A. Очередь с приоритетами

Ограничение времени	1 секунда
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Первая строка входа содержит число операций $1 \le n \le 10^5$. Каждая из последующих n строк задают операцию одного из следующих двух типов:

- `Insert x`, где $0 \le x \le 10^9$ — целое число;
- `ExtractMax`.

Первая операция добавляет число x в очередь с приоритетами, вторая — извлекает максимальное число и выводит его.

Задача из курса «Алгоритмы: теория и практика. Методы»: https://stepik.org/course/217/syllabus

Пример 1

Ввод	Вывод
7 Insert 10 Insert 200 ExtractMax ExtractMax Insert 5 Insert 500 ExtractMax	200 10 500

Пример 2

Ввод	Вывод
8 Insert 2 Insert 2	3 2 2

Ввод	Вывод
Insert 1 Insert 3 ExtractMax ExtractMax ExtractMax ExtractMax	1

B. Параллельная обработка

Ограничение времени	1 секунда
Ограничение памяти	256.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

В данной задаче ваша цель - реализовать симуляцию параллельной обработки списка задач. Такие обработчики (диспетчеры) есть во всех операционных системах.
У вас имеется n процессоров и последовательность из m задач. Для каждой задачи дано время, необходимое на её обработку. Очередная работа поступает к первому доступному процессору (то есть если доступных процессоров несколько, то доступный процессор с минимальным номером получает эту работу).
Задача из курса "Алгоритмы: теория и практика. Структуры данных": https://stepik.org/course/1547/syllabus

Формат ввода

Первая строка входа содержит числа n и m. Вторая содержит числа $t_0,\ldots,t_{m-1}$, где t_i --- время, необходимое на обработку i-й задачи ($1 \le n \le 10^5$; $1 \le m \le 10^5$; $0 \le t_i \le 10^9$). Считаем, что и процессоры, и задачи нумеруются с нуля.

Формат вывода

Выход должен содержать ровно m строк: i-я (считая с нуля) строка должна содержать номер процессора, который получит i-ю задачу на обработку, и время, когда это произойдёт.

Пример 1

Ввод	**Вывод**
2 5 1 2 3 4 5	0 0 1 0 0 1 1 2 0 4

Пример 2

Ввод	**Вывод**
2 6 1 1 1 1 1 1	0 0 1 0 0 1 1 1 0 2 1 2

Пример 3

Ввод	**Вывод**
2 8 1 2 1 2 1 2 1 2	0 0 1 0 0 1 0 2 1 2 1 3 0 4 0 5

C. Построение кучи

Ограничение времени	1 секунда
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Построение кучи — ключевой шаг алгоритма сортировки кучей. Данный алгоритм имеет время работы $O(n \log n)$ в худшем случае в отличие от алгоритма быстрой сортировки, который гарантирует такую оценку только в среднем случае. Алгоритм быстрой сортировки чаще используют на практике, поскольку в большинстве случаев он работает быстрее, но алгоритм сортировки кучей используется для внешней сортировки данных, когда необходимо отсортировать данные огромного размера, не помещающиеся в память компьютера.
Чтобы превратить данный массив в кучу, необходимо произвести несколько обменов его элементов. Обменом мы называем базовую операцию, которая меняет местами элементы $A[i]$ и $A[j]$. Ваша цель в данной задаче — преобразовать заданный массив в кучу за линейное количество обменов.
Задача из курса «Алгоритмы: теория и практика. Структуры данных»: https://stepik.org/course/1547/syllabus

Формат ввода

Первая строка содержит число n. Следующая строка задаёт массив чисел $A[0], \ldots, A[n - 1]$ ($1 \le n \le 10^5$; $0 \le A[i] \le 10^9$ для всех $0 \le i \le n - 1$; все $A[i]$ попарно различны; $i \ne j$).

