунда	512 МБ
6. Расшифровка ДНК	2 секунды	512 МБ
7. Курьерская служба	2 секунды	512 МБ
8. Тренажёр «10_2-пальцевый набор»	3 секунды	512 МБ

Замечания

В задачах с большим размером входных данных жюри рекомендует участникам, использующим языки группы `C/C++`, отправлять решения на проверку с использованием компилятора `LINUX GNU C++` и считывать данные, используя `scanf`.

Обратите внимание, что на этом туре доступна возможность отправки решений на Java и Python, а также решений интерактивной задачи под операционной системой Ubuntu Linux.

Задача 5. Управление видеонаблюдением

Имя входного файла:	`video.in`
Имя выходного файла:	`video.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Охранное агентство получило заказ на обеспечение видеонаблюдения двух зданий. В каждом из зданий установлено множество видеокамер.

На одной из стен зала видеонаблюдения расположена панель, представляющая собой прямоугольник, состоящий из n горизонтальных рядов по m мониторов в каждом. На каждый монитор выводится изображение с камеры, находящейся в одном из зданий. Зал видеонаблюдения оборудован инновационным пультом управления с четырьмя кнопками: «влево», «вправо», «вверх» и «вниз».

Кнопка «влево» перемещает изображение с каждого монитора на монитор, находящийся слева от него. При этом изображение из самого левого монитора в каждом ряду перемещается на самый правый монитор этого ряда.

Аналогичным образом действуют кнопки «вправо», «вверх» и «вниз». Кнопка «вправо» перемещает изображение с каждого монитора на монитор, находящийся справа от него. Изображения из самого правого монитора в каждом ряду перемещаются на самый левый монитор этого ряда. Кнопка «вверх» перемещает изображение с каждого из мониторов на монитор, находящийся над ним. Изображения из самого верхнего ряда перемещаются на мониторы самого нижнего ряда. Кнопка «вниз» перемещает изображение с каждого из мониторов на монитор, находящийся под ним. Изображения из самого нижнего ряда перемещаются на мониторы самого верхнего ряда.

Назовём блок мониторов размером 2×2 *удобным для наблюдения*, если эти мониторы показывают изображения из одного и того же здания. В результате перемещения изображений по командам с пульта управления количество удобных для наблюдения блоков может изменяться. При этом один и тот же монитор может входить в несколько удобных для наблюдения блоков.

Требуется написать программу, определяющую максимальное количество удобных для наблюдения блоков, которое можно получить, управляя мониторами с пульта.

Формат входных данных

Первая строка входных данных содержит два целых числа: n — количество рядов и m — количество мониторов в каждом ряду ($2 \leqslant n, m \leqslant 1000$). Следующие n строк описывают ряды мониторов в порядке сверху вниз. Каждая из этих строк содержит по m символов, описывающих мониторы в соответствующем ряду в порядке слева направо. Символ «`1`» означает, что на монитор выводится изображение из первого здания, а символ «`2`» — из второго здания.

Формат выходных данных

Выходные данные должны содержать единственное целое число — максимальное количество удобных для наблюдения блоков, которые можно получить, перемещая изображения на мониторах.

Таблица системы оценивания

Номер подзадачи	Баллы	Ограничения	Необх. подзадачи
		n, m	
1	37	$2 \leqslant n, m \leqslant 50$	
2	28	$2 \leqslant n, m \leqslant 300$	1
3	35	$2 \leqslant n, m \leqslant 1000$	1, 2

Примеры

video.in	video.out
2 4 1221 1221	2
3 2 22 22 22	2
3 3 111 121 111	3

Пояснения к примерам

В первом примере с помощью команды «вправо», можно получить слева удобный для наблюдения блок из единиц, а справа — удобный для наблюдения блок из двоек.

Во втором примере изначально на мониторе присутствуют два удобных для наблюдения блока.

В третьем примере, например, командами «вправо» и «вниз» можно добиться наличия трёх удобных для наблюдения блоков из единиц.

Задача 6. Расшифровка ДНК

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

Это интерактивная задача.

