A. Контест для роботов

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Поликарп готовит первый контест по программированию для роботов. В этом контесте n задач, и большое количество самых разных роботов попробует их решить. Каждый робот, решивший задачу i, получает p_i баллов, и итоговый результат робота в соревновании считается как сумма p_i по всем задачам i, которые он решил. Для каждой задачи p_i — целое число не меньше 1.

Две компании, специализирующиеся на робототехнике, «Robo-Coder Inc.» и «BionicSolver Industries», также собираются отправить по одному роботу на соревнование. Поликарп знает все сильные и слабые стороны роботов, производимых этими компаниями, поэтому для каждой задачи он может точно предсказать, какие из этих двух роботов решат ее, а какие — не решат. Зная эту информацию, он может оценить результаты соревнования или повлиять на них.

По какой-то причине (которая совершенно точно никак не связана с подкупом) Поликарп хочет, чтобы робот «Robo-Coder Inc.» выступил лучше, чем робот «BionicSolver Industries». Поликарп хочет выставить баллы за задачи p_i таким образом, что робот «Robo-Coder Inc.» получит **строго больше** баллов, чем робот «BionicSolver Industries». Однако если значения p_i будут большими, то наблюдатели могут что-то заподозрить, поэтому Поликарп хочет минимизировать максимум p_i по всем задачам. Можете ли вы помочь Поликарпу определить минимально возможное верхнее ограничение на количество баллов за каждую задачу?

Входные данные

В первой строке задано одно целое число n ($1 \le n \le 100$) — количество задач в контесте.

Во второй строке заданы n целых чисел r_1, r_2, ..., r_n ($0 \le r_i \le 1$). $r_i=1$ оз-начает, что робот «Robo-Coder Inc.» решит i-ю задачу, $r_i = 0$ означает, что он не решит i-ю задачу.

В третьей строке заданы n целых чисел b_1, b_2, ..., b_n ($0 \le b_i \le 1$). $b_i=1$ означает, что робот «BionicSolver Industries» решит i-ю задачу, $b_i = 0$ означает, что он не решит i-ю задачу.

Выходные данные

Если роботу «Robo-Coder Inc.» ни при какой разбалловке не удастся набрать больше баллов, чем наберет робот «BionicSolver Industries», выведите одно число -1.

Иначе выведите минимально возможное значение $\max\limits_{i=1}^{n} p_i$, при котором можно расставить такие значения p_i, что робот «Robo-Coder Inc.» наберет **строго больше** баллов, чем робот «BionicSolver Industries».

Пример

входные данные
5 1 1 1 0 0 0 1 1 1 1
выходные данные
3

входные данные
3 0 0 0 0 0 0
выходные данные
-1

входные данные
4 1 1 1 1 1 1 1 1
выходные данные
-1

входные данные
8 1 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1
выходные данные
7

Примечание

В первом примере одна из возможных разбалловок — следующая: $p=[3,1,3,1,1]$. Тогда «Robo-Coder» получит 7 баллов, «BionicSolver» — 6 баллов.

Во втором примере оба робота получат 0 баллов, поэтому разбалловка ни на что не влияет.

В третьем примере оба робота решают все задачи, поэтому разбалловка также ни на что не влияет.

B. Планирование путешествия

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Таня запланировала путешествие по городам Берляндии. Всего в Берляндии n городов, расположенных вдоль главного железнодорожного пути. Города пронумерованы от 1 до n.

План путешествия Тани выглядит следующим образом: для начала она выбирает город c_1, с которого она начнет путешествие. Она посещает его, а затем отправляется в другой город $c_2 > c_1$, затем в следующий город $c_3 > c_2$, и так далее, пока не решит закончить свое путешествие в некотором городе $c_k > c_k-1$. Таким образом, последовательность посещенных городов $[c_1, c_2, \ldots, c_k]$ должна быть строго возрастающей.

На последовательность посещенных городов есть еще одно ограничение. Город i имеет красоту b_i. Если Таня решит посетить только один город, красоты городов не накладывают никаких дополнительных

ограничений; однако если она решила посетить несколько городов, то для каждой пары соседних городов c_i и c_{i+1} в плане должно выполняться условие $c_{i+1} - c_i = b_{Ci+1} - b_{Ci}$.

