Задача A. Любитель камней

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

Один мальчик очень любил камни, поэтому он мог тратить целые годы на их изучение. За эти долгие годы он понял, что у каждого камня есть вид, обозначаемый буквой латинского алфавита. Мальчик считает, что про камни должны знать все, поэтому он работает в музее, где показывает камни.

В музее предложено n экспонатов камней в порядке, задаваемом строкой s, на i-м стенде предложен камень вида s_i. К мальчику пришло m посетителей, причем каждый посетитель хочет посмотреть только на определенные камни, виды которых задаются строкой t. Мальчик поведет посетителя по стендам в порядке с 1 по n, и на для каждого стенда он может выбрать, показывать камень этого стенда посетителю или пропустить. Мальчик должен исполнить желание посетителя, но он хочет сделать просмотр камней максимально сложным.

Пусть мальчик покажет стенды с номерами $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant n$, тогда эти камни в порядке показа должны образовывать строку t, иными словами, должно выполняться $k = |t|$, и для всех $1 \leqslant j \leqslant k$ должно выполняться $s_{i_j} = t_j$. Сложностью просмотра называется минимальное расстояние между соседними показанными стендами, то есть $\min(i_2 - i_1, i_3 - i_2, \ldots, i_k - i_{k-1})$. Если невозможно выбрать такие k стендов, то просмотр называется невозможным, иначе мальчик выберет просмотр максимальной возможной сложности.

Определите для каждого посетителя, является ли его просмотр возможным, и если да, то какая может быть максимальная сложность просмотра камней.

Формат входных данных

Первая строка содержит два целых числа n и m ($1 \leqslant n, m \leqslant 200\,000$) — количество экспонатов в музее и количество посетителей.

Вторая строка содержит строку s длиной n, состоящую из строчных латинских букв — виды камней на экспонатах в музее.

В следующих m строках описываются посетители. В i-й из них содержится строка t_i, состоящая из строчных латинских букв ($2 \leqslant |t_i| \leqslant 200\,000$) — виды камней, которые хочет посмотреть посетитель i.

Гарантируется, что сумма всех $|t_i|$ не превосходит $400\,000$.

Формат выходных данных

Для каждого из m посетителей в отдельной строке выведите максимальную сложность просмотра экскурсии или -1, если экскурсию провести невозможно.

Примеры

стандартный ввод	стандартный вывод
7 3 abacaba aa acb ba	6 2 5
12 2 openolympiad oli goal	4 -1
8 5 abbaabba bab baba bbbb aaaa abbaabba	2 2 1 1 1

Замечание

Рассмотрим первый пример.

Для первого посетителя оптимально выбрать стенды с номерами $[1, 7]$, тогда сложность просмотра будет $7 - 1 = 6$.

Для второго посетителя оптимально выбрать стенды с номерами $[1, 4, 6]$, тогда сложность просмотра будет $\min(4 - 1, 6 - 4) = 2$.

Для третьего посетителя оптимально выбрать стенды с номерами $[2, 7]$, тогда сложность просмотра будет $7 - 2 = 5$.

Система оценки

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

Обозначим за T суммарную длину всех строк t.

Группа	Баллы	Доп. ограничения			Необх. группы	Комментарий
		n	m	T		
0	0	–	–	–	–	Тесты из условия.
1	16	$n \leqslant 20$	–	$T \leqslant 1000$	0	
2	14	$n \leqslant 500$	–	$T \leqslant 500$	0	
3	12	–	–	–	–	s и t состоят только из буквы ‘a’
4	16	–	$m = 1$	–	–	s и t состоят только из букв ‘a’ и ‘b’
5	11	–	$m = 1$	–	4	–
6	17	–	–	–	3, 4	s и t состоят только из букв ‘a’ и ‘b’
7	14	–	–	–	0 – 6	

Задача B. Отличительные черты

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 разрабатываете систему для помощи консультантам в магазинах смартфонов.

Все смартфоны в магазине упорядочены в ряд и пронумерованы целыми числами от 1 до n в том порядке, в котором они лежат.

Каждый смартфон обладает какими-то **отличительными чертами**, например, прочностью или большим аккумулятором. Всего различных отличительных черт m, и они пронумерованы целыми числами от 1 до m.

Самый частый вопрос, на который приходится отвечать консультантам: чем данный смартфон отличается от тех, которые лежат рядом с ним? Формализуем этот вопрос следующим образом:

Имея отрезок смартфонов с номерами от l_i до r_i включительно, а так же номер некоторого смартфона p_i на этом отрезке ($l_i \leqslant p_i \leqslant r_i$), сказать, сколько существует отличительных черт, которые есть у данного смартфона p_i, но нет ни у одного другого смартфона на отрезке $[l_i, r_i]$.

Чтобы облегчить работу консультантов, вам поручено разработать систему, способную эффективно отвечать на такие запросы.

Формат входных данных

Первая строка содержит три целых числа n, m и g ($1 \leqslant n, m \leqslant 500\,000$, $0 \leqslant g \leqslant 9$) — количество смартфонов в магазине, количество различных отличительных черт и номер группы для данного теста.

Каждая из следующих n строк описывает отличительные черты очередного смартфона в следующем формате:

В начале строки находится целое число k_i ($0 \leqslant k_i \leqslant m$) — количество отличительных черт у i-го смартфона. Затем в той же строке находятся k_i целых чисел $1 \leqslant a_{i,1} < \ldots < a_{i,k_i} \leqslant m$ — отличительные черты i-го смартфона в возрастающем порядке.

Следующая строка содержит целое число q ($1 \leqslant q \leqslant 500\,000$) — количество запросов.

Следующие q строк описывают запросы. В i-й из них содержится три целых числа l_i, r_i и p_i ($1 \leqslant l_i \leqslant p_i \leqslant r_i \leqslant n$) — параметры i-го запроса.

Обозначим за s суммарное количество отличительных черт у всех смартфонов ($s = \sum_{i=1}^{n} k_i$). Гарантируется, что $n, m \leqslant s \leqslant 500\,000$.

Формат выходных данных

Выведите q целых чисел — количество отличительных черт для каждого запроса.

Пример

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

Замечание

Рассмотрим пример.

В первом запросе у второго смартфона нет отличительных черт, поэтому ответ на запрос равен нулю.

Во втором запросе отрезок состоит из одного элемента, поэтому все отличительные черты четвертого смартфона подходят.

В третьем запросе отличительные черты третьего смартфона — $[1, 4]$, четвертого — $[2, 3, 4]$, пятого — $[1, 2]$. Среди всех отличительных черт четвертого смартфона только черта 3 есть только у него, поэтому ответ на запрос равен 1.

