

Задача А. Биномиальная куча

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Реализуйте биномиальную кучу.

Формат входных данных

В первой строке содержится два целых числа: N — общее количество куч и M — количество операций ($1 \leq N \leq 1000, 1 \leq M \leq 1\,000\,000$). Изначально все кучи пусты.

Требуется поддерживать следующие операции:

- $0\ a\ v$ — добавить элемент со значением v в кучу с номером a . Вновь добавленный элемент имеет уникальный индекс равный порядковому номеру соответствующей операции добавления. Нумерация начинается с единицы.
- $1\ a\ b$ — переложить все элементы из кучи с номером a в кучу с номером b . После этой операции куча a становится пустой.
- $2\ i$ — удалить элемент с индексом i .
- $3\ i\ v$ — присвоить элементу с индексом i значение v . Гарантируется, что элемент существует.
- $4\ a$ — вывести на отдельной строке значение минимального элемента в куче с номером a . Гарантируется, что куча не пуста.
- $5\ a$ — удалить минимальный элемент из кучи с номером a . Если таковых несколько, то выбирается элемент с минимальным индексом. Гарантируется, что куча не пуста.

Формат выходных данных

Для каждой операции поиска минимального элемента выведите единственное число: значение искомого элемента.

Пример

стандартный ввод	стандартный вывод
3 19	10
0 1 10	5
4 1	7
0 2 5	7
0 2 7	10
4 2	3
3 2 20	10
4 2	8
1 2 1	
4 1	
5 1	
4 1	
3 2 3	
4 1	
2 2	
4 1	
0 1 9	
1 1 3	
0 3 8	
4 3	

Задача В. Персистентная приоритетная очередь

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мегабайт

Требуется реализовать структуру данных, которая хранит мультимножество и умеет изменять любую свою предыдущую версию, выполняя одну из этих операций:

1. Заданы v и x , требуется добавить в множество v элемент со значением x , после чего вывести минимальный элемент в получившемся множестве.
2. Заданы v и u , требуется объединить множества с номерами v и u , после чего вывести минимальный элемент в получившемся множестве.
3. Задано v , требуется вывести минимальный элемент в множестве v , после чего удалить минимальный элемент из множества v . Если множество пустое, то вывести, что множество пустое, и создать новое пустое множество.

Изначально есть одно пустое множество с номером 0. После операции с номером i множество, получаемое во время этой операции, получает номер i .

Формат входных данных

Первая строка содержит число n — количество операций для выполнения.

От вас потребуется отвечать на запросы в онлайн, при этом поддерживая переменную s . Она изначально равна нулю. После каждой операции, она пересчитывается следующим образом через предыдущее значение: если ответ на запрос равен x , то $s = (s_{old} + x) \bmod 239017$. Если же ответом на запрос является слово `empty`, то s не изменяется.

В следующих n строках заданы запросы.

Запросы первого типа описываются строкой **1 a b**, где a и b — неотрицательные целые числа, которые описывают v и x для соответствующего запроса, как $v = (a + s) \bmod i$ и $x = (b + 17s) \bmod (10^9 + 1)$, где i — номер соответствующего запроса.

Запросы второго типа описываются строкой **2 a b**, где a и b — неотрицательные целые числа, которые описывают v и u для соответствующего запроса, как $v = (a + s) \bmod i$ и $u = (b + 13s) \bmod i$, где i — номер соответствующего запроса.

Запросы третьего типа описываются строкой **3 a**, где a — неотрицательное целое число, которые описывает v для соответствующего запроса, как $v = (a + s) \bmod i$, где i — номер соответствующего запроса.

Число запросов не превышает 200 000. Гарантируется, что мощность любого созданного мультимножества не превышает 2^{63} .

Формат выходных данных

Требуется вывести ровно n строк, в каждой строке должно находиться неотрицательное целое число либо слово `empty`.

Для запросов первого и второго типа требуется вывести значение минимального элемента в только что созданном множестве, либо слово `empty`, если множество пустое.

Для запросов третьего типа требуется вывести минимальный элемент в множестве, либо слово `empty`, если множество пустое.