Например, если $n=8$ и $b=[3,4,4,6,6,7,8,9]$, следующие планы путешествия корректны:

- $c=[1,2,4]$;
- $c=[3,5,6,8]$;
- $c=[7]$ (путешествие, состоящее из одного города, считается корректным).

Также существуют другие планы посещения городов, не описанные выше.

Таня хочет, чтобы ее путешествие было максимально красивым. Красота путешествия равна суммарной красоте всех посещенных городов. Можете ли вы составить план путешествия, максимизирующий суммарную красоту посещенных городов?

Входные данные

Первая строка содержит целое число n ($1 \le n \le 2 \cdot 10^5$) — количество городов в Берляндии.

Вторая строка содержит n целых чисел $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 4 \cdot 10^5$), где b_i равно красоте i-го города.

Выходные данные

Выведите одно число — максимально возможную красоту путешествия Тани.

Пример

входные данные
6 10 7 1 9 10 15

выходные данные
26

входные данные
1 400000

выходные данные
400000

входные данные
7 8 9 26 11 12 29 14

выходные данные
55

Примечание

Оптимальный план путешествия в первом примере: $c=[2,4,5]$.

Оптимальный план путешествия во втором примере: $c=[1]$.

Оптимальный план путешествия в третьем примере: $c=[3,6]$.

C. Удаление соседних

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам задана строка s, состоящая из строчных букв латинского алфавита.

Пусть длина s равна $|s|$.

За один ход вы можете выбрать любой индекс i и **удалить** i-й символ s (s_i), если **хотя бы один** из его соседних символов является *предыдущей* буквой в латинском алфавите для s_i. Например, для буквы b *предыдущей* буквой является a, для буквы s *предыдущей* букой является r, для буквы a *предыдущей* буквы не существует. Заметьте, что после каждого удаления длина строки уменьшается на единицу. Таким образом, индекс i должен удовлетворять условию $1 \le i \le |s|$ в течение каждого хода.

Соседними символами для символа s_i являются символы s_{i-1} и s_{i+1}. Первый и последний символы s имеют только один соседний символ (за исключением случая $|s|=1$).

Рассмотрим следующий пример. Пусть $s=$ bacabcab.

1. В течение первого хода вы можете удалить первый символ $s_1 =$ b, так как $s_2=$a. Тогда строка станет равна $s=$acabcab.
2. В течение второго хода вы можете удалить пятый символ $s_5=$ c, так как $s_4=$b. Тогда строка станет равна $s=$acabab.
3. В течение третьего хода вы можете удалить шестой символ $s_6=$ b, так как $s_5=$ a. Тогда строка станет равна $s=$ acaba.
4. В течение четвертого хода единственным символом, который вы можете удалить, является $s_4=$ b, так как $s_3=$ a (или $s_5=$ a). Строка станет равна $s=$ acaa, и вы больше не сможете ничего с ней сделать.

Ваша задача — найти **максимальное** количество символов, которые вы можете удалить, если вы выберете последовательность ходов оптимально.

Входные данные

Первая строка входных данных содержит одно целое число $|s|$ ($1 \le |s| \le 100$) — длину s.

Вторая строка входных данных содержит строку s, состоящую из $|s|$ строчных букв латинского алфавита.

Выходные данные

Выведите одно целое число — максимально возможное количество символов, которые вы можете удалить, если вы выберете последовательность ходов оптимально.

Пример

входные данные
8 bacabcab
выходные данные
4

входные данные
4 bcda
выходные данные
3

входные данные
6 abbbbb
выходные данные
5

Примечание

Первый тестовый пример разобран в условии задачи. Заметьте, что последовательность ходов, представленная в условии, не является единственной, но можно показать, что максимально возможный ответ на этот тест равен 4.

Во втором тестовом примере вы можете удалить все символы s, кроме одного. Единственный возможный ответ описан ниже.

1. В течение первого хода следует удалить третий символ s_3= d, s станет равна bca.
2. В течение второго хода следует удалить второй символ s_2=c, s станет равна ba.
3. Наконец, в течение третьего хода следует удалить первый символ s_1= b, s станет равна a.