При проведении раскопок на территории Республики Татарстан были обнаружены останки неизвестного древнего животного, обитавшего в окрестностях современной Казани миллионы лет назад. Как и у всех живых организмов, молекула его ДНК представляет собой последовательность из n нуклеотидов, однако число встречающихся в ней различных нуклеотидов может отличаться от современных организмов.

Для изучения находки был создан специальный прибор, который может просканировать последовательный участок нуклеотидов в ДНК и вычислить, сколько различных видов нуклеотидов содержится на нём. К сожалению, молекула ДНК может выдержать не более q операций сканирования, после чего разрушается.

Исследователи хотят с помощью этого прибора найти k — количество различных нуклеотидов в ДНК, и определить позиции, в которых в ДНК находятся одинаковые нуклеотиды. Ученые хотят закодировать последовательность нуклеотидов в молекуле последовательностью целых положительных чисел $a_1, a_2, \ldots, a_n$ $(1 \leqslant a_i \leqslant k)$, таких что одинаковые числа кодируют одинаковые нуклеотиды, а различные числа — различные нуклеотиды.

Требуется написать программу, которая, взаимодействуя с программой жюри, определит количество различных нуклеотидов в последовательности, а также числовую последовательность, кодирующую последовательность нуклеотидов в молекуле ДНК.

Формат взаимодействия с тестирующей системой

При запуске решения на вход подается целое число n — длина молекулы ДНК $(1 \leqslant n \leqslant 3\,000)$.

Для каждого теста зафиксированы числа k — количество различных нуклеотидов $(1 \leqslant k \leqslant n)$ и q — максимальное количество запросов. Гарантируется, что q запросов достаточно, чтобы решить задачу. Эти числа не сообщаются программе участника, но ограничения на эти числа в различных подзадачах приведены в таблице системы оценивания. Если программа участника делает более q запросов программе жюри, на этом тесте она получает в качестве результата тестирования «Неверный ответ».

Чтобы сделать запрос, следует вывести строку «`?` i j», где i и j — целые положительные числа, номера первого и последнего нуклеотида непрерывного участка молекулы ДНК, для которого требуется узнать число различных нуклеотидов в нём $(1 \leqslant i \leqslant j \leqslant n)$.

В ответ на каждый запрос программа получает целое число p — количество различных нуклеотидов в фрагменте ДНК, указанном в запросе.

Если программа определила ответ на задачу, то она должна вывести три строки. Первая строка должна содержать слово «`Ready!`». Вторая строка должна содержать целое число k — количество различных нуклеотидов в молекуле. Третья строка должна содержать последовательность n целых чисел, разделенных пробелами: $a_1, a_2, \ldots, a_n$ — коды нуклеотидов $(1 \leqslant a_i \leqslant k)$. Если подходящих последовательностей несколько, то допускается вывести любую из них.

После этого программа должна завершиться.

Примеры

стандартный ввод	стандартный вывод
2 2	? 1 2 Ready! 2 1 2
3 1 2	? 1 2 ? 1 3 Ready! 2 1 1 2

Пояснения к примерам

В первом примере $n = 2$, за один запрос можно определить, равны ли первый и второй нуклеотид друг другу.

В втором примере $n = 3$, результат первого запроса показывает, что первые два нуклеотида одинаковые, результат второго запроса позволяет сделать вывод о том, что третий нуклеотид от них отличается. В этом случае допустимы два варианта ответа: `1 1 2` и `2 2 1`.

В точности соблюдайте формат выходных данных. После каждого вывода обязательно выводите один перевод строки и сбрасывайте буфер вывода — для этого используйте `flush(output)` на языке Паскаль или Delphi, `fflush(stdout)` или `cout.flush()` в C/C++, `sys.stdout.flush()` на языке Python, `System.out.flush()` на языке Java.