Пример

стандартный ввод	стандартный вывод
9	2
1 0 2	3
1 0 999999970	2
2 2 0	2
3 0	2
2 4 4	2
3 0	2
3 0	3
3 0	empty
3 8	

Задача С. Отсортируй отрезки

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 5 секунд
Ограничение по памяти: 512 мегабайт

Дан массив a , состоящий из n целых чисел. Поступает q запросов следующего вида:

- В ответ на i -й запрос решение должно выводить «YES», если отрезок $[l_i, r_i]$ массива отсортирован и «NO» в ином случае.
- После ответа на i -й запрос требуется отсортировать отрезок с заданными границами.

От участника требуется быстро обработать все запросы.

Формат входных данных

В первой строке содержится пара целых чисел n, q ($1 \leq n \leq 3 \cdot 10^5, 1 \leq q \leq 3 \cdot 10^5$) — количество элементов массива и количество запросов.

Следующая строка содержит n целых чисел a_i ($1 \leq a_i \leq 10^9$) — элементы массива.

Следующие q строк содержат пары целых чисел l_i, r_i ($1 \leq l_i \leq r_i \leq n$) — описание запросов.

Формат выходных данных

Выходной файл должен содержать q строк, в каждой из которых записано «NO» или «YES» в зависимости от ответа на i -й запрос.

Примеры

стандартный ввод	стандартный вывод
5 3 5 2 3 4 1 1 3 3 4 1 4	NO NO YES
7 4 3 6 2 1 2 5 7 2 5 1 4 5 7 3 6	NO NO NO YES

Задача D. Фибоначчиева куча

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	1024 мегабайта

Эта задача для тех, кто завалил регоин (или его дисквалифицировали), отчаялся в олимпиадной проге и решил полностью посвятить себя теоретической информатике и продвинутым алгоритмам. Вам предстоит написать самую настоящую Фибоначчиеву кучу. Все запросы будут задаваться массивом a_i , где a_1 вам дано изначально, а для $i > 1$ $a_i = (a_{i-1} * b + c) \bmod 2^{32}$.

К вам будут поступать запросы двух типов:

1. В случае, если a_i чётно, то в i -м запросе от вас будет требоваться добавить число a_i в кучу.
2. В случае, если a_i нечётно от вас будет требоваться узнать значение верхнего (максимального) элемента в куче.

Формат входных данных

В первой строке вам дано единственное число n ($1 \leq n \leq 10^8$). В следующей строке вам дано 3 числа a_1 b и c ($0 \leq a_1, b, c \leq 2^{32}$). Гарантируется, что a_1 чётно.

Формат выходных данных

Для всех запросов второго типа выведите сумму ответов на них.

Пример

стандартный ввод	стандартный вывод
10 2 2 1	18

Замечание

Так как Фибоначчиева куча сложная и её код трудно читать, в этой задаче не будет ревью кода.

Задача E. К кратчайших путей

Имя входного файла: стандартный ввод
 Имя выходного файла: стандартный вывод
 Ограничение по времени: 1 секунда
 Ограничение по памяти: 256 мегабайт

Дан ориентированный взвешенный граф из n вершин и m рёбер. В нем зафиксированы вершины $s = 0$ и $t = n - 1$. Найдите длины k кратчайших путей из s в t . Пути выбираются среди множества всех путей, не обязательно простых. В графе могут присутствовать кратные рёбра, но петель быть не может.

Необходимо для всех i от 1 до k найти длину i -го кратчайшего пути из s в t .

Формат входных данных

В первой строке заданы числа n , m и k — количество вершин, ребер и необходимое количество путей соответственно ($2 \leq n \leq 50\,000$, $1 \leq m \leq 50\,000$, $1 \leq k \leq 50\,000$). Следующие m строк содержат описание рёбер в формате a_i, b_i, w_i . Это означает, что i -е ребро соединяет вершины a_i и b_i , и при этом имеет вес w_i ($1 \leq w_i \leq 100\,000$). Вершины нумеруются с нуля.