D. Система навигации

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 512 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Карта Бертауна может быть представлена как набор из n перекрестков, пронумерованных от 1 до n и соединенных между собой m односторонними дорогами. Гарантируется, что возможно добраться от любого перекрестка до любого другого, путешествуя по существующим дорогам. Длина какого-либо пути от одного перекрестка до другого равна количеству переходов по дорогам в этом пути. Кратчайшим путем от перекрестка v до другого перекрестка u называется путь, который начинается в v, заканчивается в u и имеет минимальную длину среди всех таких путей.

Поликарп живет около перекрестка s и работает в здании неподалеку от перекрестка t. Каждый день он ездит от s до t на машине. Сегодня он выбрал следующий путь до своей работы: p_1, p_2, ..., p_k, где $p_1 = s$, $p_k = t$, а все другие элементы этой последовательности являются промежуточными перекрестками, перечисленными в том порядке, в котором Поликарп их посещал. Поликарп никогда не посещал один и тот же перекресток дважды, таким образом, все элементы этой последовательности попарно различны. **Заметьте, что вы знаете путь Поликарпа заранее (он фиксирован), и он не обязательно является одним из кратчайших путей от s до t.**

Машина Поликарпа оснащена сложной навигационной системой. Опишем, как она работает. Когда Поликарп начинает свою поездку с перекрестка s, система выбирает какой-либо кратчайший путь от s до t и показывает его Поликарпу. Пусть следующий перекресток в выбранном пути равен v. Если Поликарп выбирает ехать дорогой от s до v, то навигатор показывает ему тот же самый кратчайший путь (очевидно, он будет стартовать с v с того момента, как Поликарп доедет до этого перекрестка). Однако если Поликарп выбирает доехать до другого перекрестка w, навигатор **перестраивает** путь: как только Поликарп доезжает до w, навигационная система выбирает какой-то кратчайший путь от w до t и показывает его Поликарпу. Тот же самый процесс продолжается до тех пор, пока Поликарп не доедет до t: если Поликарп едет по дороге, рекомендованной системой, то она оставляет кратчайший путь, который уже был построен; но если Поликарп выбирает какой-либо другой путь, система **перестраивает** путь по тем же самым правилам.

Рассмотрим пример. Пусть карта Бертауна выглядит следующим образом, а Поликарп едет путем $[1,2,3,4]$ ($s=1$, $t=4$):

Ознакомьтесь с картинкой по ссылке http://tk.codeforces.com/a.png

1. Когда Поликарп начинает свой путь у перекрестка 1, система выбирает какой-то кратчайший путь от 1 до 4. Существует только один такой путь, он равен $[1,5,4]$;
2. Поликарп выбирает поехать к перекрестку 2, который не лежит на кратчайшем пути, выбранным системой. Когда Поликарп доезжает до 2, навигатор **перестраивает** путь, выбирая какой-то кратчайший путь от 2 до 4, например, $[2,6,4]$ (заметьте, что он может выбрать $[2,3,4]$);
3. Поликарп выбирает поехать к перекрестку 3, который не лежит на кратчайшем пути, выбранным системой. Когда Поликарп доезжает до 3, навигатор **перестраивает** путь, выбирая единственный кратчайший путь от 3 до 4, который равен $[3,4]$;
4. Поликарп доезжает до 4 по дороге, выбранной навигатором, таким образом, система больше ничего не перестраивает.

В этом сценарии случилось 2 перестроения маршрута. Заметьте, что если система выберет $[2,3,4]$ вместо $[2,6,4]$ в течение второго шага, то будет только 1 перестроение (так как Поликарп едет по этому же пути, система сохранит путь $[3,4]$ в течение третьего шага).

Пример показывает, что количество перестроений может различаться даже несмотря на то, что карта Бертауна и путь, выбранный Поликарпом, никогда не меняются. Имея эту информацию (карту и путь Поликарпа), сможете ли вы определить минимальное и максимальное количество перестроений маршрута, которые могли произойти в течение поездки?

Входные данные

Первая строка содержит два целых числа n и m ($2 \le n \le m \le 2 \cdot 10^5$) — количество перекрестков и односторонних дорог в Бертауне соответственно.

Далее следуют m строк, каждая из которых описывает дорогу. Каждая строка содержит два целых числа u и v ($1 \le u, v \le n$, $u \ne v$), обозначающих дорогу от перекрестка u к перекрестку v. Все дороги в Бертауне попарно различны, и это значит, что каждая упорядоченная пара (u,v) встречается не более одного раза среди этих m строк (но если существует дорога (u,v), то дорога (v,u) также может существовать).

