Задача 1. Обходы дерева

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

По заданной последовательности целых чисел построить дерево двоичного поиска и обойти его в прямом и обратном порядках. Повторяющиеся числа в дерево не вставлять.

Формат входных данных

Во входном файле через пробел записаны целые числа. Количество чисел не превосходит 1000.

Формат выходных данных

В первую строку выходного файла нужно вывести значения, содержащиеся в построенном дереве в прямом порядке обхода, а во вторую — в обратном. Числа выводить через пробел.

Пример

input.txt	output.txt
5 1 10 -3 12 1 9 4	5 1 -3 4 10 9 12 -3 4 1 9 12 10 5

Задача 2. Высота дерева

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

По заданной последовательности целых чисел построить дерево двоичного поиска. Повторяющиеся числа в дерево не вставлять. Необходимо посчитать высоту получившегося дерева.

Формат входных данных

Во входном файле через пробел записаны целые числа. Количество чисел не превосходит 1000.

Формат выходных данных

В выходной файл нужно вывести одно целое число — высоту получившегося дерева.

Пример

input.txt	output.txt
5 1 10 -3 12 1 9 4	2

Задача 3. Количество листьев в дереве

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

По заданной последовательности целых чисел построить дерево двоичного поиска. Повторяющиеся числа в дерево не вставлять. Необходимо посчитать количество вершин, не имеющих сыновей, т.е. листьев дерева.

Формат входных данных

Во входном файле через пробел записаны целые числа. Количество чисел не превосходит 1000.

Формат выходных данных

В выходной файл нужно вывести одно целое число — количество листьев в заданном дереве.

Пример

input.txt	output.txt
5 1 10 -3 12 1 9 4	4

Задача 4. Количество вершин в дереве на заданном уровне

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

По заданной последовательности целых чисел построить дерево двоичного поиска. Повторяющиеся числа в дерево не вставлять. Необходимо посчитать количество вершин, расположенных на заданном уровне получившегося дерева.

Формат входных данных

В первой строке входного файла указан уровень дерева, на котором нужно посчитать количество вершин — это неотрицательное целое число, не превосходящее 1000.

В следующей строке через пробел записаны целые числа. Количество чисел не превосходит 1000.

Формат выходных данных

В выходной файл нужно вывести одно целое число — количество вершин, расположенных на заданном уровне получившегося дерева.

Пример

input.txt	output.txt
1 5 1 10 -3 12 1 9 4	2

Задача 5. Дерево двоичного поиска слов

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

По заданным словам построить дерево двоичного поиска и обойти его в инфиксном порядке. Повторяющиеся слова в дерево не вставлять.

Формат входных данных

Входной файл содержит строки, в каждой из которых записано по одному слову. Длина каждого слова не превосходит 100 символов. Количество слов не превосходит 1000. Пустых строк нет.

В следующей строке через пробел записаны целые числа. Количество чисел не превосходит 1000.

Формат выходных данных

В выходной файл нужно выдать эти слова, упорядоченные в лексикографическом порядке, по одному на строке.

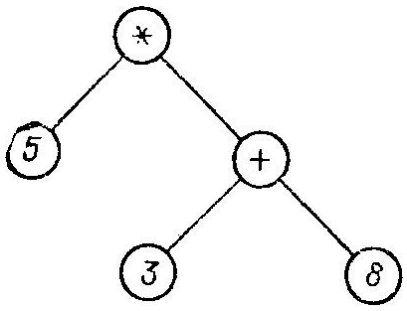
Пример

input.txt	output.txt
orange mallon apple grapes plum banana	apple banana grapes mallon orange plum

Задача 6. Дерево-формула

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Деревом-формулой называется двоичное дерево, в листьях которого расположены цифры, а в вершинах, которые не являются листьями — знаки арифметических операций $(+, -, *, /)$. На рисунке показано дерево-формула, соответствующее формуле $(5 * (3 + 8))$.



Описать рекурсивную функцию, которая вычисляет (как целое число) значение дерева-формулы.

Формат входных данных

Во входном файле записано выражение в префиксной форме. По этой записи нужно построить двоичное дерево. Длина строки во входном файле не превосходит 1000 знаков.

Формат выходных данных

В выходной файл нужно выдать одно целое число — значение выражения, заданного деревом.

Если возникнет деление на 0, то выдать слово `NO`.