Формат выходных данных

Необходимо вывести k целых чисел — длины путей в порядке возрастания. Если очередного пути не существует, выведите вместо длины -1 .

Примеры

стандартный ввод	стандартный вывод
2 2 4 0 1 2 0 1 3	2 3 -1 -1
2 3 6 0 1 2 0 1 3 1 0 1	2 3 5 6 6 7
5 7 10 0 1 1 1 2 2 2 0 2 2 4 3 2 3 1 3 4 1 4 3 1	5 6 7 8 9 10 10 11 11 12

Задача F. Чапаев на дереве

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	512 мегабайт

Вова и Марина любят играть в игры, а особенно — придумывать к ним свои правила. Недавно они открыли для себя веселую игру «Чапаев», в которой игроки должны сбивать щелчками шашки вражеского цвета с шахматной доски (также эта игра известна под названием «Щелкунчики»). Вдоволь наигравшись, они решили модифицировать правила, добавив игре математическую сложность.

Теперь они играют в «Чапаева» не на шахматной доске, а на доске в форме дерева! Их дерево состоит из N вершин. Вершина 1 является корнем дерева, а из каждой из оставшихся вершин проведено ребро в некоторую вершину с меньшим номером — её непосредственного предка.

В игре участвуют шашки одного цвета, изначально расположенные в некоторых вершинах дерева. За один ход игрок выбирает некоторую шашку и щелчком отправляет ее к корню по ребрам дерева, сбивая при этом с доски все встреченные на пути шашки. Сама шашка, по которой производился удар, после попадания в корень дерева также слетает с доски.

Игроки делают ходы по очереди. Проигрывает тот игрок, к ходу которого на доске не остается шашек.

Придуманная ими игра замечательна также тем, что на одной и той же доске можно играть, начиная с разных начальных позиций шашек. Практика показала, что самые интересные партии получаются, если исходно расставить фишки во все вершины, являющиеся потомками (непосредственными или косвенными) некоторой вершины $Root$, при этом в саму вершину $Root$ фишка не ставится.

Дети решили сыграть N партий, перебрав в качестве вершины $Root$ каждую вершину дерева по одному разу. Если у очередной вершины $Root$ нет потомков, и на доске исходно не оказывается ни одной фишки, то игры не происходит, и дети переходят к следующей расстановке. В каждой партии Марина ходит первой.

Вова интересуется у вас, в скольких партиях Марина сможет одержать победу, если игроки будут действовать оптимально.

Формат входных данных

В первой строке находится целое число N ($1 \leq N \leq 500\,000$) — количество вершин в дереве.

Во второй строке следуют целые числа $p_2, p_3, \dots, p_N$, разделенные пробелами, где p_i — это номер вершины, являющейся предком вершины i ($1 \leq p_i \leq i$).

Формат выходных данных

Выведите единственное целое число — количество партий, в которых Марина одержит победу.

Пример

стандартный ввод	стандартный вывод
7 1 2 3 1 5 5	3

Замечание

Разберем тест из условия. Доска для игры показана на рисунках ниже. Дети сыграют четыре партии, выбирая в качестве $Root$ вершины 1, 2, 3 и 5. Если выбрать в качестве $Root$ любую из трех оставшихся вершин, на доске исходно не окажется ни одной фишки, поэтому игры не произойдет.

Если выбрать в качестве $Root$ вершину 5, фишки будут исходно находиться в вершинах 6 и 7. В такой партии Марина проигрывает: после того, как она сбивает любую из этих двух фишек с доски, Вова сбивает оставшуюся и заканчивает партию.

Если выбрать в качестве $Root$ вершину 2 или 3, у Марины будет возможность выиграть игру за один ход, щелкнув по фишке из вершины 4 (при этом, в случае $Root = 2$, она по пути также собьет фишку из 3 вершины по правилам игры)

Можно убедиться, что если выбрать в качестве *Root* вершину 1, у Марины также будет выигрышная стратегия. Для этого первым ходом Марина должна сбить фишку из вершины 2. Пример партии с таким начальным расположением показан ниже.

Таким образом, Марина выигрывает в трех партиях