Следующая строка содержит одно целое число k ($2 \le k \le n$) — количество перекрестков в пути Поликарпа от его дома до его работы.

Последняя строка содержит k целых чисел p_1, p_2, ..., p_k ($1 \le p_i \le n$, все эти целые числа попарно различны) — перекрестки в пути Поликарпа в том порядке, в котором он их посещал. p_1 равно перекрестку, около которого живет Поликарп ($s = p_1$), а p_k равно перекрестку, около которого Поликарп работает ($t = p_k$). Гарантируется, что для каждого $i \in [1, k-1]$ существует дорога от p_i до p_{i+1}, таким образом, путь идет по дорогам Бертауна.

Выходные данные

Выведите два числа: минимальное и максимальное количество **перестроений маршрута**, которые могут произойти в течение поездки.

Пример

входные данные
6 9 1 5

```
5 4
1 2
2 3
3 4
4 1
2 6
6 4
4 2
4
1 2 3 4
```

выходные данные

```
1 2
```

входные данные

```
7 7
1 2
2 3
3 4
4 5
5 6
6 7
7 1
7
1 2 3 4 5 6 7
```

выходные данные

```
0 0
```

входные данные

```
8 13
8 7
8 6
7 5
7 4
6 5
6 4
5 3
5 2
4 3
4 2
3 1
2 1
1 8
5
8 7 5 2 1
```

выходные данные

```
0 3
```

E. World of Darkraft: Battle for Azathoth

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 512 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Рома играет в новое дополнение к его любимой игре — World of Darkraft. Рома только что закончил создавать персонажа и готов выйти на первую охоту на монстров.

Перед охотой Рома должен купить оружие и комплект брони. Он может выбрать **ровно одно** из n различных типов оружия и **ровно один** из m различных комплектов брони. Оружие i имеет модификатор атаки a_i и стоит ca_i монет, комплект брони j имеет модификатор защиты b_j и стоит cb_j монет.

После выбора экипировки Рома может начать охотиться на монстров. Всего есть p монстров, которых можно попытаться победить. У каждого монстра k есть три параметра: защита x_k, атака y_k и награда за его

убийство z_k. Рома может победить монстра только в том случае, если модификатор атаки его оружия больше, чем защита монстра, а модификатор защиты его брони больше, чем атака монстра. То есть монстра k можно победить с использованием оружия i и комплекта брони j, если $a_i > x_k$ и $b_j > y_k$. После победы над монстром Рома получает награду за него. Во время охоты можно попробовать победить любое количество монстров в любом порядке — но они не появляются заново после смерти, поэтому каждого монстра можно победить только один раз.

Благодаря внесенным в игру деньгам Рома может себе позволить любое оружие и любой комплект брони. Несмотря на это, ему все равно необходимо распланировать охоту так, чтобы получить максимальную прибыль. Прибыль с охоты считается как разность между количеством монет, полученных за убийство монстров, и ценой выбранного оружия и брони. Обратите внимание, что Рома **должен** купить оружие и броню, даже если охота не покроет расходы на них.

Помогите Роме максимизировать прибыль с охоты на монстров.

Входные данные

В первой строке заданы три целых числа n, m и p ($1 \le n, m, p \le 2 \cdot 10^5$) — количество доступных типов оружия, комплектов брони и монстров соответственно.

В следующих n строках описаны доступные типы оружия. В i-й из этих строк заданы два целых числа a_i и ca_i ($1 \le a_i \le 10^6$, $1 \le ca_i \le 10^9$) — модификатор атаки и цена оружия i.

В следующих m строках описаны доступные комплекты брони. В j-й из этих строк заданы два целых числа b_j и cb_j ($1 \le b_j \le 10^6$, $1 \le cb_j \le 10^9$) — модификатор защиты и цена комплекта брони j.

В следующих p строках описываются монстры. В k-й из этих строк заданы три целых числа x_k, y_k, z_k ($1 \le x_k, y_k \le 10^6$, $1 \le z_k \le 10^3$) — защита, атака и награда за убийство монстра k.

Выходные данные

Выведите одно целое число — максимальную прибыль с охоты.

