Задача A. Нужно больше тренироваться

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Мальчик Антон учится в 6 классе и активно готовится к олимпиадам по программированию. На данный момент он уже сдал n задач, причем i-ю задачу он сдал в час t_i.

Существует m часовых поясов, пронумерованных от 0 до $m-1$. В каждом часовом поясе один день состоит из m последовательных часов. При том в k-м часовом поясе d-й день состоит из часов с номерами от $d \cdot m + k$ до $(d+1) \cdot m + k - 1$ (включительно). Обратите внимание, что d может быть отрицательным.

В начале года Антон поставил себе цель сдавать как минимум по одной задаче каждый день. Сейчас он хочет проверить, существует ли такой часовой пояс и два дня l и r, такие что в любой из этих дней он сдал хотя-бы одну задачу, а также любая сданная задача была сдана именно в один из дней от l до r в этом часовом поясе.

Помогите Антону найти такой часовой пояс, или определите, что его не существует. Если подходящих часовых поясов несколько — найдите минимальный из них.

Формат входных данных

Первая строка содержит два целых числа n и m ($1 \leqslant n \leqslant 200\,000$, $1 \leqslant m \leqslant 10^9$) — количество сданных задач и количество часов в каждом дне соответственно.

Вторая строка содержит n целых чисел $t_1,\ t_2,\ t_3,\ \ldots,\ t_n$ ($0 \leqslant t_i \leqslant 10^9$, $t_i \leqslant t_{i+1}$) — время сдачи каждой задачи в неубывающем порядке.

Формат выходных данных

Выведите одно целое число — минимальный номер подходящего часового пояса, или -1, если его не существует.

Примеры

стандартный ввод	стандартный вывод
3 3 4 5 10	2
4 5 2 4 14 17	-1
6 3 1 2 6 10 11 12	2

Замечание

В первом примере день идёт 3 часа и Антон сдал 3 задачи: в часы 4, 5 и 10 соответственно.

- В часовом поясе 0 посылки Антона попадают в дни 1, 1 и 3 соответственно. Так как в день 2 Антон не сделал ни одной посылки, этот пояс не подходит.
- В часовом поясе 1 посылки Антона попадают в дни 1, 1 и 3 соответственно. Так как в день 2 Антон не сделал ни одной посылки, этот пояс не подходит.
- В часовом поясе 2 посылки Антона попадают в дни 0, 1 и 2 соответственно. Дни образуют непрерывный отрезок, а значит этот пояс подходит. Так как он минимальный среди подходящих, 2 является ответом на задачу.

Во втором примере ни один из часовых поясов не подходит.
В третьем примере:

- В часовом поясе 0 посылки Антона попадают в дни 0, 0, 2, 3, 3 и 4 соответственно. Так как в день 1 Антон не сделал ни одной посылки, этот пояс не подходит.

- В часовом поясе 1 посылки Антона попадают в дни 0, 0, 1, 3, 3 и 3 соответственно. Так как в день 2 Антон не сделал ни одной посылки, этот пояс не подходит.
- В часовом поясе 2 посылки Антона попадают в дни -1, 0, 1, 2, 3 и 3 соответственно. Дни образуют непрерывный отрезок от -1 до 3, поэтому этот пояс и является ответом.

Система оценки

Тесты к этой задаче состоят из 5 групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп.

Группа	Баллы	Доп. ограничения			Необх. группы	Комментарий
		n	m	t_i		
0	0	–	–	–	–	Тесты из условия.
1	18	$n \leqslant 500$	$m \leqslant 100$	–	0	
2	19	–	$m \leqslant 100$	–	0, 1	
3	16	$n \leqslant 500$	–	$t_i \leqslant 500$	0	
4	21	$n \leqslant 5000$	–	$t_i \leqslant 500$	0, 3	
5	26	–	–	–	0 – 4	

Задача B. Опять запросы

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	512 мегабайт

Дан массив $a_0,\ a_1,\ \ldots,\ a_{2^n-1}$ размера 2^n. Обратите внимание, что элементы нумеруются с нуля. Поступает q запросов двух видов:

- $1\ l\ r\ k\ v$. В этом случае нужно элементы $a_{l\oplus k}$, $a_{(l+1)\oplus k}$, …, $a_{r\oplus k}$ заменить на v.
- $2\ l\ r\ k$. В этом случае нужно посчитать сумму $a_{l\oplus k} + a_{(l+1)\oplus k} + \cdots + a_{r\oplus k}$.

Напомним, что символом «$\oplus$» обозначается операция побитового исключающего ИЛИ (XOR).

Формат входных данных

Первая строка содержит единственное целое число n ($0 \leqslant n \leqslant 20$).

Вторая строка содержит 2^n целых чисел $a_0,\ a_1,\ \ldots,\ a_{2^n-1}$ ($0 \leqslant a_i \leqslant 10^7$) — элементы массива.

Третья строка содержит единственное целое число q ($1 \leqslant q \leqslant 10^6$) — количество запросов.

Следующие q строк содержат описание запросов. В i-й из них находится целое число t_i ($1 \leqslant t_i \leqslant 2$).

- Если $t_i = 1$, то далее строка содержит четыре целых числа l_i, r_i, k_i и v_i ($0 \leqslant l_i \leqslant r_i < 2^n$, $0 \leqslant k_i < 2^n$, $0 \leqslant v_i \leqslant 10^9$). В этом случае нужно каждому из элементов $a_{l_i\oplus k_i}$, $a_{(l_i+1)\oplus k_i}$, …, $a_{r_i\oplus k_i}$ присвоить v_i.
- Если $t_i = 2$, то далее строка содержит три целых числа l_i, r_i и k_i ($0 \leqslant l_i \leqslant r_i < 2^n$, $0 \leqslant k_i < 2^n$). В этом случае нужно посчитать значение выражения $a_{l_i\oplus k_i} + a_{(l_i+1)\oplus k_i} + \ldots + a_{r_i\oplus k_i}$.