Таблица системы оценивания

Номер подзадачи	Баллы	Ограничения			Необх. подзадачи
		n	k	q	
1	20	$1 \leqslant n \leqslant 300$	$1 \leqslant k \leqslant 2$	$q = 72\,000$	
2	25	$1 \leqslant n \leqslant 300$	$1 \leqslant k \leqslant n$	$q = 72\,000$	1
3	25	$1 \leqslant n \leqslant 3\,000$	$1 \leqslant k \leqslant 10$	$q = 72\,000$	1
4	15	$1 \leqslant n \leqslant 3\,000$	$1 \leqslant k \leqslant n$	$q = 72\,000$	1–3
5	15	$1 \leqslant n \leqslant 3\,000$	$1 \leqslant k \leqslant n$	$q = 36\,000$	1–4

Задача 7. Курьерская служба

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

В корпорации работают n сотрудников, один из которых является директором. У каждого сотрудника, кроме директора, имеется ровно один непосредственный руководитель.

Каждый сотрудник выполняет свою строго определённую работу. Если сотруднику a требуется выполнить работу, которой занимается сотрудник b, то он должен отправить сотруднику b заявку. Согласно корпоративной политике напрямую заявки могут передаваться только либо от сотрудника его непосредственному руководителю, либо наоборот — от руководителя к непосредственному подчинённому. Заявка передается от сотрудника к сотруднику до тех пор, пока не достигнет сотрудника b.

Чтобы разгрузить сотрудников, корпорация наняла k курьеров. Курьер с номером i закреплён за парой сотрудников a_i и b_i и занимается доставкой заявок между ними. Если одному сотруднику из этой пары требуется передать заявку другому, то он отдает заявку курьеру, который последовательно передает заявку между сотрудниками в соответствии с корпоративной политикой, пока заявка не пройдет всех необходимых промежуточных сотрудников и окажется доставленной адресату. В процессе доставки одной заявки курьер не посещает одного и того же сотрудника более одного раза.

С целью оптимизации расходов решено найти пару курьеров, пути следования которых перекрываются наиболее сильно, и отказаться от услуг одного из них, поручив его работу второму. Назовём *степенью дублирования* для пары курьеров количество переходов между сотрудником и его непосредственным начальником в любом направлении, входящих в путь как первого, так и второго курьера.

Требуется написать программу, которая определит двух курьеров с наибольшей степенью дублирования.

Формат входных данных

Первая строка входных данных содержит два целых числа: n — количество сотрудников и k — количество курьеров ($2 \leqslant n, k \leqslant 2 \cdot 10^5$). Сотрудники пронумерованы от 1 до n, директор имеет номер 1.

Вторая строка содержит $(n-1)$ целых чисел: $p_2, p_3, \ldots, p_n$ — номера непосредственных руководителей каждого сотрудника, кроме директора ($1 \leqslant p_i < i$).

Следующие k строк содержат по два целых числа: a_i и b_i — номера сотрудников в паре, за которой закреплён курьер с номером i ($1 \leqslant a_i, b_i \leqslant n$; $a_i \neq b_i$). За одной и той же парой сотрудников может быть закреплено несколько курьеров.

Формат выходных данных

Первая строка выходных данных должна содержать единственное целое число — наибольшую степень дублирования двух курьеров.

Вторая строка выходных данных должна содержать два различных целых числа от 1 до k — номера курьеров, входящих в пару с наибольшей степенью дублирования. Если таких пар несколько, можно вывести любую из них.

Замечание

Будем говорить, что сотрудник a *подчиняется приказам* сотрудника b, если b является непосредственным начальником a, или же непосредственный начальник a подчиняется приказам сотрудника b. Естественно называть *важным* курьера, передающего заявки от сотрудника к кому-либо, кто подчиняется его приказам (или в обратном направлении).