Пример

входные данные
2 3 3 2 3 4 7 2 4 3 2 5 11 1 2 4 2 1 6 3 4 6
выходные данные
1

F. Древляндия и вирусы

ограничение по времени на тест: 3 секунды
ограничение по памяти на тест: 512 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

В Древляндии n городов, соединенных $n-1$ двунаправленными дорогами таким образом, что из любого города можно добраться в любой другой (другими словами, граф городов и дорог является деревом). Древлядния готовится к сезонной вирусной эпидемии, и сейчас они пытаются оценить различные сценарии распространения инфекции.

В каждом из сценариев некоторые города исходно заражены различными видами вирусов. Пусть в i-м сценарии присутствует k_i типов вирусов. Обозначим за v_j исходный город для вируса j, и за s_j скорость распространения вируса j. Распространение вирусов происходит пошагово: сперва распространяется вирус 1, затем вирус 2, и так далее. После распространения вируса k_i процесс начинается заново с вируса 1.

Шаг распространения вируса j происходит следующим образом. Для каждого города x, еще не зараженного никаким вирусом в начале шага, заражение вирусом j происходит тогда и только тогда, когда найдется другой город y со следующими свойствами:

- город y заражен вирусом j в начале шага;
- путь между городами x и y содержит не более s_j рёбер;
- ни один город на пути между x и y (не считая y) не был заражен никаким вирусом в начале шага.

Если город заражен вирусом, он всегда остается зараженным и больше не может быть заражен никаким другим вирусом. Распространение вирусов останавливается, когда все города заражены.

Вам требуется обработать q независимых сценариев. Каждый сценарий описывается k_i типами вирусов и списком из m_i важных городов. Для каждого важного города определите тип вируса, которым он будет заражен в конце сценария.

Входные данные

Первая строка содержит одно целое число n ($1 \le n \le 2 \cdot 10^5$) — количество городов в Древляндии.

Следующие $n-1$ строк описывают дороги. i-я из этих строк содержит два целых числа x_i и y_i ($1 \le x_i, y_i \le n$) — номера городов, соединенных i-й дорогой. Гарантируется, что данный граф городов и дорог является деревом.

Следующая строка содержит одно целое число q ($1 \le q \le 2 \cdot 10^5$) — количество сценариев заражения. Далее следуют q описаний сценариев.

Описание i-го сценария начинается со строки, содержащей два целых числа k_i и m_i ($1 \le k_i, m_i \le n$) — количество типов вируса и количество важных городов в этом сценарии соответственно. Гарантируется, что $\sum_{i=1}^{q} k_i$ и $\sum_{i=1}^{q} m_i$ не превосходят $2 \cdot 10^5$.

Следующие k_i строк описывают типы вирусов. j-я из этих строк содержит два целых числа v_j и s_j ($1 \le v_j \le n$, $1 \le s_j \le 10^6$) – номер исходного города и скорость распространения вируса j. Гарантируется, что исходные города всех типов вирусов внутри одного сценария различны.

Следующая строка содержит m_i различных чисел $u_1, \ldots, u_{m_i}$ ($1 \le u_j \le n$) — номера важных городов .

Выходные данные

Выведите q строк. i-я строка должна содержать m_i целых чисел — номера типов вирусов, которыми будут заражены города $u_1, \ldots, u_{m_i}$ соответственно в конце i-го сценария.

Примеры

входные данные
7 1 2

```
1 3
2 4
2 5
3 6
3 7
3
2 2
4 1
7 1
1 3
2 2
4 3
7 1
1 3
3 3
1 1
4 100
7 100
1 2 3
```

выходные данные

```
1 2
1 1
1 1 1
```

G. Достижимые строки

ограничение по времени на тест: 3 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

В этой задаче мы будем иметь дело с бинарными строками. Каждый символ бинарной строки — это либо `0`, либо `1`. Мы также будем иметь дело с подстроками; напомним, что подстрока — это непрерывная подпоследовательность строки. Для удобства будем обозначать подстроку строки s, начинающуюся в позиции l и заканчивающуюся в позиции r, как $s[l...r]$. Символы в строках нумернуются с 1.

Мы можем проводить операции над строками. Есть две различных операции: заменить подстроку `011` на `110`, или заменить подстроку `110` на `011`. Например, если мы применим ровно одну операцию к строке `110011110`, она может превратиться в `011011110`, `110110110`, или `110011011`.