Формат выходных данных

Для каждого запроса второго типа выведите искомую сумму.

Пример

стандартный ввод	стандартный вывод
3 1 5 8 4 3 7 8 4 3 2 4 7 1 1 1 5 3 8 2 4 7 1	22 26

Замечание

В первом запросе $a_{4\oplus1} + a_{5\oplus1} + a_{6\oplus1} + a_{7\oplus1} = a_5 + a_4 + a_7 + a_6 = 22$

Во втором запросе нужно заменить элементы $a_{1\oplus3}, a_{2\oplus3}, a_{3\oplus3}, a_{4\oplus3}$ и $a_{5\oplus3}$ на 8. Массив после этих изменений будет равен $8, 8, 8, 4, 3, 7, 8, 8$.

В последнем запросе $a_{4\oplus1} + a_{5\oplus1} + a_{6\oplus1} + a_{7\oplus1} = 26$.

Система оценки

Тесты к этой задаче состоят из 8 групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание, прохождение тестов из условия не требуется для некоторых групп. **Offline-проверка** означает, что результаты тестирования вашего решения на данной группе станут доступны только после окончания соревнования.

Группа	Баллы	Доп. ограничения				Необх. группы	Комментарий
		n	q	a_i	v_i		
0	0	–	–	–	–	–	Тесты из условия.
1	19	$n \leqslant 10$	$q \leqslant 1000$	–	–	0	
2	17	$n \leqslant 17$	$q \leqslant 200\,000$	–	–	–	k_i степень двойки. Если $t_i = 1$, то $l_i = r_i$
3	11	–	–	$a_i \leqslant 1$	–	–	$t_i = 2$
4	13	–	–	$a_i \leqslant 1$	$v_i \leqslant 1$	3	
5	12	$n \leqslant 17$	$q \leqslant 200\,000$	–	–	–	$t_i = 2$
6	9	$n \leqslant 17$	$q \leqslant 200\,000$	–	–	2	Если $t_i = 1$, то $l_i = r_i$
7	10	$n \leqslant 17$	$q \leqslant 200\,000$	–	–	0, 1, 2, 5, 6	
8	9	–	–	–	–	0 – 7	**Offline-проверка.**

Задача C. Задача для разминки ног

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 мегабайт

Однажды мальчик Кирилл долго и упорно пытался решить задачу с говорящим названием «Задача для разминки рук». Получилось ли у него решить задачу мы не знаем, однако, в процессе размышлений Кирилл придумал новую задачу, которую вам и предстоит решить. У Кирилла получилось следующее условие задачи:

В стране X есть n городов, соединённых m дорогами. Множество дорог обладает двумя интересными свойствами. Во-первых, все дороги односторонние. Во-вторых, даже если бы дороги были двусторонними, для любой пары городов существовало бы не более одного простого пути между ними. Нетрудно заметить, что без ориентации граф на данных n городах в качестве вершин и m дорогах в качестве рёбер является лесом (то есть в каждой компоненте связности число ребер на единицу меньше числа вершин).

Функцию $cost(u, v)$ определим следующим образом: если путь из u в v существует, то она равна количеству дорог на этом пути, а если пути нет, то она равна 0.

Красотой карты страны назовём величину $\sum_{u=1}^{n} \sum_{v=1}^{n} cost(u, v)$, то есть сумму величин *cost* для всех пар городов.

Правительство страны X очень любит считать различные статистики. В этот раз было решено посчитать *красоту* текущей карты страны, а так же обработать q изменений карты страны, после каждого изменения подсчитывая её *красоту*.

Изменения бывают 4 типов:

1. Добавить дорогу из города u в город v.
2. Удалить дорогу между городами u и v.
3. Изменить ориентацию дороги между городами u и v на противоположную.
4. Ориентировать все дороги на пути из u в v по направлению из u в v. Гарантируется, что это возможно, то есть существует путь из города u в город v по дорогам, если игнорировать их ориентацию.

Гарантируется, что после каждого изменения множество городов и дорог продолжает образовывать лес.

Обратите внимание на то, что в данной задаче запросы меняют карту страны и зависят от предыдущих.

Формат входных данных

Первая строка содержит одно целое число g ($0 \leqslant g \leqslant 10$) — номер группы тестов.

Вторая строка содержит три целых числа n, m и q ($2 \leqslant n \leqslant 400\,000$, $0 \leqslant m < n$, $1 \leqslant q \leqslant 400\,000$) — количество городов, дорог и изменений соответственно.

Каждая из следующих m строк содержит по два целых числа u и v ($1 \leqslant u, v \leqslant n$) — дороги на изначальной карте ориентированные из u в v.

Каждая из следующих q строк содержит по три целых числа t, u и v ($1 \leqslant t \leqslant 4$, $1 \leqslant u, v \leqslant n$) — тип запроса и пару вершин, задающую его.

Формат выходных данных

В первой строке выведите изначальную *красоту* карты страны.

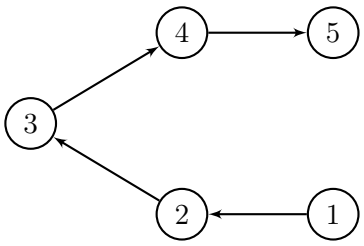
В следующих q строках выведите по одному числу — значения *красоты* карты страны после каждого изменения.