Таблица системы оценивания

Номер подзадачи	Баллы	Ограничения			Необх. подзадачи
		n	k	Дополнительно	
1	29	$2 \leqslant n \leqslant 100$	$2 \leqslant k \leqslant 100$	—	
2	12	$2 \leqslant n \leqslant 4000$	$2 \leqslant k \leqslant 1000$	—	1
3	7	$2 \leqslant n \leqslant 10^5$	$2 \leqslant k \leqslant 1000$	—	1-2
4	8	$2 \leqslant n \leqslant 10^5$	$2 \leqslant k \leqslant 5000$	—	1-3
5	10	$2 \leqslant n \leqslant 10^5$	$2 \leqslant k \leqslant 50\,000$	Число переходов между любым сотрудником и директором не превышает 20	
6	12	$2 \leqslant n \leqslant 10^5$	$2 \leqslant k \leqslant 50\,000$	Каждый курьер является важным	
7	12	$2 \leqslant n \leqslant 10^5$	$2 \leqslant k \leqslant 50\,000$	—	1-6
8	10	$2 \leqslant n \leqslant 2 \cdot 10^5$	$2 \leqslant k \leqslant 2 \cdot 10^5$	—	1-7

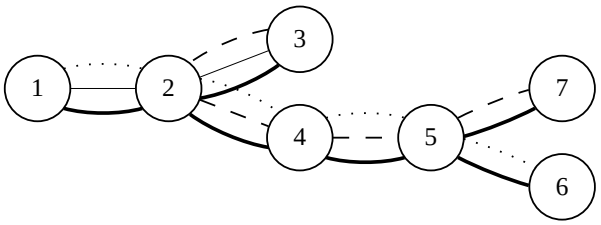
Примеры

twopaths.in	twopaths.out
4 2 1 2 2 1 3 1 4	1 2 1
4 2 1 2 3 1 2 3 4	0 1 2
7 3 1 2 2 4 5 5 1 3 3 7 6 1	2 2 3
4 3 1 2 3 1 4 4 1 1 4	3 2 1

Пояснения к примерам

В первом примере имеется два курьера. Первый курьер доставляет завки от сотрудника 1 сотруднику 3 или обратно. Так, доставляя завку от сотрудника 1 к сотруднику 3, он сначала передает заявку от сотрудника 1 сотруднику 2, а затем от сотрудника 2 сотруднику 3. Второй курьер доставляет завки от сотрудника 1 сотруднику 4 или обратно. Например, доставляя заявку от сотрудника 4 сотруднику 1, он сначала передаёт её от сотрудника 4 сотруднику 2, а затем от сотрудника 2 сотруднику 1. Степень дублирования этой пары курьеров равна 1, так как для них общим является переход между сотрудником 1 (директором) и сотрудником 2.

Во втором примере также имеется два курьера, но их пути не пересекаются.



Третий пример изображён на рисунке выше. Жирными линиями обозначены переходы между сотрудниками. Тонкими линиями обозначен путь первого курьера между сотрудниками 1 и 3. Длинным пунктиром обозначен путь второго курьера между сотрудниками 3 и 7. Коротким пунктиром обозначен путь третьего курьера между сотрудниками 1 и 6.

В пути первого и второго курьера одновременно входит только переход между сотрудниками 2 и 3. В пути первого и третьего курьера одновременно входит только переход между сотрудниками 1 и 2. В пути второго и третьего курьера одновременно входят два перехода: между сотрудниками 2 и 4 и между сотрудниками 4 и 5. Таким образом, пара из курьеров с номерами 2 и 3 имеет наибольшую степень дублирования, равную 2.

В четвертом примере все курьеры переносят заявки между директором и сотрудником номер 4. Таким образом, все пары курьеров имеют степень дублирования равную 3. Разрешается вывести любую пару.

Задача 8. Тренажёр «10_2-пальцевый набор»

Имя входного файла:	`typing.in`
Имя выходного файла:	`typing.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Современный робот-программист должен владеть слепым 10_2-пальцевым методом набора бинарных строк, состоящих из символов «`0`» и «`1`». В инновационном тренажёре, созданном специально для уже освоивших 10_2-пальцевый набор роботов, предлагается следующее упражнение. В верхней части экрана выводится строка, состоящая из нулей и единиц. Ниже выводятся пары (c_i, w_i), каждая из которых состоит из бинарного слова w_i и его *стоимости* c_i — количества штрафных баллов, начисляемых за каждое использование слова w_i при наборе строки.

Робот должен набрать заданную строку в виде последовательности записанных подряд префиксов или суффиксов предложенных ему бинарных слов. Одно и то же слово можно использовать произвольное количество раз, но за каждое использование префикса или суффикса начисляются штрафные баллы, равные стоимости этого слова.