Бинарная строка a считается *достижимой* из бинарной строки b, если существует последовательность s_1, s_2, ..., s_k, такая, что $s_1=a$, $s_k=b$, и для каждого $i\in[1,k-1]$, s_i можно превратить в s_{i+1} ровно за одну операцию. Обратите внимание, что k может быть равно 1, т. е., **каждая строка достижима из самой себя**.

Вам дана строка t и q запросов к ней. Каждый запрос состоит из трех целых чисел l_1, l_2 и l_{en}. Для каждого запроса вы должны определить, является ли $t[l_1...l_1+l_{en}-1]$ достижимой из $t[l_2...l_2+l_{en}-1]$.

Входные данные

В первой строке содержится одно целое число n ($1\le n\le 2\cdot10^5$) — длина строки t.

Во второй строке задана одна строка t ($|t|=n$). Каждый символ t равен либо 0, либо 1.

В третьей строке задано одно целое число q ($1\le q\le 2\cdot10^5$) — количество запросов.

Затем следуют q строк, в каждой из которых содержится описание очередного запроса. В i-й строке заданы три целых числа l_1, l_2 и len ($1 \le l_1, l_2 \le |t|$, $1 \le len \le |t| - max(l_1, l_2) + 1$) для i-го запроса.

Выходные данные

Для каждого запроса выведите либо `YES`, если $t[l_1 \dots l_1 + len - 1]$ достижима из $t[l_2 \dots l_2 + len - 1]$, либо `NO` в противном случае. Вы можете выводить каждую букву в любом регистре.

Примеры

входные данные
5 11011 3 1 3 3 1 4 2 1 2 3
выходные данные
Yes Yes No

H. Блоки и сенсоры

ограничение по времени на тест: 3 секунды
ограничение по памяти на тест: 512 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Поликарп играет в одну очень известную компьютерную игру (мы не будем упоминать ее имя). Каждый объект в этой игре состоит из трехмерных блоков — кубов $1 \times 1 \times 1$, ребра которых параллельны осям координат. Эти блоки не подвержены гравитации, поэтому могут находиться в воздухе без всякой опоры. Блоки располагаются в ячейках $1 \times 1 \times 1$; каждая ячейка либо пуста, либо содержит ровно один блок. Каждая ячейка обозначается своими координатами (x, y, z) (ячейка с такими координатами — это куб с противоположными вершинами в (x, y, z) и $(x+1, y+1, z+1)$ и значением $a_{x,y,z}$; если ячейка пуста, $a_{x,y,z} = 0$, иначе $a_{x,y,z}$ равно типу блока в ячейке (типы блоков — целые числа от 1 до $2 \cdot 10^5$).

Поликарп построил большую структуру из блоков. Эта структура находится внутри прямоугольного параллелепипеда $n \times m \times k$, содержащего все такие клетки (x, y, z), что $x \in [1, n]$, $y \in [1, m]$ и $z \in [1, k]$. После построения структуры Поликарп окружил параллелепипед $2nm + 2nk + 2mk$ сенсорами. Сенсор — специальный блок, который выпускает луч в некотором направлении и показывает тип первого блока, до которого дошел луч (за исключением других сенсоров). Сенсоры, поставленные Поликарпом, расположены вплотную к границам параллелепипеда; луч каждого сенсора параллелен одной из координатных осей и направлен внутрь параллелепипеда. Более формально, сенсоры можно разделить на 6 типов:

- есть mk сенсоров первого типа; каждый такой сенсор расположен в $(0, y, z)$, где $y \in [1, m]$ и $z \in [1, k]$, и выпускает луч в положительном направлении оси Ox;
- есть mk сенсоров второго типа; каждый такой сенсор расположен в $(n+1, y, z)$, где $y \in [1, m]$ и $z \in [1, k]$, и выпускает луч в отрицательном направлении оси Ox;

- есть nk сенсоров третьего типа; каждый такой сенсор расположен в $(x,0,z)$, где $x \in [1,n]$ и $z \in [1,k]$, и выпускает луч в положительном направлении оси Oy;
- есть nknk сенсоров четвертого типа; каждый такой сенсор расположен в $(x,m+1,z)$, где $x \in [1,n]$ и $z \in [1,k]$, и выпускает луч в отрицательном направлении оси Oy;
- есть nm сенсоров пятого типа; каждый такой сенсор расположен в $(x,y,0)$, где $x \in [1,n]$ и $y \in [1,m]$, и выпускает луч в положительном направлении оси Oz;
- наконец, есть nm сенсоров шестого типа; каждый такой сенсор расположен в $(x,y,k+1)$, где $x \in [1,n]$ и $y \in [1,m]$, и выпускает луч в отрицательном направлении оси Oz.