Пример

стандартный ввод	стандартный вывод
0 5 4 4 1 2 2 3 3 4 4 5 3 4 3 2 3 2 1 4 2 4 1 4	20 6 3 4 16

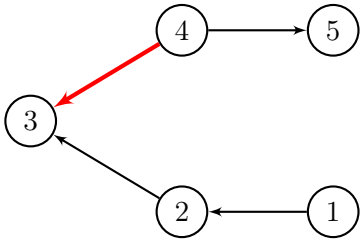
Замечание

В тестовом примере до всех изменений граф выглядит так:

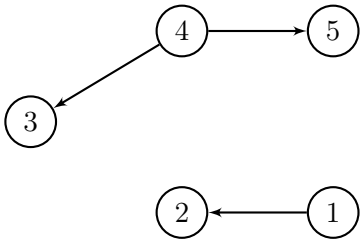


В этом случае, например, $cost(1,2) = cost(2,3) = cost(3,4) = cost(4,5) = 1$, так как между этими парами городов есть прямая дорога. Аналогично $cost(1,3) = cost(2,4) = cost(3,5) = 2$, так как второй город в каждой паре достижим только за две дороги от первого. Так же, например, $cost(2,1) = cost(4,1) = cost(5,3) = 0$, так как второй город в паре не достижим от первого. Суммируя эти величины по всем парам городов, получим 20.

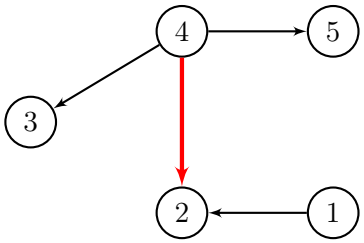
После этого надо изменить направление дороги между городами 3 и 4. Так как до этого она шла от города 3 в город 4, теперь она должна идти от города 4 в город 3. После этого ответ на задачу равен 6, а граф имеет следующий вид:



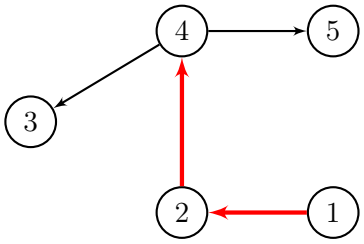
Далее дорога между городами 2 и 3 удаляется. После этого ответ на задачу равен 3, а граф имеет следующий вид:



После этого добавляется дорога из города 4 в город 2. После этого ответ на задачу равен 4, а граф имеет следующий вид:



После этого, все дороги между городами 1 и 4 ориентируются по направлению из города 1 в город 4. После этого ответ на задачу равен 16, а граф имеет следующий вид:



Система оценки

Тесты к этой задаче состоят из десяти групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание, прохождение тестов из условия не требуется для некоторых групп. **Offline-проверка** означает, что результаты тестирования вашего решения на данной группе станут доступны только после окончания соревнования.

Группа	Баллы	Доп. ограничения		Необх. группы	Комментарий
		n, m, q	Типы изменений		
0	0	–	–	–	Тесты из условия.
1	10	$n, m, q \leqslant 100$	–	0	
2	8	$n, m, q \leqslant 5000$	–	0, 1	
3	11	$n, m, q \leqslant 100\,000$	1	–	
4	7	$n, m, q \leqslant 100\,000$	1, 2	3	
5	13	$n, m, q \leqslant 100\,000$	3	–	
6	9	$n, m, q \leqslant 100\,000$	3, 4	5	
7	12	$n, m, q \leqslant 100\,000$	–	–	$\lvert u-v\rvert = 1$ всегда, кроме запросов 4 типа.
8	18	$n, m, q \leqslant 100\,000$	–	0 – 7	
9	6	$n, m, q \leqslant 200\,000$	–	0 – 8	**Offline-проверка.**
10	6	–	–	0 – 9	**Offline-проверка.**

Задача D. Сложная задача

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Мальчик Гена очень любит решать сложные задачи по программированию. Недавно он составил себе подборку из n задач, которые собирается прорешать. Задачи имеют уровни сложности $a_1,\ a_2,\ \ldots,\ a_n$, причём известно, что у любых двух задачек уровни сложности различны.

Гена очень опытный программист, и поэтому может решить сколь угодно сложные задачи. Гена ежедневно решает задачи, у него есть определенная стратегия, по которой он это делает.

Для каждого дня у Гены зафиксирован свой параметр m — минимальный уровень сложности задач, которые он готов решать. Каждый день Гена проходится по всем задачам из подборки начиная с первой до последней, и для каждой из них он делает следующее действие:

1. Если текущая задача уже была сдана, то он её пропускает и переходит к следующей.
2. Если у текущей задачи уровень сложности меньше m, то он её пропускает и переходит к следующей.
3. Если условия предыдущих пунктов не выполнились, и в текущий день он не решал задач, то он решает текущую задачу и переходит к следующей.
4. Если условия предыдущих пунктов не выполнились, и последняя решенная за день задача была легче текущей, то он решает текущую задачу и переходит к следующей.
5. Если условия первых четырех пунктов не выполнились, то Гена пропускает задачу и переходит к следующей.

Проще говоря, Гена каждый день решает задачи в порядке возрастания сложности, но решает только те задачи, которые раньше не решал, и которые по сложности не меньше m.

В первый день минимальная сложность задач, которые он готов решать равна t, а каждый следующий день он будет уменьшать m на значение s. Таким образом в i-й день (в нумерации с 1) Гена готов решать задачи сложности не меньше $t - s \cdot (i - 1)$. Гена будет ежедневно повторять описанный выше алгоритм до тех пор, пока не сдаст все задачи.

Мальчик Леша давно наблюдает за успехами Гены, и хочет узнать секрет его стратегии решения задач. Поэтому у Леши есть q запросов. Для каждого запроса требуется проверить, можно ли так переставить задачи в подборке Гены, чтобы задачу с уровнем сложности d_i он сдал p_i-й по счёту среди всех сданных задач за все дни. Помогите Леше ответить на все его запросы.

Обратите внимание, что запросы **независимы**, то есть для разных запросов могут использоваться различные порядки задач в подборке Гены.

Формат входных данных

Первая строка содержит три целых числа n, t, s ($1 \leqslant n \leqslant 200\,000$, $1 \leqslant t, s \leqslant 10^9$) — количество задач подборке, минимальный уровень сложности в первый день и шаг понижения уровня сложности задач.