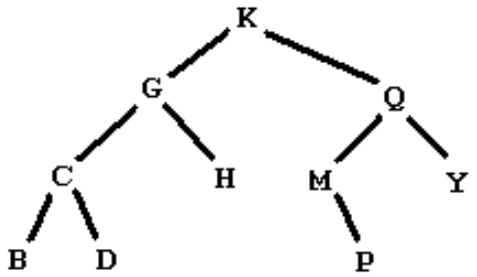
Пример

input.txt	output.txt
*5+38	55

Задача 7. Листопад

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

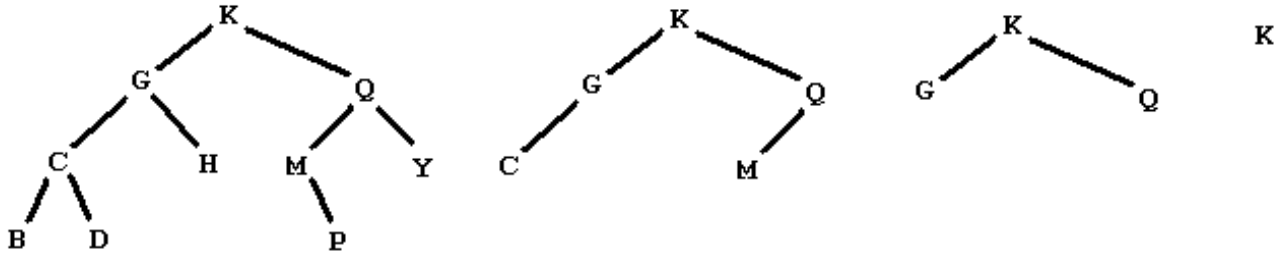
На рисунке изображено графическое представление двоичного дерева поиска символов.



Рассмотрим следующую последовательность действий на дереве двоичного поиска символов:

- Удалить листья и вывести данные удаленных листьев.
- Повторять пока дерево не пусто

Начиная от дерева, нижнего слева, мы получим последовательность показанных деревьев, и затем пустое дерево.



При этом мы последовательно будем удалять следующие данные:

```
BDHPY
CM
GQ
K
```

Ваша задача состоит в том, чтобы исходя из такой последовательности строк листьев дерева бинарного поиска символов, выдать префиксный обход этого дерева.

Формат входных данных

Входной файл состоит последовательности одной или нескольких строк заглавных символов. Строки содержат листья, удаленные из бинарного дерева поиска в последовательности, описанной выше. Символы в строке перечисляются в возрастающем алфавитном порядке. Входной файл не содержит пробелов и пустых строк.

Формат выходных данных

Для входных данных существует единственное бинарное дерево поиска, которое даст последовательность листьев. Выходной файл состоит из строки, состоящей из префиксного обхода такого дерева, без пробелов.

Примеры

input.txt	output.txt
BDHPY CM GQ K	KGCBDHQMPY
AC B	BAC

Задача 8. Пляски с бубном

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

— А это, кажется, будет тебе интересно, — Эдик Амперян положил на стол передо мной копию какой-то статьи.

В статье рассказывалось о магической системе, основанной на применении бубнов. Каждой нештатной ситуации поставлена в соответствие последовательность ударов правого и левого бубна так, что в результате получается полное двоичное дерево из $M = 2^N - 1$ вершин. Статья обосновывала применимость данного метода к поиску и устранению ошибок в компьютерных системах. При этом проблема заключалась в том, что авторы статьи использовали различные источники, в которых вершины были занумерованы, начиная с единицы, одним из следующих способов:

- **инфиксный**

  Вершины нумеруются в следующем порядке: сначала левая ветка, потом корень, потом правая (источники, связанные с чукотскими шаманами и Windows Vista).

- **постфиксный**

  В этом случае сначала нумеруем левую ветку, потом правую, а потом корень (источники, связанные с якутскими шаманами и Windows XP).

