

Задача А. Примитивы

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

Дано дерево с корнем в вершине 0 (необязательно двоичное). Посчитайте:

1. Высоту дерева – максимальное расстояние от корня до листа.
2. Диаметр дерева – максимальная длина пути между двумя вершинами (естественно, путь не должен проходить через одну вершину несколько раз).
3. Для каждой вершины найдите её глубину – длину пути от корня до вершины.

Формат входных данных

В первой строке вводится натуральное число  $n$  – размер дерева ( $2 \leq n \leq 10^5$ ). В следующей строке записано  $n - 1$  целое число  $p_i$  – предок вершины  $i$  ( $0 \leq p_i < i$ ).

Формат выходных данных

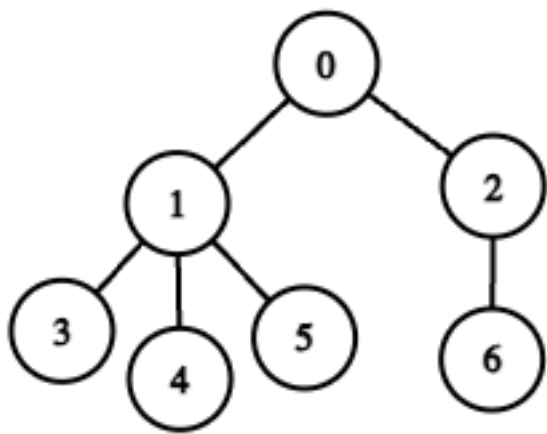
Выведите две строки. В первой строке выведите два числа: высоту и диаметр дерева. Во второй строке для каждой вершины выведите её глубину.

Примеры

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

Замечание

Дерево из первого примера:



## Задача В. AVL?

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

Дано двоичное дерево с корнем в вершине  $r$ . Проверьте, является ли оно корректным AVL-деревом.

Напоминание: AVL-дерево – это дерево, для которого выполняются следующие условия:

- оба поддерева – левое и правое – являются AVL-деревьями;
- все вершины левого поддерева вершины  $X$ , меньше самой вершины  $X$ ;
- все вершины правого поддерева вершины  $X$ , больше самой вершины  $X$ ;
- для каждой вершины высота её двух поддеревьев различается не более чем на 1 (высота – расстояние до самого дальнего листа).

### Формат входных данных

В первой строке вводится натуральное число  $n$  – размер дерева ( $1 \leq n \leq 10^5$ ) и  $r$  – корень дерева ( $0 \leq r < n$ ).

В следующих  $n$  строках записаны два числа  $l_i, r_i$  – левый и правый ребенок  $i$ -й вершины ( $-1 \leq l_i, r_i < n$ ;  $l_i, r_i = -1$ , если у вершины нет соответствующего ребенка).

Гарантируется, что задано корректное двоичное дерево.

### Формат выходных данных

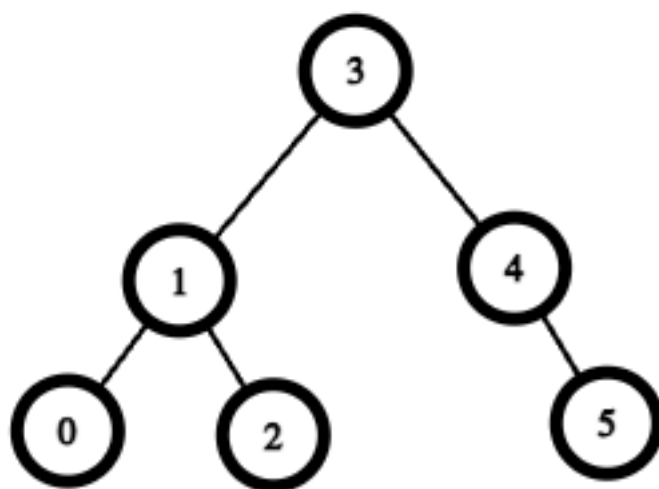
Выведите одно число: 0, если дерево заданное дерево не является AVL-деревом и 1 иначе.

### Примеры

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

### Замечание

Дерево из первого примера:



Задача C. LCA

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

Дано подвешенное дерево с корнем в вершине 0. Вам нужно ответить на  $m$  запросов вида «найти LCA двух вершин». LCA вершин  $u$  и  $v$  в подвешенном дереве – это наиболее удаленная от корня дерева вершина, лежащая на обоих путях от  $u$  и  $v$  до корня.