Вторая строка содержит n различных целых чисел $a_1,\ a_2,\ \ldots,\ a_n$ ($1 \leqslant a_i \leqslant 10^9$, $a_i < a_{i+1}$) — уровни сложности задач.

Третья строка содержит единственное целое число q ($1 \leqslant q \leqslant 200\,000$) — количество запросов.

Следующие q строк содержат по два целых числа d_i и p_i ($1 \leqslant p_i \leqslant n$) — параметры запроса: можно ли переставить задачи в подборке так, чтобы задача **сложности** d_i была сдана p_i-й по счёту. Гарантируется, что для каждого запроса существует задача в подборке, сложность которой равна d_i.

Формат выходных данных

Для каждого запроса выведите «`Yes`» (без кавычек), если можно переставить задачи в подборке, чтобы условие запроса выполнялось, и «`No`» (без кавычек) в противном случае.

Примеры

стандартный ввод	стандартный вывод
5 10 2 4 5 9 10 12 5 10 2 10 3 10 4 5 5 12 2	Yes Yes No Yes Yes
7 4 2 2 3 5 6 9 10 11 4 5 6 11 7 2 2 10 7	Yes No Yes Yes

Замечание

Пусть в первом примере для второго запроса можно переставить задачи в подборке в таком порядке: 12, 4, 5, 9, 10. Тогда Гена будет прорешивать задачи следующим образом:

1. В первый день минимальная сложность задач $m = 10$. Под это условие подходит первая задача в подборке. Гена её сразу решит. Среди следующих задач есть задача сложности 10, но так как Гена уже сдал за первый день задачу сложности 12, то задачу сложности 10 он пропускает. Таким образом после окончания первого дня Гена сдаст только задачу сложности 12.

2. Во торой день минимальная сложность задач $m = 8$. Во время прохода по задачам из подборке Гена пропустит задачу 12, так как уже сдавал её. Задачи сложности 4 и 5 он пропустит, так как их сложность меньше m. Далее он встретит задачу сложности 9, и так как за второй день он ещё не сдавал задач, то он сдаст её. Следующей задачей идет задача сложности 10, и так её сложность больше сложности последней сданной за день задачи, то он сдаст её. Таким образом после первых двух дней Гена сдаст задачи сложности 12, 9, 10 в соответствующем порядке.

3. В третий день Гена не сдаст ни одной задачи, так как все задачи сложности хотя бы $m = 6 = 10 - 2 \cdot 2$ уже решены.

4. В последний четвёртый день Гена сдаст оставшиеся задачи, итоговая за все дни он сдаст задачи сложности 12, 9, 10, 4, 5 в соответствующем порядке.

При данной перестановке задач в подборке Гена сдаст задачу сложности 10 третьей по счёту среди всех, поэтому ответ на второй запрос «`Yes`».

Система оценки

Тесты к этой задаче состоят из 8 групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание, прохождение тестов из условия не требуется для некоторых групп.

Группа	Баллы	Доп. ограничения				Необх. группы	Комментарий
		n	q	a_i	t		
0	0	–	–	–	–	–	Тесты из условия
1	13	$n \leqslant 8$	–	$a_i \leqslant 10$	$t \leqslant 10$	–	
2	10	$n \leqslant 8$	–	–	–	0, 1	
3	14	$n \leqslant 500$	–	–	$t \leqslant 10$	0, 1	
4	19	$n \leqslant 500$	–	–	–	0 – 3	
5	9	–	$q = 1$	–	–	–	
6	1	–	–	–	$t = 1$	–	
7	9	–	–	–	–	–	$s = 10^9$
8	25	–	–	–	–	0 – 7	

Задача E. Разноцветный граф

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	512 мегабайт

Дан неориентированный граф на n вершинах, содержащий m ребер, пронумерованных от 1 до m. Каждое ребро покрашено в один из k цветов. i-е ребро соединяет вершины v_i и u_i и имеет цвет c_i.

Назовём граф *хорошим*, если в нём можно оставить ровно $n - 1$ ребро так, чтобы граф был связным и среди оставленных ребер было хотя-бы одно ребро каждого цвета.

Дано q изменений цветов ребер графа. Каждое изменение задаётся двумя числами e_i и w_i и означает, что цвет e_i-го ребра становится равным w_i. После каждого изменения графа определите, является ли он *хорошим*.

Формат входных данных

В первой строке находятся три целых числа n, m и k ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 100\,000$, $1 \leqslant k \leqslant 8$) — количество вершин, рёбер и цветов соответственно.

Следующие m строк содержат описание рёбер. i-я из них содержит три числа v_i, u_i и c_i ($1 \leqslant v_i, u_i \leqslant n$, $1 \leqslant c_i \leqslant k$, $v_i \neq u_i$) — концы i-го ребра и его цвет соответственно.

Следующая строка содержит единственное целое число q ($1 \leqslant q \leqslant 100\,000$) — количество изменений.

Следующие q строк содержат описание изменений. i-я из них содержит два целых числа e_i и w_i ($1 \leqslant e_i \leqslant m$, $1 \leqslant w_i \leqslant k$) — номер ребра и его новый цвет.

Гарантируется, что в графе нет петель и кратных ребер.

Формат выходных данных

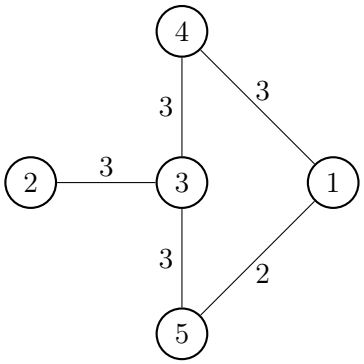
В i-й строке выведите «`Yes`» (без кавычек), если граф после i-го запроса *хороший*, и «`No`» (без кавычек) иначе.