Формат вывода

Первая строка выхода должна содержать число обменов m, которое должно удовлетворять неравенству $0 \le m \le 4n$. Каждая из последующих m строк должна задавать обмен двух элементов массива A. Каждый обмен задаётся парой различных индексов $0 \le i \ne j \le n - 1$, причем выполнено одно из равенств $j = 2i + 1$, $j = 2i + 2$, $i = 2j + 1$ или $i = 2j + 2$.
После применения всех обменов в указанном порядке массив должен превратиться в мин-кучу, то есть для всех $0 \le i \le n - 1$ должны выполняться следующие два условия:

- если $2i + 1 \le n - 1$, то $A[i] < A[2i + 1]$.
- если $2i + 2 \le n - 1$, то $A[i] < A[2i + 2]$.

Пример 1

Ввод	Вывод
6 0 1 2 3 4 5	0

Пример 2

Ввод	Вывод
6 7 6 5 4 3 2	4 2 5 1 4 0 2 2 5

D. Объединение таблиц

Ограничение времени	1 секунда
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Ваша цель в данной задаче — реализовать симуляцию объединения таблиц в базе данных. В базе данных есть n таблиц, пронумерованных от 1 до n, над одним и тем же множеством столбцов (атрибутов). Каждая таблица содержит либо реальные записи в таблице, либо символьную ссылку на другую таблицу. Изначально все таблицы содержат реальные записи, и i-я таблица содержит r_i записей. Ваша цель — обработать m запросов типа $(\text{destination}_i, \text{source}_i)$:

1. Рассмотрим таблицу с номером destination_i. Пройдясь по цепочке символьных ссылок, найдём номер реальной таблицы, на которую ссылается эта таблица:
   пока таблица destination_i *содержит символическую ссылку:* $\text{destination}_i \leftarrow \text{symlink}(\text{destination}_i)$
2. Сделаем то же самое с таблицей source_i.
3. Теперь таблицы destination_i и source_i содержат реальные записи.
   Если $\text{destination}_i \neq \text{source}_i$, скопируем все записи из таблицы source_i в таблицу destination_i, очистим таблицу source_i и пропишем в неё символическую ссылку на таблицу destination_i.
4. Выведем максимальный размер среди всех n таблиц. Размером таблицы называется число строк в ней. Если таблица содержит символическую ссылку, считаем её размер равным нулю.

Задача из курса «Алгоритмы: теория и практика. Структуры данных»: https://stepik.org/course/1547/syllabus

Формат ввода

Первая строка содержит числа n и m — число таблиц и число запросов, соответственно. Вторая строка содержит n целых чисел $r_1,\ldots,r_n$ — размеры таблиц. Каждая из последующих m строк содержит два номера таблиц $destination_i$ и $source_i$, которые необходимо объединить ($1 \le n,m \le 100000$; $0 \le r_i \le 10000$; $1 \le destination_i, source_i \le n$).

Формат вывода

Для каждого из m запросов выведите максимальный размер таблицы после соответствующего объединения.

Пример 1

Ввод	Вывод
5 5 1 1 1 1 1 3 5 2 4 1 4 5 4 5 3	2 2 3 5 5

Пример 2

Ввод	Вывод
5 5 1 2 3 4 5 3 5 2 4	8 8 8 15

Ввод	Вывод
1 4 5 4 5 3	15

E. Хеширование цепочками

Ограничение времени	1 секунда
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Хеширование цепочками — один из наиболее популярных методов реализации хеш-таблиц на практике. Ваша цель в данной задаче — реализовать такую схему, используя таблицу с m ячейками и полиномиальной хеш-функцией на строках $h(S)=(\sum_{|S|-1}^{i=0} S[i]x^i \bmod p) \bmod m$, где $S[i]$ — ASCII-код i-го символа строки S, $p=1000000007$ — простое число, а $x=263$. Ваша программа должна поддерживать следующие типы запросов:

- `add string`: добавить строку string в таблицу. Если такая строка уже есть, проигнорировать запрос;
- `del string`: удалить строку string из таблицы. Если такой строки нет, проигнорировать запрос;
- `find string`: вывести `yes` или `no` в зависимости от того, есть в таблице строка string или нет;
- `check i`: вывести i-й список (используя пробел в качестве разделителя); если i-й список пуст, вывести пустую строку.