Поликарп пригласил своего друга Монокарпа поиграть с ним. Естественно, как только Монокарп увидел огромный параллелепипед, окруженный сенсорами, ему стало интересно, что находится внутри. Поликарп решил не сообщать Монокарпу точную форму структуры; вместо этого он предложил Монокарпу догадаться, какая структура построена внутри, по показаниям сенсоров.

После нескольких часов безуспешных попыток угадать, что же находится внутри, Монокарп все еще не хочет сдаваться. Вместо этого он решил попросить помощи. Можете ли вы написать программу, которая проанализирует показания сенсоров и построит любую структуру, которая не противоречит показаниям?

Входные данные

В первой строке заданы три целых числа n, m и k ($1 \le n,m,k \le 2 \cdot 10^5$, $nmk \le 2 \cdot 10^5$) — размеры параллелепипеда.

Затем следуют показания сенсоров. Показания каждого сенсора — это либо 0, если луч, выпущенный этим сенсором, достиг противоположного сенсора (между ними нет блоков), либо целое число от 1 до $2 \cdot 10^5$, соответствующее типу первого блока, которого достиг луч. Все показания разделены на 6 секций (по одной на каждый тип сенсоров), каждая пара соседних секций разделяется пустой строкой, а также первая секция отделяется от размеров параллелепипеда пустой строкой.

Первая секция состоит из m строк по k целых чисел. j-е число в i-й строке соответствует показаниям сенсора, расположенного в $(0,i,j)$.

Вторая секция состоит из m строк по k целых чисел. j-е число в i-й строке соответствует показаниям сенсора, расположенного в $(n+1,i,j)$.

Третья секция состоит из n строк по k целых чисел. j-е число в i-й строке соответствует показаниям сенсора, расположенного в $(i,0,j)$.

Четвертая секция состоит из n строк по k целых чисел. j-е число в i-й строке соответствует показаниям сенсора, расположенного в $(i,m+1,j)$.

Пятая секция состоит из n строк по m целых чисел. j-е число в i-й строке соответствует показаниям сенсора, расположенного в $(i,j,0)$.

Наконец, шестая секция состоит из n строк по m целых чисел. j-е число в i-й строке соответствует показаниям сенсора, расположенного в $(i,j,k+1)$.

Выходные данные

Если показания сенсоров противоречивы, выведите -1.

Иначе выведите описание структуры внутри параллелепипеда следующим образом. Выходные данные должны содержать nmk целых чисел: $a_{1,1,1}$, $a_{1,1,2}$, ..., $a_{1,1,k}$, $a_{1,2,1}$, ..., $a_{1,2,k}$, ..., $a_{1,m,k}$, $a_{2,1,1}$, ..., $a_{n,m,k}$, где $a_{i,j,k}$ — тип блока в (i,j,k), или 0, если эта ячейка пуста. Если существует несколько структур, которым соответствуют показания сенсоров, выведите любую из них.

Для вашего удобства выходные данные в примерах отформатированы: они разделены на nm секций для блоков с $x=1$, $x=2$, ..., $x=n$; каждая секция состоит из m строк по k чисел. **Можно выводить ответ в таком виде, но любой другой формат вывода тоже будет принят; можно, например, вывести все числа на одной строке, важен только их порядок.**

Примеры

входные данные
4 3 2 1 4 3 2 6 5 1 4 3 2 6 7 1 4 1 4 6 5 1 4 6 5 1 3 6 1 3 6 4 3 5 4 2 5

выходные данные
1 4 3 0 6 5 1 4 3 2 6 5 0 0 0 0 0 0 0 0 0 0 0 7

входные данные
1 1 1 0 0 0 0 0 0
выходные данные
0

входные данные
1 1 1 0 0 1337 0 0 0
выходные данные
-1

входные данные
1 1 1 1337 1337 1337 1337 1337 1337
выходные данные
1337