Система оценки

Тесты к этой задаче состоят из девяти групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание, что 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, s		
0	0	–	–	Тесты из условия.
1	10	$n, m, q \leqslant 500$	0	
2	7	$q \leqslant 5000, s \leqslant 10\,000$	0	
3	13	$q, s \leqslant 100\,000, m \leqslant 500$	0	
4	19	–	–	$p_i = l_i$
5	7	–	–	$l_i = 1$
6	10	–	–	$l_i \leqslant l_{i+1}, r_i \leqslant r_{i+1}$
7	12	$q, s \leqslant 100\,000$	0, 2, 3	–
8	7	$q, s \leqslant 200\,000$	0, 2, 3, 7	
9	15	–	0 – 8	**Offline-проверка.**

Задача C. Освещение дорог

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

Берляндия — страна с не очень развитой дорожной системой. Всего в Берляндии есть n городов и $n-1$ двухсторонняя дорога, i-я из которых соединяет города v_i и u_i. Известно, что между каждой парой городов можно добраться, используя только данные дороги.

Сейчас в Берляндии ночь и на некоторых дорогах надо включить освещение. Есть закон, созданный в целях экономии электричества, запрещающий включать освещение одновременно на двух дорогах, имеющих общий конец. Жителям Берляндии интересно максимальное количество дорог, на которых можно включить освещение так, чтобы не нарушить закон, поэтому они обратились к вам за помощью.

К сожалению, в Берляндии на некоторых дорогах может идти сильная метель, из-за чего связность некоторых пар городов может нарушиться. Изначально ни на какой дороге не идёт метель. Поступает q запросов двух типов:

1. Изменить погоду на дороге e_i ($1 \leqslant e_i \leqslant n-1$): если на дороге e_i сейчас не идёт метель, то метель начинается, и наоборот.

2. Требуется включить освещение на максимальном количестве дорог, таких что до их обоих концов можно добраться от города x_i, используя только дороги, на которых не идёт метель. Разумеется, освещение можно включать не нарушая закон, то есть не должно быть пары дорог со включенным освещением, выходящих из одного города. Другими словами, надо решить исходную задачу, если оставить только города, достижимые из x_i по дорогам, на которых не идёт метель.

Формат входных данных

Первая строка содержит единственное целое число g ($0 \leqslant g \leqslant 7$) — номер группы тестов.

Вторая строка содержит единственное целое число n ($2 \leqslant n \leqslant 300\,000$) — количество городов.

Следующие $n-1$ строки описывают дороги. В i-й из них содержатся два целых числа v_i и u_i ($1 \leqslant v_i, u_i \leqslant n$) — номера городов, соединённых i-й дорогой. Гарантируется, что между каждой парой городов можно добраться, используя только данные дороги.

Следующая строка содержит единственное целое число q ($1 \leqslant q \leqslant 300\,000$) — количество запросов.

В следующих q строках заданы запросы. В i-й из них в начале содержится целое число t_i ($1 \leqslant t_i \leqslant 2$).

- Если $t_i = 1$, то это запрос первого типа, далее строка содержит единственное целое число e_i ($1 \leqslant e_i \leqslant n-1$) — номер дороги, на которой меняется погода.

- Если $t_i = 2$, то это запрос второго типа, далее строка содержит единственное целое число x_i ($1 \leqslant x_i \leqslant n$). В этом случае надо решить исходную задачу, если оставить только города, достижимые из x_i по дорогам, на которых не идёт метель.

Формат выходных данных

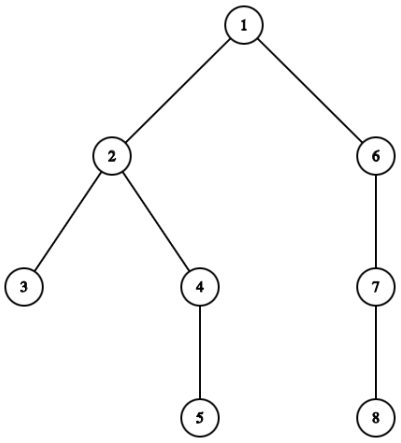
Для каждого запроса второго типа выведите максимальное число дорог, соединяющих города, достижимые из x_i, на которых можно включить освещение, не нарушая закон.

Пример

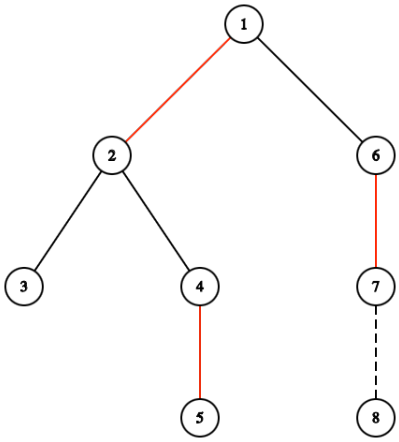
стандартный ввод	стандартный вывод
0 8 1 2 2 3 2 4 4 5 1 6 6 7 7 8 8 1 7 2 1 1 7 2 1 1 3 2 3 1 5 2 6	3 4 3 1

Замечание

В тестовом примере Берляндия изначально имеет следующий вид:

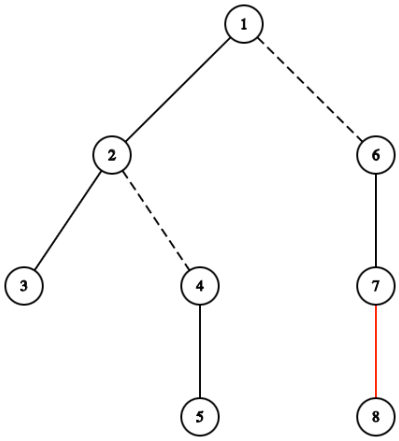


После первого запроса на дороге между городами 7 и 8 начинает идти метель. Далее приходит запрос про город 1. В этом случае мы рассматриваем все города, кроме 8-го, так как он не достижим из города 1. Ниже приведен один из оптимальных способов включить освещение на дорогах (красным отмечены дороги, на которых включено освещение):



Следующий запрос означает, что на дороге между городами 7 и 8 метель прекращается, то есть все возвращается в исходное положение.

В самом последнем запросе из вершины 6 достижимы только вершины 6, 7 и 8, поэтому можно включить освещение максимум на одной дороге, например:



Система оценки

Тесты к этой задаче состоят из семи групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание, что 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		
0	0	–	–	–	Тесты из условия.
1	14	$n \leqslant 100$	$q \leqslant 100$	0	
2	13	$n \leqslant 100\,000$	$q \leqslant 100\,000$	–	$v_i = i$, $u_i = i + 1$
3	10	$n \leqslant 100\,000$	$q = 1$	–	
4	12	$n \leqslant 100\,000$	$q \leqslant 100\,000$	–	$v_i = i + 1$, $u_i = \lfloor \frac{v_i}{2} \rfloor$
5	19	$n \leqslant 100\,000$	$q \leqslant 100\,000$	3	Метель не может закончиться
6	20	$n \leqslant 100\,000$	$q \leqslant 100\,000$	0 – 5	
7	12	–	–	0 – 6	**Offline-проверка.**

Задача D. Перекрашивание таблицы

Ограничение по времени: 1.5 секунд
Ограничение по памяти: 512 мегабайт

Данную задачу можно решать только на языках программирования C++ или Python3.