- **префиксный**

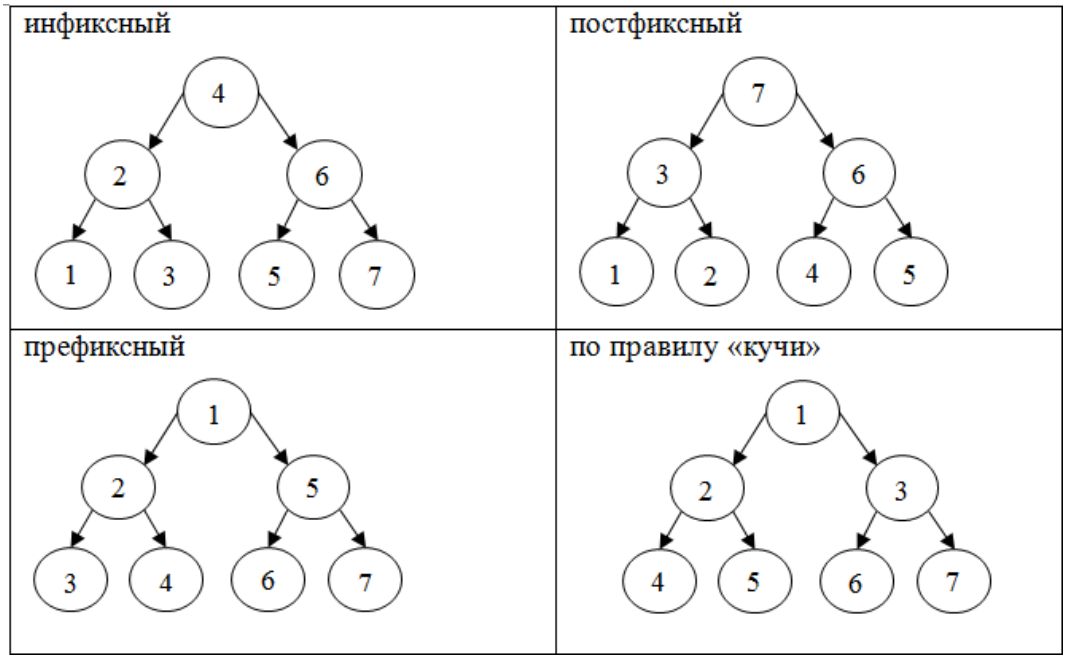
  Тут наоборот: сначала корень, потом левую ветку, потом правую (источники, связанные с эскимосскими шаманами и FreeBSD).

- **согласно правилу «кучи»**

  Вершины нумеруются слева направо по уровням, сверху вниз, начиная с корня (источники, связанные с шаманами племени яки и Linux).

Я перечитал эту статью и понял, что перед тем, как проверять её утверждения и начинать пляски с бубном, необходимо научиться делать преобразование из любого типа нумерации в любой.

Ниже дан пример нумерации дерева для $N = 3, M = 7$:



Формат входных данных

В первой строке входного файла дано целое число L — количество деревьев ($0 < L \leqslant 1000$).

В каждой из следующих L строк даны два целых числа N_i и K_i и одна строка S. N_i задаёт высоту дерева, K_i задаёт номер элемента в нём, а S задаёт, каким способом задана нумерация: если $S =$ `INF` — нумерация инфиксная, `POSTF` — постфиксная, `PREF` — префиксная, `HEAP` — согласно правилу «кучи» ($0 < N_i \leqslant 60, 1 \leqslant K_i \leqslant 2^{N_i} - 1$).

Формат выходных данных

В L строках выходного файла нужно вывести в порядке, соответствующему входному файлу, по четыре числа в каждой строке: номер элемента в инфиксной записи, номер элемента в постфиксной записи, номер элемента в префиксной записи, и номер элемента, вычисленный по правилу «кучи».

Пример

input.txt	output.txt
3 3 1 PREF 3 7 INF 4 5 HEAP	4 7 1 1 7 5 7 7 6 6 6 5

Задача 9. sql join: дерево поиска

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды
Ограничение по памяти:	разумное

Данная задача является продолжением задачи «sql join» из задания 10. Прочитайте условие оригинальной задачи перед тем, как читать дальше!

В данной задаче необходимо использовать **дерево поиска** для ускорения соединения. Занесите все записи одной таблицы в бинарное дерево поиска (при этом нужно решить, что делать с равными ключами). Далее переберите все записи второй таблицы: для каждой из них можно найти в дереве поиска список всех записей с совпадающим именем.

Входные/выходные данные в этой задаче такие же, как в задаче «sql join». Ограничения такие же ($1 \leqslant N, M \leqslant 10^5$), за одним важным исключением:

В этой задаче **не** гарантируется, что $N \cdot M \leqslant 10^5$. Вместо этого гарантируется, что после соединения количество записей R не превышает 10^5.