Префиксом слова называется последовательность подряд идущих символов этого слова, начинающаяся с первого символа слова, а *суффиксом* — последовательность подряд идущих символов этого слова, заканчивающаяся последним символом слова. Слово целиком является как своим префиксом, так и своим суффиксом.

Требуется написать программу, которая вычисляет минимально возможное суммарное количество штрафных баллов, начисляемых роботу за набор заданной строки с использованием префиксов и суффиксов предложенных бинарных слов, или определяет, что строку набрать невозможно.

Формат входных данных

Первая строка входных данных содержит три целых числа m, n и L — длину заданной строки, количество слов, префиксы и суффиксы которых можно использовать при её наборе, и суммарную длину этих слов ($1 \leqslant m \leqslant 300\,000$; $1 \leqslant n \leqslant 300\,000$; $1 \leqslant L \leqslant 300\,000$).

Во второй строке находится заданная строка, состоящая из m символов «`0`» или «`1`». Следующие n строк описывают предлагаемые для использования бинарные слова. Сначала указывается целая стоимость слова в штрафных баллах c_i ($1 \leqslant c_i \leqslant 10^9$). Затем в той же строке через пробел следует непустое слово, состоящее из символов «`0`» или «`1`» Длина каждого слова не превосходит значения l_{max}, дополнительные ограничения на которое накладываются в некоторых подзадачах.

Формат выходных данных

Выходные данные должны содержать одно целое число — минимальное количество штрафных баллов, которое потребуется, чтобы набрать заданную строку, или число -1, если требуемым образом набрать её невозможно.

Примеры

typing.in	typing.out
9 2 8 000110100 1 100 1 11001	4
9 3 10 010110101 3 0101 10 011 2 100	8
3 1 3 100 1 101	-1

Пояснения к примерам

В первом примере можно сначала набрать суффикс первого слова длины два, затем его же суффикс длины один, далее префикс второго слова длины три после чего первое слово целиком.

Таблица системы оценивания

В таблице системы оценивания этой задачи указаны только дополнительные ограничения, накладываемые на различные параметры входных данных. Значение l_{max} задает максимальную длину каждого из предложенных роботу бинарных слов, префиксы и суффиксы которых он может использовать.

Номер подзадачи	Баллы	Ограничения					Необх. подзадачи
		m	n	L	c_i	l_{max}	
1	20	$m \leqslant 50$	$n \leqslant 50$	$L \leqslant 500$	$c_i \leqslant 1000$	$l_{max} \leqslant 50$	
2	10	$m \leqslant 5000$	—	$L \leqslant 5000$	—	$l_{max} \leqslant 1000$	1
3	8	$m \leqslant 10\,000$	—	$L \leqslant 50\,000$	—	$l_{max} \leqslant 1000$	1, 2
4	8	$m \leqslant 50\,000$	—	$L \leqslant 50\,000$	—	$l_{max} \leqslant 2000$	1–3
5	10	$m \leqslant 50\,000$	$n \leqslant 20$	$L \leqslant 50\,000$	—	—	
6	5	$m \leqslant 50\,000$	$n \leqslant 200$	$L \leqslant 50\,000$	—	—	5
7	9	$m \leqslant 50\,000$	—	$L \leqslant 50\,000$	$c_i = 1$	—	
8	5	$m \leqslant 50\,000$	—	$L \leqslant 50\,000$	$c_i \leqslant 10$	—	7
9	5	$m \leqslant 50\,000$	—	$L \leqslant 50\,000$	$c_i \leqslant 100$	—	7, 8
10	5	$m \leqslant 50\,000$	—	$L \leqslant 50\,000$	—	—	1–9
11	5	$m \leqslant 100\,000$	—	$L \leqslant 100\,000$	—	—	1–10
12	5	$m \leqslant 200\,000$	—	$L \leqslant 200\,000$	—	—	1–11
13	5	$m \leqslant 300\,000$	—	$L \leqslant 300\,000$	—	—	1–12