Захар очень любит играть в шахматы. Обычные черно-белые шахматные доски ему наскучили, поэтому он задумался, как можно разнообразить игру.

Шахматная доска представляет собой квадратную таблицу из n строк и n столбцов. Пронумеруем строки целыми числами от 0 до $n-1$ сверху вниз, а столбцы целыми числами от 0 до $n-1$ слева направо. Захар называет доску k-цветной, если клетка покрашена в цвет, равный остатку от деления номера идущей вверх-вправо диагонали клетки по модулю k. Диагонали нумеруются последовательными целыми числами с нуля, начиная с верхнего левого угла таблицы. Иными словами, номер диагонали, которой принадлежит клетка (i, j), равен $i + j$ по модулю k.

	0	1	2	3	4
0	0	1	2	0	1
1	1	2	0	1	2
2	2	0	1	2	0
3	0	1	2	0	1
4	1	2	0	1	2

Рисунок k-цветной доски для $n = 5, k = 3$.

Захар нашел на даче доску, и теперь хочет получить из нее k-цветную доску для какого-то положительного целого значения k. Какие-то t клеток таблицы уже покрашены, клетка $(r[i], c[i])$ покрашена в цвет $v[i]$, и Захару может потребоваться перекрасить их в правильный цвет. Непокрашенные клетки Захара не волнуют, потому что нанести краску проще, чем перекрасить уже имеющийся цвет.

Определите минимальное количество перекрашиваний, которое потребуется Захару, чтобы получить k-цветную доску для какого-то положительного целого значения k.

Формат решения

Это необычная задача. Она имеет формат тестирования с грейдером, где вам необходимо реализовать только функцию `solve` с решением. Эта функция будет вызвана тестирующей программой жюри (грейдером), и возвращаемое значение функции будет принято как решение задачи.

В частности, это означает, что в отправленном вами коде **не должно быть ввода или вывода**. Если вы пишете на языке C++, то ваш код **не должен** содержать функцию `main`. При необходимости вы можете реализовать произвольное количество вспомогательных функций, структур, классов и глобальных переменных, но весь код вашего решения должен находиться в одном файле.

Для решения на языке C++ вы должны реализовать следующую функцию:

```
int solve(int n, int t, std::vector<int> r, std::vector<int> c, std::vector<int> v)
```

Для решения на языках Python3 или Pypy3, вы должны реализовать следующую функцию:

```
def solve(n, t, r, c, v):
```

Здесь n и t — целые числа, а r, c и v — списки целых чисел длины t.

Обратите внимание, что жюри не гарантирует возможность сдачи задачи на полный балл на языках Python3 или Pypy3.

В обоих языках функция `solve` принимает следующие аргументы:

- n ($1 \leqslant n \leqslant 10^6$) — количество строк и столбцов в таблице.
- t ($1 \leqslant t \leqslant 10^6$) — количество покрашенных клеток.

- r $(0 \leqslant r[i] \leqslant n-1)$ — массив размера t, состоящий из номеров строк покрашенных клеток.
- c $(0 \leqslant c[i] \leqslant n-1)$ — массив размера t, состоящий из номеров столбцов покрашенных клеток.
- v $(0 \leqslant v[i] \leqslant 10^9)$ — массив размера t, состоящий из цветов покрашенных клеток.

Все покрашенные клетки пронумерованы в 0-индексации, соответственно для любого i $(0 \leqslant i < t)$ i-я покрашенная клетка находится на пересечении $r[i]$-й строки и $c[i]$-го столбца и имеет цвет $v[i]$. Гарантируется, что все пары $(r[i], c[i])$ различны.

Функция `solve` должна возвращать одно целое число — минимальное количество исходно покрашенных клеток, которые придется перекрасить, чтобы сделать доску k-цветной.

Гарантируется, что функция `solve` будет вызвана ровно один раз за время запуска программы.

Тестирование

Вам предоставлены шаблоны файлов, в которых вы можете писать решения: `table.cpp` и `table.py`. Также на языке C++ вам предоставлен заголовочный файл `table.h`, содержащий определение функции `solve`.

Для удобства тестирования вам предоставлены грейдеры — файлы `grader.cpp` и `grader.py`. В этих файлах реализован ввод входных данных со стандартного потока ввода, запуск функции `solve` и вывод на стандартный поток вывода возвращаемого значения функции `solve`. В тестирующей системе эти файлы грейдеров могут отличаться.

Чтобы скомпилировать ваш код на языке C++, написанный в файле `table.cpp`, используйте команду

```
g++ -std=c++20 grader.cpp table.cpp -o grader
```

После выполнения данной команды будет создан исполняемый файл грейдера `grader` или `grader.exe`, в зависимости от вашей операционной системы, запустив который можно ввести тест в формате, указанном далее.

Чтобы запустить ваш код на языке Python, написанный в файле `table.py`, используйте команду `python3 grader.py`, далее можно ввести тест в формате, указанном далее.

Если компиляция через команды вызывает у вас трудности, для локального тестирования вы можете скопировать реализацию ввода и вывода из файлов `grader` в файлы `table` и запускать файлы `table.cpp` или `table.py`. При этом перед отправкой решения в тестирующую систему вам нужно будет стереть эти реализации ввода и вывода (в частности, вам нужно будет стереть функцию `main`, если вы пишете на C++).

Формат входных данных

Предоставленный вам грейдер читает входные данные в следующем формате:

В первой строке задаются два целых числа n и t $(1 \leqslant n, t \leqslant 10^6)$ — количество строк и столбцов таблицы, а также количество уже покрашенных клеток.

Вторая строка содержит t целых чисел $r[i]$ $(0 \leqslant r[i] \leqslant n-1)$ — номера строк покрашенных клеток.

Третья строка содержит t целых чисел $c[i]$ $(0 \leqslant c[i] \leqslant n-1)$ — номера столбцов покрашенных клеток.

Четвертая строка содержит t целых чисел $v[i]$ $(0 \leqslant v[i] \leqslant 10^9)$ — цвета уже покрашенных клеток.

Формат выходных данных

Грейдер выводит возвращаемое значение функции `solve`.

Пример

тест	ответ
5 4 0 0 1 4 0 1 0 0 2 1 4 1	2

Замечание

В примере показан образец входных данных для предоставленного вам грейдера. В нем изначально покрашены четыре клетки: $(0, 0)$ в цвет 2, $(0, 1)$ в цвет 1, $(1, 0)$ в цвет 4, и $(4, 0)$ в цвет 1. Оптимально выбрать $k = 3$, рисунок таблицы приведен в условии задачи.

Нужно перекрасить две клетки: $(0, 0)$ с цвета 2 на цвет 0, и $(1, 0)$ с цвета 4 на цвет 1.