Примеры

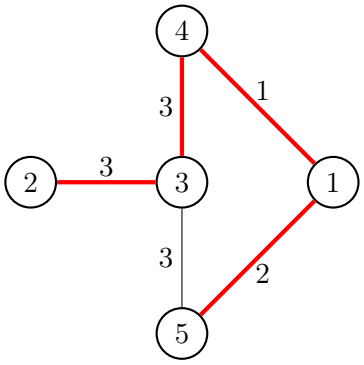
стандартный ввод	стандартный вывод
5 5 3 3 4 1 1 5 2 3 2 3 1 4 3 3 5 3 5 1 3 4 1 5 1 2 1 3 2	No Yes Yes No Yes
2 1 1 1 2 1 1 1 1	Yes

Замечание

В первом примере после первого изменения граф выглядит так:



В этом случае нет ни одного ребра цвета 1, поэтому условие задачи не может быть выполнено. После второго изменения граф выглядит так:



Красным выделены рёбра, которые можно оставить. В данном случае, они подходят, так как среди них есть все цвета 1, 2 и 3, а так же они образуют связный граф.

Система оценки

Тесты к этой задаче состоят из 8 групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание, прохождение тестов из условия не требуется для некоторых групп. **Offline-проверка** означает, что результаты тестирования вашего решения на данной группе станут доступны только после окончания соревнования.

Группа	Баллы	Доп. ограничения	Необх. группы	Комментарий
		k		
0	0	–	–	Тесты из условия.
1	10	$k \leqslant 1$	–	
2	9	$k \leqslant 2$	1	
3	15	$k \leqslant 3$	0 – 2	
4	16	$k \leqslant 4$	0 – 3	
5	14	$k \leqslant 5$	0 – 4	
6	13	$k \leqslant 6$	0 – 5	
7	12	$k \leqslant 7$	0 – 6	
8	11	$k \leqslant 8$	0 – 7	**Offline-проверка.**

Задача F. Турнир

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

Это интерактивная задача.

Загадан турнирный граф на n вершинах, пронумерованных от 1 до n. Турнирный граф — это ориентированный граф, в котором любое ребро соединяет две различные вершины, причём для любой пары различных вершин u и v есть ровно одно ребро, ориентированное либо от u к v, либо от v к u.

Вам известно лишь количество вершин в графе, за один запрос вы можете узнать в какую сторону направлено ребро между любой парой вершин.

Назовём вершину *хорошей*, если из неё исходит не более одного ребра. Ваша задача найти любую *хорошую* вершину, либо определить что таких вершин нет, сделав не более 2000 запросов.

Протокол взаимодействия

Ваша программа будет взаимодействовать с программой жюри с использованием стандартных потоков ввода и вывода. Каждое взаимодействие будет состоять из решения задачи для нескольких наборов входных данных.

Сначала ваша программа должна считать два целых числа g и t ($0 \leqslant g \leqslant 10$, $1 \leqslant t \leqslant 100$) — номер группы тестов и количество наборов входных данных в рамках одного взаимодействия с программой жюри. Затем t раз необходимо выполнить взаимодействие по решению задачи для набора входных данных.

Рассмотрим протокол взаимодействия для одного набора входных данных.

Сначала ваша программа должна считать одно число n ($1 \leqslant n \leqslant 500$) — количество вершин загаданного графа.

После этого вы можете задавать запросы. Для одного запроса выведите «`? u v`» ($1 \leqslant u, v \leqslant n$, $u \neq v$). В ответ на это программа жюри в единственное строке выведет «`forward`», если ребро направлено от u к v и «`backward`», если ребро направлено от v к u. Для каждого набора входных данных вы можете задать не более 2000 запросов.

Чтобы вывести ответ, выведите «`! u`» ($1 \leqslant u \leqslant n$ или $u = -1$), где u — номер *хорошей* вершины, или -1, если такой вершины нет. Если выведенный ответ верный, то программа жюри выведет «`OK`», после этого ваша программа должна сразу перейти к обработке следующего набора входных данных или завершится, если это был последний граф. В ином случае программа жюри выведет «`WRONG`». При считывании этой строки ваша программа должна немедленно завершить свое исполнение.

Обратите внимание, что ограничения на n даны для каждого набора входных данных, дополнительных ограничений на сумму n по всем наборам входных данных нет.

Если вы используете «`cout << ... << endl`» в C++, «`System.out.println`» в Java, «`print`» в Python, «`writeln`» в Pascal, то сброс потока вывода происходит автоматически, дополнительно ничего делать не требуется.

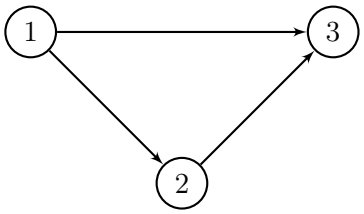
Если вы используете другой способ вывода, рекомендуется делать сброс буфера потока вывода. Обратите внимание, что перевод строки надо выводить в любом случае. Для сброса буфера потока вывода можно использовать «`fflush(stdout)`» в C++, «`flush(output)`» в Pascal, «`System.out.flush()`» в Java, «`sys.stdout.flush()`» в Python.

Пример

стандартный ввод	стандартный вывод
0 2 3 forward backward OK 5 forward forward backward backward forward forward backward forward forward forward OK	 ? 1 2 ? 3 2 ! 3 ? 1 2 ? 1 3 ? 1 4 ? 1 5 ? 2 3 ? 2 4 ? 2 5 ? 3 4 ? 3 5 ? 4 5 ! -1

Замечание

В первом тестовом примере, так как программа жюри неадаптивна, граф загадан заранее и имеет следующий вид:



Поэтому на запрос «? 1 2» программа жюри вывела «forward», так как ребро направлено от 1 к 2, а на запрос «? 3 2» программа жюри вывела «backward», так как ребро направлено от 2 к 3. В этом случае вершины 2 и 3 являются *хорошими*, а вершина 1 нет. Поэтому на ответ «! 3» программа жюри вывела «OK».