Формат входных данных

В первой строке вводится натуральное число  $n$  – размер дерева ( $2 \leq n \leq 10^3$ ). В следующей строке записано  $n - 1$  целое число  $p_i$  – предок вершины  $i$  ( $0 \leq p_i < i$ ).  
Затем дано число  $m$ . Далее заданы  $m$  ( $0 < m \leq 10^3$ ) запросов вида  $(u, v)$  – найти LCA двух вершин  $u$  и  $v$  ( $0 \leq u, v < n; u \neq v$ ).

Формат выходных данных

На каждый из  $m$  запросов выведите по одному числу – LCA заданных вершин.

Примеры

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

**Задача D. Хипуй!**

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

В этой задаче вам необходимо организовать структуру данных **Heap** для хранения целых чисел, над которой определены следующие операции:

- 1. **Insert(*X*)** — добавить в **Heap** число *X*;
- 2. **Extract** — достать из **Heap** наибольшее число (удалив его при этом).

Эту задачу нужно решить без использования встроенных структур данных для поиска максимального числа.

**Формат входных данных**

Во входном файле записано количество команд *n* ( $1 \leq n \leq 100000$ ), потом последовательность из *n* команд, каждая в своей строке.

Каждая команда имеет такой формат: «0 число» или «1», что означает соответственно операции «**Insert(число)**» и «**Extract**». Добавляемые числа находятся в интервале от 1 до  $10^7$  включительно.

Гарантируется, что при выполнении команды **Extract** в структуре находится по крайней мере один элемент.

**Формат выходных данных**

В выходной файл для каждой команды извлечения необходимо вывести число, полученное при выполнении команды «**Extract**».

**Пример**

стандартный ввод	стандартный вывод
7	100
0 100	50
0 10	
1	
0 5	
0 30	
0 50	
1	

**Задача Е. Хипуй! Сортируй!**

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

В этой задаче вам необходимо организовать структуру данных **Неар** для хранения целых чисел и с её помощью отсортировать заданный массив.  
Эту задачу нужно решить без использования встроенных алгоритмов/структур данных для сортировок.

**Формат входных данных**

В первой строке входного файла задано одно натуральное число  $n$  ( $1 \leq n \leq 100000$ ). Во второй строке задан массив  $a$  размера  $n$ , где  $-10^9 \leq a_i \leq 10^9$ .

**Формат выходных данных**

В выходной файл необходимо вывести отсортированный массив  $a$ .

**Пример**

стандартный ввод	стандартный вывод
10 1 8 2 1 4 7 3 2 3 6	1 1 2 2 3 3 4 6 7 8

## Задача F. Следующий

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

Реализуйте структуру данных, которая поддерживает множество  $S$  целых чисел, с которым разрешается производить следующие операции:

1. `add(i)` – добавить в множество  $S$  число  $i$  (если он там уже есть, то множество не меняется)
2. `next(i)` – вывести минимальный элемент множества, не меньший  $i$ . Если искомый элемент в структуре отсутствует, необходимо вывести  $-1$ .

Заметьте, что в этой задаче необычные операции ввода. Операция, которую вам задает тест, может зависеть от того, правильно ли вы ответили на предыдущий запрос. Внимательно прочитайте формат ввода. Операция *mod* означает взятие остатка.

### Формат входных данных

Исходно множество  $S$  пусто. Первая строка входного файла содержит  $n$  – количество операций ( $1 \leq n \leq 300\,000$ ). Следующие  $n$  строк содержат операции. Каждая операция имеет вид либо `«+ i»`, либо `«? i»`. Операция `«? i»` задает запрос `next(i)`.

Если операция `«+ i»` идет во входном файле в начале или после другой операции `«+»`, то она задает операцию `add(i)`. Если же она идет после запроса `«?»`, и результат этого запроса был  $y$ , то выполняется операция `add((i+y) mod  $10^9$ )`.

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до  $10^9$ .

### Формат выходных данных

Для каждого запроса выведите одно число – ответ на запрос.

### Пример

стандартный ввод	стандартный вывод
6	3
+ 1	4
+ 3	
+ 3	
? 2	
+ 1	
? 4	