Система оценки

Тесты к этой задаче состоят из четырех 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	t		
0	0	–	–	–	Тесты из условия.
1	17	$n \leqslant 100$	$t \leqslant 100$	0	
2	21	$n \leqslant 1\,000$	–	0, 1	
3	34	$n \leqslant 100\,000$	$t \leqslant 100\,000$	0, 1	
4	28	–	–	0 – 3	

Задача E. Игровая зависимость Егора

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

Мальчик Егор все время играет в новомодную игру «TOTA», в ней есть n типов героев. Линией из героев длины k назовем последовательность из героев с типами героев $b_1, b_2, \ldots, b_k$. Егор называет линию из героев длины k *ужасной*, если каждый тип героя от 1 до k встречается в этой линии ровно один раз. Иными словами, линия длины k ужасна для Егора, если для любого $1 \leqslant i \leqslant k$ существует позиция в линии j, такая что $b_j = i$.

У Егора игровая зависимость, поэтому каждый день он играет в игру со своим другом Тимофеем. Так как игра еще не до конца разработана, то каждый день для построения линий доступна одна и та же последовательность героев длины n, где i-й герой имеет тип a_i. Тимофей будет играть с Егором q дней. Каждый день фиксируется некоторое число l_i – номер первого героя, которого надо взять в линию. Соответственно, в i-й день Тимофей может собрать против Егора линию из героев с номерами $a_{l_i}, a_{l_i+1}, \ldots a_{r_i}$ для любого r_i, такого что $l_i \leqslant r_i \leqslant n$.

Тимофей очень хочет, чтобы Егор перестал все время играть в «TOTA» и вместо этого занялся учебой. Для этого Тимофей хочет лишать Егора всей радости от игры. Каждый день у Егора есть x_i радости, которую Тимофей хочет отнять, поставив против него ужасную линию. Известно, что радость Егора будет отнята, если длина ужасной линии не меньше количества радости, то есть в день i вся радость Егора пропадет, если против него выставлена линия длины хотя бы x_i.

Тимофей хочет лишать Егора радости как можно чаще, поэтому для каждого i от 1 до q он хочет узнать минимальную длину линии, которая лишит Егора радости в день i, а также количество возможных вариантов линий в день i, которые лишат радости Егора. К сожалению, из-за частой игры в «TOTA» когнитивные способности Тимофея понизились, и для решения этой задачи ему потребуется ваша помощь.

Формат входных данных

Первая строка содержит единственное целое число g ($0 \leqslant g \leqslant 8$) — номер группы тестов.

Вторая строка содержит два целых числа n и q ($1 \leqslant n, q \leqslant 10^6$) — количество героев и количество дней соответственно.

Третья строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant n$) — типы героев в исходном списке.

В i-й из следующих q строк содержатся два целых числа l_i и x_i ($1 \leqslant l_i \leqslant n, 1 \leqslant x_i \leqslant n - l_i + 1$) — номер первого игрока линии в i-й день и количество радости Егора в i-й день.

Формат выходных данных

Выведите q строк, в i-й из них выведите ответ для i-го дня — минимальную длину ужасной линии и количество ужасных линий, которые лишат Егора радости в i-й день. Если все линии в i-й день не лишат Егора радости, выведите `-1 0`.

Примеры

стандартный ввод	стандартный вывод
0 6 3 1 4 2 3 1 4 1 1 2 1 3 1	1 2 4 1 3 2
0 3 1 3 3 1 1 1	-1 0

Замечание

Рассмотрим первый пример.

В первый день Егор имеет показатель радости $x_1 = 1$, поэтому чтобы лишить его радости требуется ужасная линия длины хотя бы 1, но начиная с героя $l_1 = 1$ существует две ужасных линии:

- 1
- $1, 4, 2, 3$

Минимальная длина ужасной линии — 1, а их количество равняется двум.

Во второй день уже новое число $l_2 = 2$, для него существует единственная ужасная линия: $4, 2, 3, 1$. Обратите внимание, что линия $4, 2, 3, 1, 4$ не является ужасной, потому что герой типа 4 встречается на ней дважды.

В третий день линии формируются начиная с героя под номером $l_3 = 3$, в этот день существует две ужасных линии:

- $2, 3, 1$
- $2, 3, 1, 4$

Система оценки

Тесты к этой задаче состоят из восьми групп. Баллы за каждую группу ставятся только при прохождении всех тестов группы и всех тестов некоторых из предыдущих групп. Обратите внимание, что 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		
0	0	–	–	–	Тесты из условия.
1	8	$n \leqslant 1\,000$	$q \leqslant 1\,000$	0	
2	7	$n \leqslant 5\,000$	$q \leqslant 5\,000$	0, 1	
3	9	$n \leqslant 200\,000$	$q \leqslant 200\,000$	–	$a_i = (i-1) \mod x + 1$ для фиксированного $x > 0$
4	12	$n \leqslant 200\,000$	$q \leqslant 200\,000$	–	a — перестановка чисел от 1 до n
5	19	$n \leqslant 200\,000$	$q \leqslant 200\,000$	–	$x_i = 1$ для всех запросов
6	10	$n \leqslant 200\,000$	$q \leqslant 200\,000$	0 – 5	
7	14	$n \leqslant 500\,000$	$q \leqslant 500\,000$	0 – 6	
8	21	–	–	0 – 7	**Offline-проверка.**

Задача F. Инопланетные омофоны

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	1024 мегабайта

Окарун одержим идеей, что инопланетяне существуют. В детстве он всяческими способами пытался с ними связаться, но безуспешно. Это неудивительно, ведь инопланетяне говорят на совершенно другом языке! Однажды он наткнулся на статью, где было сказано, что инопланетяне на самом деле используют латинские строчные буквы для записи, но читают слова совершенно по другому.

В инопланетном языке есть набор из $n+26$ различных звуков, каждый из которых обозначается строкой из букв латинского алфавита s_i. Известно, что для любого $1 \leqslant i \leqslant 26$, i-й звук обозначается строкой длины один, состоящей из i-й по счету буквы латинского алфавита. Для любого $i > 26$ звук с номером i обозначается строкой, состоящей хотя бы из двух символов латинского алфавита.

Когда инопланетянин хочет прочитать слово x, он начинает читать его с первой позиции. Когда инопланетянин находится в i-й позиции, он ищет в наборе звуков такой звук s_j, что он входит как подстрока[†] в x, начиная с позиции i. Если таких звуков несколько, то он выбирает такой звук s_j, что его длина **максимальна**. Далее он выговаривает этот звук и переходит к позиции $i + |s_j|$ и читает слово дальше, пока не прочитает его полностью. Заметьте, что мы всегда можем прочитать любое слово, потому что все латинские буквы являются звуками.

Также выяснилось, что некоторые инопланетные звуки на самом деле звучат одинаково, но пишутся по разному. Это означает, что существуют некоторые пары строк-звуков s_i и s_j $(i \neq j)$, такие что $s_i \neq s_j$, но звуки, обозначающиеся этими строками, одинаковые. Обратите внимание, что если s_i и s_j считаются одним звуком, s_j и s_l считаются одним звуком, то s_i и s_l тоже считаются одним звуком.