Во втором тестовом примере из каждой вершины исходит больше одно ребра, поэтому на ответ «! −1» программа жюри вывела «OK».

Система оценки

Тесты к этой задаче состоят из 10 групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание,

прохождение тестов из условия не требуется для некоторых групп. **Offline-проверка** означает, что результаты тестирования вашего решения на данной группе станут доступны только после окончания соревнования.

В данной задаче программа жюри может действовать двумя способами:

- Неадаптивно. То есть структура графа фиксируется заранее и не подстраивается под запросы.
- Адаптивно. То есть структура графа может меняться во время взаимодействия. При этом гарантируется, что всегда существует хотя бы один граф, который подходит под ответы на все уже заданные запросы. В этом случае, ответ вашего решения будет признан корректным, только если он будет верен для всех подходящих графов.

Группа	Баллы	Доп. ограничения		Необх. группы	Комментарий
		n	Программа жюри		
0	0	–	Неадаптивна	–	Тесты из условия.
1	14	$n \leqslant 63$	Адаптивна	0	
2	8	–	Адаптивна	–	Граф является *особым*[1]
3	11	–	Адаптивна	–	Граф является *особым*[2]
4	12	–	Адаптивна	–	Граф является *особым*[3]
5	9	$n \leqslant 100$	Адаптивна	0, 1	
6	10	$n \leqslant 400$	Неадаптивна	0	
7	9	$n \leqslant 400$	Адаптивна	0, 1, 5, 6	
8	8	$n \leqslant 450$	Неадаптивна	0, 6	
9	10	–	Неадаптивна	0, 6, 8	
10	9	–	Адаптивна	0 – 9	**Offline-проверка.**

[1] Граф называется *особым*[1], если существует перестановка вершин v_1, v_2, …, v_n, что для любых $i < j$ верно, что ребро ориентированно от вершины v_i к вершине v_j.

[2] Граф называется *особым*[2], если для любой вершины с номером $v > 2$, ребро ориентировано от нее к вершине 1 или к вершине 2, или к обеим вершинам 1 и 2.

[3] Граф называется *особым*[3], если $n \geqslant 4$ и существует перестановка вершин v_1, v_2, …, v_n, что ребро (v_i, v_{i+1}) ориентированно от v_i к v_{i+1}, а любое другое ребро (v_i, v_j) $(i + 1 < j)$ ориентировано от v_j к v_i.

Задача G. Космическое происшествие

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

Вы являетесь членом экипажа космического корабля «Гиперион», и недавно вы попали в ожесточённое космическое сражение, из которого чудом выбрались. Однако реактор вашего корабля получил серьёзные повреждения и перегрелся, из-за чего запустилась экстренная система охлаждения.

Реактор корабля состоит из n независимых блоков, каждый из которых имеет свою температуру, которая выражается целым числом градусов. Чтобы он мог работать стабильно, каждый блок должен иметь **строго** отрицательную температуру. Благодаря достижениям науки, система охлаждения за один цикл охлаждает $n - 1$ блок на B градусов, а один оставшийся — на A градусов.

Система охлаждения не пострадала, но из-за повреждений корабельный компьютер не может рассчитать минимальное необходимое количество циклов для полного охлаждения реактора, и только вы можете решить эту непростую задачу.

Формат входных данных

Первая строка содержит три целых числа n, A, B ($1 \leqslant n \leqslant 100\,000$, $1 \leqslant A, B \leqslant 10^9$) — количество блоков в реакторе и параметры A и B.

Следующая строка содержит n целых чисел $t_1,\ t_2,\ \ldots,\ t_n$ ($-10^9 \leqslant t_i \leqslant 10^9$), где t_i — температура i-го блока в реакторе.

Формат выходных данных

Выведите единственное число — минимальное число циклов до полного охлаждения реактора.

Примеры

стандартный ввод	стандартный вывод
5 1 2 1 2 3 4 5	3
3 42 42 -273 -273 -273	0
1 3 2 9	4
4 4 2 5 5 1 0	2

Замечание

В четвертом тестовом примере система охлаждения может на первом цикле охладить первый блок на 4, а остальные на 2, тогда получатся следующие температуры блоков: $\{1, 3, -1, -2\}$, а на следующем цикле охладить второй блок на 4, после чего температуры всех блоков станут отрицательными.

Система оценки

Тесты к этой задаче состоят из 6 групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание, прохождение тестов из условия не требуется для некоторых групп.

Группа	Баллы	Доп. ограничения				Необх. группы	Комментарий
		n	t_i	A	B		
0	0	–	–	–	–	–	Тесты из условия.
1	15	$n \leqslant 5$	$\lvert t_i \rvert \leqslant 5$	$A \leqslant 5$	$B \leqslant 5$	–	
2	16	–	–	$A = 1$	$B = 2$	–	
3	10	–	–	–	–	–	A=B
4	15	n = 2	–	–	–	–	
5	24	$n \leqslant 500$	$\lvert t_i \rvert \leqslant 500$	$A \leqslant 500$	$B \leqslant 500$	0, 1	
6	20	–	–	–	–	0 – 5	

Задача H. Обед

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Дан выпуклый многоугольник из N точек. **Не гарантируется**, что никакие три из них не лежат на одной прямой. В многоугольнике даны M различных *особых* точек. Также дана точка C с координатами (x_0, y_0), лежащая внутри данного выпуклого многоугольника, но не на его границе.

Алиса и Боб играют в игру, делая ходы по очереди, начиная с Алисы. На очередном ходу, игрок должен выбрать вершину многоугольника отличную от C и переместить её в точку C. Если в многоугольнике уже есть вершина в точке C, то эти вершины объединяются. Ход можно совершить только в случае, если существует *особая* точка, лежащая в многоугольнике до хода, но не лежащая в нём после. Гарантируется, что точка никогда не может оказаться на границе многоугольника после любого количества ходов.