При добавлении строки в цепочку, строка должна добавляться **в начало** цепочки.
Задача из курса «Алгоритмы: теория и практика. Структуры данных»: https://stepik.org/course/1547/syllabus

Формат ввода

Первая строка размер хеш-таблицы m. Следующая строка содержит количество запросов n. Каждая из последующих n строк содержит запрос одного из перечисленных выше четырёх типов ($1 \le n \le 10^5$; $n^5 \le m \le n$).
Все строки имеют длину от одного до пятнадцати и содержат только буквы латинского алфавита.

F. Обход двоичного дерева

Ограничение времени	2 секунды
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Требуется построить in-order, pre-order и post-order обходы двоичного дерева.
In-order обход соответствует следующей рекурсивной процедуре, получающей на вход корень v текущего поддерева: произвести рекурсивный вызов для $v.left$, напечатать $v.key$, произвести рекурсивный вызов для $v.right$. Pre-order обход: напечатать $v.key$, произвести рекурсивный вызов для $v.left$, произвести рекурсивный вызов для $v.right$. Post-order: произвести рекурсивный вызов для $v.left$, произвести рекурсивный вызов для $v.right$, напечатать $v.key$.
Задача из курса «Алгоритмы: теория и практика. Структуры данных»: https://stepik.org/course/1547/syllabus

Формат ввода

Первая строка содержит число вершин n. Вершины дерева пронумерованы числами от 0 до $n-1$. Вершина 0 является корнем. Каждая из следующих n строк содержит информацию о вершинах $0, 1, \ldots, n-1$: i-я строка задаёт числа key_i, $left_i$ и $right_i$, где key_i — ключ вершины i, $left_i$ — индекс левого сына вершины i, а $right_i$ — индекс правого сына вершины i. Если у вершины i нет одного или обоих сыновей, соответствующее значение равно −1.
Ограничения. $1 \le n \le 10^5$; $0 \le key_i \le 10^9$; $-1 \le left_i, right_i \le n - 1$. Гарантируется, что вход задаёт корректное двоичное дерево: в частности, если $left_i \neq -1$ и $right_i \neq -1$, то $left_i \neq right_i$; никакая вершина не является сыном двух вершин; каждая вершина является потомком корня.

Формат вывода

Три строки: in-order, pre-order и post-order обходы.

Пример

Ввод	Вывод
10	50 70 80 30 90 40 0 20 10 60

Ввод	Вывод
0 7 2 10 -1 -1 20 -1 6 30 8 9 40 3 -1 50 -1 -1 60 1 -1 70 5 4 80 -1 -1 90 -1 -1	0 70 50 40 30 80 90 20 60 10 50 80 90 30 40 70 10 60 20 0

G. Стек с поддержкой максимума

Ограничение времени	1 секунда
Ограничение памяти	256.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Стек - абстрактная структура данных, поддерживающая операции push*push* и pop*pop*. Несложно реализовать стек так, чтобы обе эти операции работали за константное время. В данной задаче ваша цель - расширить интерфейс стека так, чтобы он дополнительно поддерживал операцию max*max* и при этом чтобы время работы всех операций по-прежнему было константным.
Задача из курса "Алгоритмы: теория и практика. Структуры данных": https://stepik.org/course/1547/syllabus

Формат ввода

Первая строка содержит число запросов q*q*. Каждая из последующих q*q* строк задаёт запрос в одном из следующих форматов: `push v`, `pop`, or `max` ($1 \le q \le 400000$, $0 \le v \le 1000000$).

Формат вывода

Для каждого запроса $\max$*max* выведите (в отдельной строке) текущий максимум на стеке.

Пример 1

Ввод	**Вывод**
5 push 2 push 1 max pop max	2 2

Пример 2

Ввод	**Вывод**
5 push 1 push 2 max pop max	2 1

Пример 3

Ввод	**Вывод**
10 push 2 push 3 push 9	9 9 9 9

Ввод	Вывод
push 7 push 2 max max max pop max	