Инопланетяне называют некоторые слова *омофонами* — слова, которые звучат одинаково, а их написание может как совпадать, так и различаться. Другими словами, если у нас есть два слова v и w, то инопланетяне их называют омофонами, если последовательность звуков, которые выговаривают инопланетяне, читая слово v, совпадает с последовательностью звуков, которые выговаривают инопланетяне, читая слово w.

Окарун написал некоторый текст t, состоящий из строчных латинских букв. В один момент его заинтересовало рассмотреть q пар подстрок текста. В каждой такой паре подстрок первая подстрока с a_i-го по b_i-й символ в тексте t включительно, вторая от c_i-го до d_i-го символов в тексте t включительно. Для каждой пары подстрок Окарун хочет узнать, являются ли эти подстроки омофонами, если их читать как слова инопланетного языка.

[†]Подстрока некоторой строки s — это строка, которая может быть получена удалением некоторого числа символов (возможно, ноль) в начале строки s и некоторого числа символов (возможно, ноль) в конце строки s.

Формат входных данных

Первая строка содержит непустую строку t $(1 \leqslant |t| \leqslant 500\,000)$, состоящую из строчных латинских букв — текст, написанный Окаруном.

Вторая строка содержит два целых числа n и k $(0 \leqslant n \leqslant 500\,000,\ 0 \leqslant k < n + 26)$ — число звуков в наборе, которые имеют длину не меньше двух и число пар одинаковых звуков, выписанных Окаруном.

В следующих n строках описываются звуки с номерами от 27-го. В i-й из них содержится непустая строка s_{i+26} $(2 \leqslant |s_{i+26}| \leqslant 10^6)$, состоящая из строчных латинских букв — запись $(i + 26)$-го звука. Гарантируется, что все строки звуков различны. Обратите внимание, что во входных данных нет строк от «`a`» до «`z`», тем не менее в каждом тесте они считаются звуками с номерами от 1 до 26.

В следующих k строках описываются пары одинаковых звуков, исходно выписанные Окаруном. Каждая из них содержит по паре целых чисел x_i и y_i $(1 \leqslant x_i, y_i \leqslant n + 26;\ x_i \neq y_i)$ — пара номеров звуков, которые Окарун считает одинаковыми по звучанию. Гарантируется, что каждая пара номе-

ров встречается не более одного раза. Обратите внимание, что одинаковость некоторых других пар звуков может следовать из данного набора.

Следующая строка содержит одно целое число q ($1 \leqslant q \leqslant 300\,000$) — число запросов пар подстрок текста, для которых нужно узнать, являются ли они инопланетными омофонами.

В следующих q строках описываются запросы. i-я из них содержит четыре целых числа a_i, b_i, c_i, d_i ($1 \leqslant a_i \leqslant b_i \leqslant |t|$, $1 \leqslant c_i \leqslant d_i \leqslant |t|$), которые задают подстроки текста — первая подстрока от позиции a_i до позиции b_i (включительно) в тексте t, вторая — от позиции c_i до позиции d_i (включительно) в тексте t.

Обозначим за S сумму длин звуков $\sum |s_i|$ (без учёта звуков «a»-«z»). Гарантируется, что $S \leqslant 10^6$.

Формат выходных данных

Выведите q строк. Если пара строк из i-го запроса являются одинаковыми по звучанию, в i-й строке выведите «`Yes`» (без кавычек), иначе в i-й строке выведите «`No`» (без кавычек).

Пример

стандартный ввод	стандартный вывод
abracadabra 2 3 cada ca 1 27 1 28 1 4 4 5 11 1 4 4 6 5 7 5 7 5 8 2 5 2 5	Yes Yes No Yes

Замечание

В первом запросе:

- Первая подстрока **cadabra** читается по звукам: **cada**, **b**, **r**, **a**;
- Вторая подстрока **abra** читается по звукам: **a**, **b**, **r**, **a**.

Звуки **cada** и **a** указаны как одинаковые (пара (1, 27)), поэтому эти две подстроки являются инопланетными омофонами.

Во втором запросе:

- Первая подстрока **aca** читается по звукам: **a**, **ca**;
- Вторая подстрока **cad** читается по звукам: **ca**, **d**.

Звуки **a** и **ca** указаны как одинаковые (пара (1, 28)), звуки **ca** и **d** тоже (так как одинаковы звуки с номерами (1, 4) и с номерами (1, 28), то звуки с номерами 4 и 28 тоже одинаковы) поэтому эти две подстроки тоже являются инопланетными омофонами.

В третьем запросе:

- Первая подстрока **cad** читается по звукам: **ca**, **d**;
- Вторая подстрока **cada** читается по звукам: **cada**.

Число звуков уже не совпадает, поэтому эти две подстроки точно не инопланетные омофоны.

В четвёртом запросе даны две одинаковые подстроки, поэтому они читаются одинаково и являются инопланетными омофонами.

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, что 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
		$\lvert t\rvert$	S	q		
0	0	–	–	–	–	Тесты из условия.
1	8	$\lvert t\rvert \leqslant 500$	$S \leqslant 500$	$q \leqslant 500$	0	
2	7	$\lvert t\rvert \leqslant 6\,000$	$S \leqslant 6\,000$	$q \leqslant 6\,000$	–	$b_i = d_i = \lvert t\rvert$
3	9	$\lvert t\rvert \leqslant 6\,000$	$S \leqslant 6\,000$	$q \leqslant 200\,000$	2	$b_i = d_i = \lvert t\rvert$
4	15	$\lvert t\rvert \leqslant 200\,000$	$S \leqslant 200\,000$	$q \leqslant 200\,000$	2, 3	$b_i = d_i = \lvert t\rvert$
5	8	$\lvert t\rvert \leqslant 6\,000$	$S \leqslant 6\,000$	$q \leqslant 200\,000$	–	$a_i = c_i = 1$
6	6	$\lvert t\rvert \leqslant 500$	$S \leqslant 500$	$q \leqslant 200\,000$	0, 1	
7	7	$\lvert t\rvert \leqslant 6\,000$	$S \leqslant 6\,000$	$q \leqslant 6\,000$	0 – 2	
8	10	$\lvert t\rvert \leqslant 6\,000$	$S \leqslant 200\,000$	$q \leqslant 6\,000$	0 – 2, 7	
9	19	$\lvert t\rvert \leqslant 200\,000$	$S \leqslant 400\,000$	$q \leqslant 200\,000$	0 – 8	
10	11	–	–	–	0 – 9	**Offline-проверка.**

Задача G. Сверхбыстрый поезд

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

Берляндия, 2224 год. В Берляндии есть n городов и m двухсторонних железнодорожных путей, каждый из которых соединяет некоторую пару городов. Известно, что железнодорожная сеть Берляндии связна, что означает, что от любого города можно добраться до любого другого, двигаясь по путям (возможно, через промежуточные города).

Берляндские железные дороги (БЖД) хотят запустить пассажирский поезд, который будет ездить между городами 1 и n. Поскольку поездка может быть очень долгой, то поезд решили ускорить, оснастив его машиной времени.