После хода многоугольник не обязан быть выпуклым, а так же может стать вырожденным (то есть стать отрезком). Проигрывает тот, кто не может сделать ход.

Также есть q изменений двух видов:

- $+\ x\ y$, что означает, что точка с координатами (x, y) становится *особой*. Гарантируется, что до этого эта точка не была *особой*.
- $-\ x\ y$, что означает, что точка с координатами (x, y) перестаёт быть *особой*. Гарантируется, что до этого эта точка была *особой*.

После каждого изменения, а также в самом начале, определите, какой игрок должен победить при оптимальной игре обоих. После каждого изменения игра начинается с исходного многоугольника, с учетом примененных изменений к особым точкам.

Формат входных данных

Первая строка содержит три целых числа n, m и q ($3 \leqslant n \leqslant 10\,000$, $0 \leqslant m \leqslant 100\,000$, $0 \leqslant q \leqslant 1\,000\,000$) — количество точек в многоугольнике, количество *особых* точек и количество изменений.

Вторая строка содержит два целых числа x_0 и y_0 ($-10^9 \leqslant x_0, y_0 \leqslant 10^9$) — координаты точки C. Гарантируется, что точка лежит внутри данного многоугольника и не лежит на его границе.

Следующие n строк содержат по два целых числа x_i и y_i ($-10^9 \leqslant x_i, y_i \leqslant 10^9$) — координаты i-й точки многоугольника. Точки вводятся в порядке обхода против часовой стрелки.

Следующие m строк содержат по два целых числа x_i и y_i ($-10^9 \leqslant x_i, y_i \leqslant 10^9$) — координаты i-й *особой* точки. Гарантируется, что в любой момент времени точки различны и лежат внутри многоугольника.

Следующие q строк содержат описание запросов. В i-й из них находится символ c и два целых числа x и y ($c =$ «`+`» или «`-`» (без кавычек), $-10^9 \leqslant x, y \leqslant 10^9$) — описание очередного запроса.

- Если $c =$ «`+`», то точка (x, y) становится *особой*. Гарантируется, что до этого эта точка не была *особой*.
- Если $c =$ «`-`», то точка (x, y) перестаёт быть *особой*. Гарантируется, что до этого эта точка была *особой*.

Гарантируется, что в любой момент времени после любого количества корректных ходов никакая из особых точек не может оказаться на границе многоугольника.

Формат выходных данных

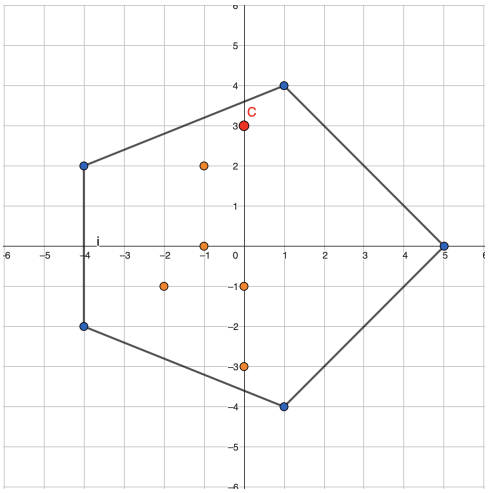
Выведите $q+1$ строку. В первой строке выведите «`Alice`» (без кавычек), если до всех изменений при оптимальной игре побеждает Алиса, и «`Bob`» (без кавычек) иначе. Далее в i-й строке выведите победителя игры после $(i-1)$-го изменения в аналогичном формате.

Пример

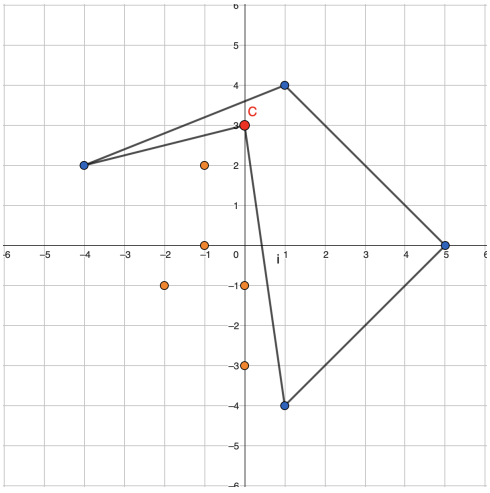
стандартный ввод	стандартный вывод
5 5 3 0 3 -4 -2 1 -4 5 0 1 4 -4 2 0 -1 -2 -1 -1 0 0 -3 -1 2 + 4 0 + -2 2 + 0 2	Alice Bob Bob Bob

Замечание

Рассмотрим многоугольник перед всеми изменениями:



На первом ходу Алиса может передвинуть вершину многоугольника с координатами (-4; -2), тогда многоугольник будет выглядеть так:



В этом состоянии многоугольника Боб уже не может сделать ход, а значит проигрывает.

Система оценки

Тесты к этой задаче состоят из 6 групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание, прохождение тестов из условия не требуется для некоторых групп. **Offline-проверка** означает, что результаты тестирования вашего решения на данной группе станут доступны только после окончания соревнования.