Машина времени работает следующим образом: перед отправкой из любого города машину времени можно включить или выключить. Если она выключена, то поезд доедет до следующего города за 1 час по любой железной дороге. Если же машина времени включена, то поезд доедет до следующего города за -1 час и прибудет в следующий город на час раньше времени отправления.

БЖД необходимо составить маршрут для этого поезда, при этом должны выполняться следующие условия:

1. Маршрут должен начаться в городе 1 и завершиться в городе n, посещая некоторые промежуточные города. Каждая пара посещенных подряд городов должна быть соединена железнодорожными путями.

2. В процессе движения поезд может посещать уже посещенные города, в частности, поезд может несколько раз проезжать через город 1 или n.

3. В целях избежания пространственно-временных парадоксов, в любой момент времени маршрут должен занимать неотрицательное количество часов. Это означает, что время прибытия в поезда в любой город должно быть не меньше времени первого отправления из города 1.

4. За время следования поезда он может проехать не более чем через $3n + 2$ городов, включая стартовый и конечный, так как иначе маршрут посчитают слишком сложным и непривлекательным. Если поезд несколько раз за время следования проедет через некоторый город, этот город будет учитываться несколько раз.

5. Время прибытия в город n должно быть минимальным возможным (но не меньше времени первого отправления из города 1).

Помогите БЖД составить наиболее подходящий маршрут поезда. Обратите внимание, что вам не нужно минимизировать число городов, через которые проедет поезд.

Формат входных данных

Первая строка содержит два целых числа n и m ($2 \leqslant n \leqslant 100\,000$, $n - 1 \leqslant m \leqslant 100\,000$) — количество городов и железных дорог соответственно.

В следующих m строках описываются железнодорожные пути. В i-й строке содержатся два целых числа u_i и v_i ($1 \leqslant u_i, v_i \leqslant n$) — номера городов, между которыми есть железная дорога.

Гарантируется, что из любого города можно попасть в любой другой город по железнодорожным путям. Также гарантируется, что нет железнодорожных путей из города в самого себя и нескольких путей, соединяющих одну и ту же пару городов.

Формат выходных данных

В первой строке выведите два целых числа t и k ($t \geqslant 0, 2 \leqslant k \leqslant 3n + 2$) — минимальное время поездки поезда и количество городов, которые посетит этот поезд, включая стартовый и конечный.

В следующей строке выведите $2k - 1$ значений, составляющих маршрут поезда. При этом 1-е, 3-е, …, $(2k - 1)$-е значения должны являться целыми числами, обозначающими города посещения

в порядке следования маршрута. А 2-е, 4-е, ..., $(2k-2)$-е значения должны равняться +, если надо выключить машину времени на этих путях, или -, если машину времени надо включить.

Примеры

стандартный ввод	стандартный вывод
5 4 2 1 3 2 3 5 4 2	1 4 1 + 2 + 3 - 5
5 5 1 2 2 3 3 4 4 5 5 2	0 9 1 + 2 + 3 + 4 + 5 - 2 - 3 - 4 - 5

Замечание

В первом примере поезд за час приезжает в город 2, после чего ещё за один час приезжает в город 3, после чего включается машина времени, и поезд приезжает в город 4 за -1 час. В сумме поездка заняла $1 + 1 - 1 = 1$ час.

Во втором примере поезд за 4 часа доезжает до города 5, после чего едет назад в город 2, и оттуда снова в город 5, но уже потратив -4 часа. В сумме поездка заняла $4 - 4 = 0$ часов.

Система оценки

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

Если страна **2-раскрашиваема**, то это означает, что каждый город можно раскрасить в один из двух цветов так, чтобы каждая железная дорога соединяла города двух разных цветов.

Группа	Баллы	Доп. ограничения		Необх. группы	Комментарий
		n	m		
0	0	–	–	–	Тесты из условия.
1	17	$n \leqslant 10$	$m = n - 1$	–	
2	13	–	$m = n - 1$	1	
3	12	–	–	0 – 2	Берляндия 2-раскрашиваема.
4	26	–	–	–	$u_i = i, v_i = i + 1$ для $1 \leqslant i \leqslant n - 1$.
5	32	–	–	0 – 4	

Задача H. Классическая задача на дерево

Ограничение по времени:	4 секунды
Ограничение по памяти:	1024 мегабайта

Данную задачу можно решать только на языках программирования C++ или Python3.

Одним утром Егор вышел на улицу и увидел на асфальте классики — n квадратов, в которых записаны какие-то числа, в i-м квадрате записано число a_i. Но это были необычные классики, каждый квадрат связан с какими-то другими квадратами.

Игра в классики заключается в том, чтобы начать из какого-то квадрата совершать прыжки в какой-то другой квадрат, связанный с текущим, причем нельзя прыгать в один и тот же квадрат больше одного раза. Игра может закончиться в любой момент, тогда последовательность посещенных квадратов $v_0, v_1, \ldots, v_{k-1}$ будет называться последовательностью прыжков.

Но на асфальте были нарисованы не обычные классики, а в виде дерева, то есть от каждого квадрата до другого можно добраться единственным образом с помощью какой-то последовательности прыжков. «Крутостью» последовательности прыжков будет считаться количество таких i, что $a_{v_i} = i$.

Егор захотел узнать для некоторых пар квадратов u, v крутость последовательности прыжков таких, что $v_0 = u, v_{k-1} = v$, чтобы узнать, стоит ли совершать такую последовательность прыжков или нет.

Егор захотел показать всю свою крутость, но ему постоянно что-то мешает. Поэтому он иногда стирает число в s-м квадрате и пишет новое число x. Но так как считать чиселки для Егора является тяжелой задачей, он обратился к вам за помощью.

В следующие q минут вы должны помочь Егору в разрешении его вопросов о крутости прыжков. В минуту i Егор может совершить одно из двух действий:

1. Стереть число в квадрате s и записать вместо него число x.
2. Спросить про пару квадратов u, v крутость последовательности прыжков из квадрата u в квадрат v.

От вас требуется ответить на каждый из вопросов Егора.

Формат решения

Это необычная задача. Она имеет формат тестирования с грейдером, где вам необходимо реализовать только функцию `solve` с решением. Эта функция будет вызвана тестирующей программой жюри (грейдером), и возвращаемое значение функции будет принято как решение задачи.

В частности, это означает, что в отправленном вами коде **не должно быть ввода или вывода**. Если вы пишете на языке C++, то ваш код **не должен** содержать функцию `main`. При необходимости вы можете реализовать произвольное количество вспомогательных функций, структур, классов и глобальных переменных, но весь код вашего решения должен находиться в одном файле.

Для решения на языке C++ вы должны реализовать следующую функцию:

```
std::vector<int> solve(int n, int q, std::vector<int> a, std::vector<int> p,
                       std::vector<int> qt, std::vector<int> qx, std::vector<int> qy);
```

Для решения на языках Python3 или Pypy3, вы должны реализовать следующую функцию:

```
def solve(n, q, a, p, qt, qx, qy):
```

Здесь n и q — целые числа; a и p — списки целых чисел длины n; qt, qx и qy — списки целых чисел длины q.

Обратите внимание, что жюри не гарантирует возможность сдачи задачи на полный балл на языках Python3 или Pypy3.

В обоих языках функция `solve` принимает следующие аргументы:

- n ($1 \leqslant n \leqslant 10^6$) — количество квадратов.

- q ($1 \leqslant q \leqslant 10^6$) — количество запросов.
- a ($0 \leqslant a[i] \leqslant n$) — массив размера n, состоящий из чисел, которые исходно записаны в квадратах.
- p ($-1 \leqslant p[i] < n$) — массив размера n, где $p[i]$ равняется номеру родителя квадрата i если представлять классики в виде дерева. У корня дерева $p[i]$ будет равен -1. Гарантируется, что массив задает корректное дерево.
- qt ($1 \leqslant qt[i] \leqslant 2$) — массив размера q, состоящий из типов запросов.
- qx и qy — массивы, задающие запросы.

Для каждого $0 \leqslant i < q$, i-й запрос задается следующим образом:

- Если $qt[i] = 1$, то в i-м запросе требуется стереть число в квадрате с номером $qx[i]$ и записать на его место число $qy[i]$. В этом случае верны следующие ограничения: $0 \leqslant qx[i] < n$, $0 \leqslant qy[i] \leqslant n$.
- Если $qt[i] = 2$, то в i-м запросе требуется найти крутость последовательности прыжков из квадрата $qx[i]$ в квадрат $qy[i]$. В этом случае верны ограничения: $0 \leqslant qx[i], qy[i] < n$.

Все квадратики и запросы пронумерованы в 0-индексации.

Функция `solve` возвращает массив, состоящий из ответов на запросы второго типа. Этот массив должен иметь длину равную числу запросов второго типа. На языке C++ данный массив возвращается в типе `vector<int>`, на языке Python3 данный массив возвращается в виде списка из целых чисел.

Гарантируется, что функция `solve` будет вызвана ровно один раз за время запуска программы.

Тестирование

Вам предоставлены шаблоны файлов с решением `tree.cpp` и `tree.py`. Также на языке C++ вам предоставлен заголовочный файл `tree.h`, содержащий определение функции `solve`.

Для удобства тестирования вам предоставлены грейдеры — файлы `grader.cpp` и `grader.py`. В этих файлах реализован ввод входных данных со стандартного потока ввода, запуск функции `solve` и вывод на стандартный поток вывода возвращаемого значения функции `solve`. В тестирующей системе эти файлы грейдеров могут отличаться.

Чтобы скомпилировать ваш код `tree.cpp` на языке C++, используйте команду

```
g++ -std=c++20 grader.cpp tree.cpp -o grader
```

После выполнения данной команды будет создан исполняемый файл грейдера `grader` или `grader.exe`, в зависимости от вашей операционной системы, запустив который можно ввести тест в формате, указанном далее.

Чтобы запустить ваш код на языке Python, написанный в файле `tree.py`, используйте команду `python3 grader.py`, далее можно ввести тест в формате, указанном далее.

Если компиляция через команды вызывает у вас трудности, для локального тестирования вы можете скопировать реализацию ввода и вывода из файлов `grader` в файлы `tree` и запускать файлы `tree.cpp` или `tree.py`. При этом перед отправкой решения в тестирующую систему вам нужно будет стереть эти реализации ввода и вывода (в частности, вам нужно будет стереть функцию `main`, если вы пишете на C++).

Формат входных данных

Грейдер читает тест в следующем формате:

В первой строке задаются два целых числа n, q ($1 \leqslant n, q \leqslant 10^6$) — количество квадратов в классиках и количество действий Егора.

В следующей строке задаются n целых чисел $a_0, a_1, \ldots, a_{n-1}$ ($0 \leqslant a_i \leqslant n$) — изначальные числа в квадратах.

В следующей строке задаются n целых чисел $p_0, p_1, \ldots p_{n-1}$ ($-1 \leqslant p_i < n$), задающие родителей вершин в дереве. Если $p_i = -1$, то вершина i является корнем дерева, иначе вершина p_i является

родителем вершины i в дереве. Гарантируется, что данный массив предков образует корректное корневое дерево.

В следующих q строках задаются запросы. Для любого $0 \leqslant i < q$, в i-й из них в зависимости от типа запроса он задан в следующем формате:

1 s x $(0 \leqslant s < n, 0 \leqslant x \leqslant n)$ — Егор стирает число в квадрате s и пишет там число x. В аргументах функции `solve` верно, что $qt[i] = 1, qx[i] = s, qy[i] = x$.

2 u v $(0 \leqslant u, v < n)$ — Егор спрашивает про крутость последовательности прыжков из u в v. В аргументах функции `solve` верно, что $qt[i] = 2, qx[i] = u, qy[i] = v$.

Формат выходных данных

Грейдер выводит результаты функции `solve` — ответы на запросы второго типа.

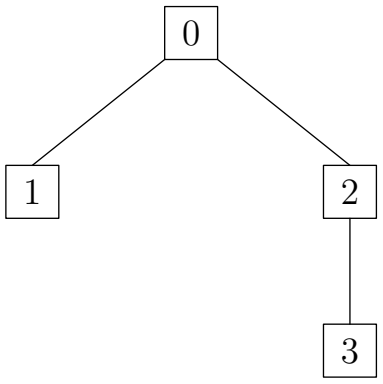
Примеры

тест	ответ
4 3 1 1 2 2 -1 0 0 2 2 0 3 1 2 1 2 0 3	1 2
5 5 0 1 2 3 4 1 2 -1 2 3 2 0 4 1 0 1 1 1 0 2 0 4 2 1 0	5 3 2

Замечание

В примерах указаны входные и выходные данные грейдера.

В первом примере дерево из квадратиков выглядит следующим образом:



Для первого запроса рассмотрим последовательность прыжков из 0 в 3:

1. $v_0 = 0, a_0 = 1 \neq 0$, поэтому этот квадрат не прибавляет Егору крутости.
2. $v_1 = 2, a_2 = 2 \neq 1$, поэтому и этот квадрат не прибавляет крутости к данной последовательности прыжков.
3. $v_2 = 3, a_3 = 2 = 2$, данный прыжок является крутым.

Отсюда крутость в первом запросе 1.

После второго запроса в квадратике под номером 2 теперь стоит число 1. Поэтому во втором запросе прыжок в квадрат под номером 2 становится крутым, теперь ответ на этот запрос 2.

Система оценки

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

Обозначим за c_i количество квадратов, связанных с i-м квадратом.