Группа	Баллы	Доп. ограничения			Необх. группы	Комментарий
		n	m	q		
0	0	–	–	–	–	Тесты из условия.
1	16	$n = 3$	$m \leqslant 3$	$q = 0$	–	
2	13	$n \leqslant 18$	$m \leqslant 18$	$q \leqslant 1$	1	
3	15	$n \leqslant 18$	$m \leqslant 18$	–	0 – 2	
4	17	$n \leqslant 5000$	$m \leqslant 5000$	$q \leqslant 1$	1, 2	
5	21	$n \leqslant 5000$	$m \leqslant 100\,000$	$q \leqslant 5000$	0 – 2, 4	
6	18	–	–	–	0 – 5	**Offline-проверка.**

Задача I. Парные дороги

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	4 секунды
Ограничение по памяти:	512 мегабайт

В некотором государстве есть n городов, между которыми пока что не построено ни одной дороги. В i-м городе живет w_i человек. Так же есть $n-1$ дорога, которую можно построить. i-я дорога при её строительстве соединит города u_i и v_i, а стоимость её строительства равна s_i.

Известно, что если построить все дороги, то от каждого города можно будет добраться до любого другого, используя только эти дороги. Другими словами, дороги образуют дерево.

В каждый из следующих k дней будет происходить следующее: В i-й день выбирается некоторый город c_i, а также две различные ещё не построенные дороги, соединяющие этот город с какими-то другими городами. После этого эти две дороги строятся, за это платится цена, равная суммарной стоимости строительства этих двух дорог. Город c_i в этом случае является *центральным* в день i.

Через k дней каждый город, являющийся *центральным* в хотя-бы один из k дней, принесёт ровно столько прибыли, сколько человек в нём живёт.

Выгодой назовём разность между прибылью, которую принесут города, и суммарной стоимостью построенных дорог. Найдите максимально возможную *выгоду*.

Формат входных данных

Первая строка содержит три целых числа n, k и t ($3 \leqslant n \leqslant 200\,000$, $1 \leqslant k \leqslant \frac{n-1}{2}$, $0 \leqslant t \leqslant 1$) — количество городов, количество пар дорог, которые надо построить, а также число t равное 1, если нужно найти какие именно дороги надо построить, и равное 0 иначе.

Вторая строка содержит n целых чисел w_1, w_2, …, w_n ($1 \leqslant w_i \leqslant 10^8$) — количество жителей в городах.

Следующие $n-1$ строк содержат по три целых числа u_i, v_i и s_i($1 \leqslant u_i, v_i \leqslant n$, $1 \leqslant s_i \leqslant 10^8$) — номера городов, которые соединяет i-я дорога и стоимость её строительства.

Гарантируется, что дороги образуют дерево, и что можно построить k пар дорог, удовлетворяющих условию задачи.

Формат выходных данных

В первой строке выведите единственное целое число — максимальную *выгоду*, которую можно получить после строительства ровно k пар дорог.

Если $t = 1$, то в следующих k строках выведите k троек чисел c_i, x_i и y_i ($1 \leqslant c_i, x_i, y_i \leqslant n$) — описание пар дорог, которые надо построить. Такая тройка задаёт пару из дороги (c_i, x_i) и дороги (c_i, y_i).

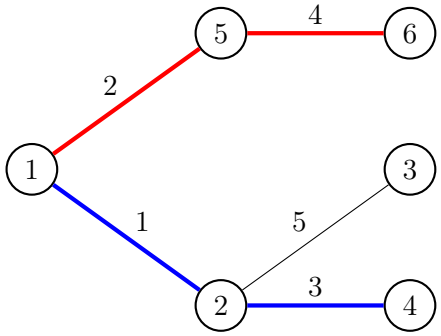
Если существует несколько способов получить максимальную *выгоду*, выведите любой из них.

Примеры

стандартный ввод	стандартный вывод
6 2 1 1 2 3 4 5 6 1 2 1 2 3 5 2 4 3 1 5 2 5 6 4	-3 5 6 1 2 4 1
8 3 0 4 5 1 2 3 1 3 5 2 1 15 7 1 5 4 8 1 8 5 2 7 8 1 6 7 5 3 7 7	-13

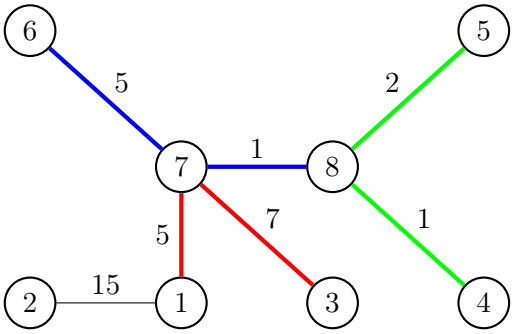
Замечание

В первом тесте из условия предлагается построить следующие дороги (одинаковым цветом отмечены дороги из одной пары):



Суммарная стоимость строительства этих дорог равна $2+4+1+3=10$. После строительства, города 2 и 5 принесут по 2 и 5 прибыли соответственно. Таким образом, *выгода* равна $2+5-10=-3$. Можно показать, что лучший ответ получить нельзя.

Во втором тесте из условия предлагается построить следующие дороги:



Суммарная стоимость строительства этих дорог равна $5+1+2+1+5+7=21$. После строительства, города 7 и 8 принесут по 3 и 5 прибыли соответственно. Таким образом, *выгода* равна $3+5-21=-13$. Можно показать, что лучший ответ получить нельзя.

Система оценки

Тесты к этой задаче состоят из 7 групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание,

прохождение тестов из условия не требуется для некоторых групп. **Offline-проверка** означает, что результаты тестирования вашего решения на данной группе станут доступны только после окончания соревнования.

Группа	Баллы	Доп. ограничения		Необх. группы	Комментарий
		n	t		
0	0	–	–	–	Тесты из условия.
1	13	$n \leqslant 200$	$t = 0$	–	
2	17	$n \leqslant 2000$	$t = 0$	1	
3	12	$n \leqslant 2000$	–	0 – 2	
4	19	–	$t = 0$	–	$u_i = i$, $v_i = i + 1$
5	11	–	–	4	$u_i = i$, $v_i = i + 1$
6	15	–	$t = 0$	1, 2, 4	**Offline-проверка.**
7	13	–	–	0 – 6	**Offline-проверка.**