Группа	Баллы	Доп. ограничения		Необх. группы	Комментарий
		n	q		
0	0	–	–	–	Тесты из условия.
1	10	$n \leqslant 1\,000$	$q \leqslant 1\,000$	0	
2	11	$n \leqslant 200\,000$	$q \leqslant 5\,000$	0, 1	
3	14	$n \leqslant 200\,000$	$q \leqslant 200\,000$	–	Нет запросов первого типа
4	5	$n \leqslant 200\,000$	$q \leqslant 200\,000$	–	Для ровно одного квадрата $c_i = 2$, для остальных $c_i \leqslant 3$
5	11	$n \leqslant 200\,000$	$q \leqslant 200\,000$	–	$c_i \leqslant 2$
6	10	$n \leqslant 200\,000$	$q \leqslant 200\,000$		Для всех запросов второго типа $v_i = 0$
7	9	$n \leqslant 100\,000$	$q \leqslant 100\,000$	0, 1	
8	8	$n \leqslant 200\,000$	$q \leqslant 200\,000$	0 – 7	
9	17	$n \leqslant 500\,000$	$q \leqslant 500\,000$	0 – 8	
10	1	$n \leqslant 600\,000$	$q \leqslant 600\,000$	0 – 9	**Offline-проверка.**
11	1	$n \leqslant 700\,000$	$q \leqslant 700\,000$	0 – 10	**Offline-проверка.**
12	1	$n \leqslant 800\,000$	$q \leqslant 800\,000$	0 – 11	**Offline-проверка.**
13	1	$n \leqslant 900\,000$	$q \leqslant 900\,000$	0 – 12	**Offline-проверка.**
14	1	–	–	0 – 13	**Offline-проверка.**

Задача I. Цветной диаметр

Имя входного файла:	`стандартный ввод` или `input.txt`
Имя выходного файла:	`стандартный вывод` или `output.txt`
Ограничение по времени:	7 секунд
Ограничение по памяти:	512 мегабайт

На плоскости задано n точек с координатами (x_i, y_i), у каждой точки есть цвет c_i. Каждая точка движется с постоянной скоростью на вектор (dx_i, dy_i) в секунду. Это означает, что каждую секунду к x-координате точки i прибавляется значение dx_i, а к y-координате точки прибавляется значение dy_i.

По всем целочисленным моментам времени за первые T секунд найдите минимум максимального расстояния между точками разных цветов. Другими словами, найдите такой целочисленный момент времени t ($0 \leqslant t \leqslant T$), что максимальное расстояние между точками разных цветов в этот момент времени минимально в сравнении с любым другим целочисленным временем от 0 до T.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке находится одно целое число t ($1 \leqslant t \leqslant 100\,000$) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа n и T ($2 \leqslant n \leqslant 200\,000$, $0 \leqslant T \leqslant 10^8$) — количество точек и рассматриваемый промежуток времени.

Каждая из следующих n строк описывает точки. В i-й из них содержится пять целых чисел x_i, y_i, dx_i, dy_i и c_i ($-10^8 \leqslant x_i, y_i, dx_i, dy_i \leqslant 10^8$, $1 \leqslant c_i \leqslant n$) — координаты, скорость по каждой координате и цвет точки i.

Гарантируется, что существуют две точки с различными цветами.

Гарантируется, что сумма n по всем наборам входных данных в каждом тесте не превосходит $200\,000$, сумма T не превосходит 10^8.

Гарантируется, что любые координаты точек в любой момент времени до T включительно по модулю не превосходят 10^9.

Формат выходных данных

Для каждого набора входных данных в отдельной строке выведите одно число — минимум максимального расстояния между точками разных цветов за первые T секунд.

Ваш ответ будет считаться правильным, если его абсолютная или относительная ошибка не превосходит 10^{-9}.

Формально, пусть ваш ответ равен a, а ответ жюри равен b. Ваш ответ будет зачтен, если и только если $\frac{|a-b|}{\max(1,|b|)} \leqslant 10^{-9}$.

Пример

стандартный ввод	стандартный вывод
3 3 0 1 1 0 0 1 2 2 0 0 2 4 4 0 0 1 2 2 0 0 1 1 1 2 2 -1 -1 2 4 100 0 0 0 1 1 4 0 -1 0 2 4 4 0 -1 1 0 4 1 0 2	2.82842712474619009753 0.00000000000000000000 2.82842712474619009753

Замечание

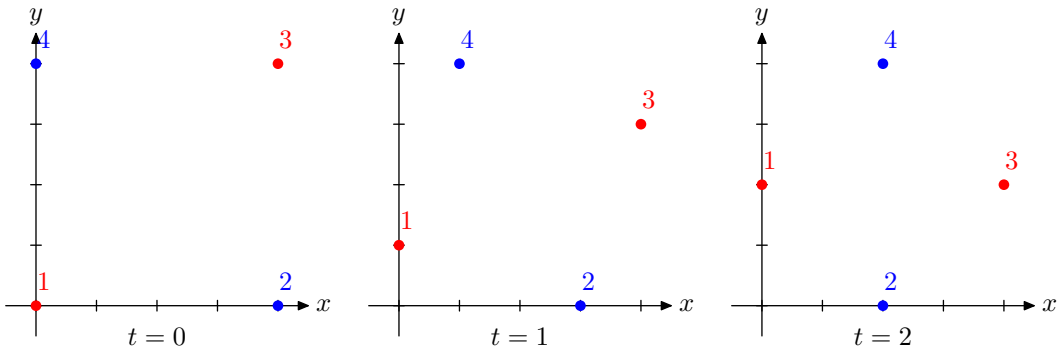
Рассмотрим пример из условия.

В первом наборе входных данных рассматривается только один момент времени. Расстояние между первой и второй точкой $\sqrt{2}$, между второй и третьей точкой $2\sqrt{2}$. Поэтому ответ — $2\sqrt{2}$.

Во втором наборе входных данных в момент времени 1 точки совпадут, расстояние между ними будет равняться нулю. Поэтому ответ равен 0.

В третьем наборе входных данных можно показать, что минимум максимального расстояния между точками различных цветов достигается в момент времени 2 и равняется $2\sqrt{2}$.

На рисунке изображен третий набор входных данных для моментов времени $t = 0, 1, 2$. Красным цветом отмечены точки цвета 1, синим — точки цвета 2.



Система оценки

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

Обозначим за $\sum n$ — сумму n по всем наборам входных данных данного теста.

Обозначим за $\sum T$ — сумму T по всем наборам входных данных данного теста.

Группа	Баллы	Доп. ограничения		Необх. группы	Комментарий
		$\sum n$	$\sum T$		
0	0	–	–	–	Тесты из условия.
1	10	$\sum n \leqslant 100$	$\sum T \leqslant 10$	0	
2	8	$\sum n \leqslant 10\,000$	$\sum T \leqslant 10$	0, 1	
3	11	$\sum n \leqslant 100$	–	0, 1	
4	14	$\sum n \leqslant 100\,000$	$\sum T \leqslant 10$	–	Различных цветов ровно 2.
5	8	$\sum n \leqslant 100\,000$	–	4	Различных цветов ровно 2.
6	11	$\sum n \leqslant 100\,000$	$\sum T \leqslant 10$	–	Все цвета различные.
7	8	$\sum n \leqslant 100\,000$	–	6	Все цвета различные.
8	19	$\sum n \leqslant 100\,000$	–	0 – 7	
9	11	–	–	0 – 8	**Offline-проверка.**