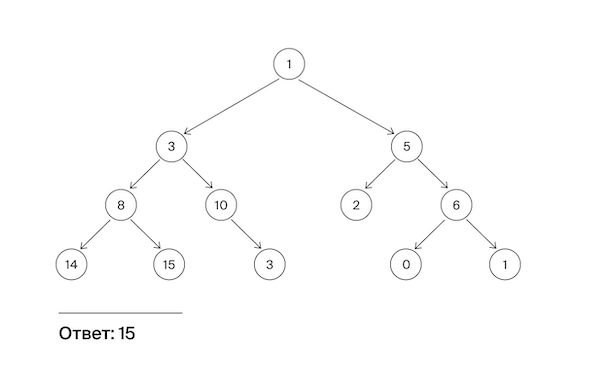
A. Лампочки

Гоша повесил на стену гирлянду в виде бинарного дерева, в узлах которого находятся лампочки. У каждой лампочки есть своя яркость. Уровень яркости лампочки соответствует числу, расположенному в узле дерева. Помогите Гоше найти самую яркую лампочку в гирлянде, то есть такую, у которой яркость наибольшая.



Формат ввода

На вход подается корень дерева.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/p8QXeIV6w_97Jw).

Python:

class Node:    
    def \_\_init\_\_(self, value, left=None, right=None):    
        self.value = value    
        self.right = right    
        self.left = left

Ваша функция должна иметь сигнатуру solution(Node) -> int.  
  
С++:

struct Node{    
  int value;    
  const Node\* left = nullptr;    
  const Node\* right = nullptr;    
};    
int Solution(const Node\* root);

Нужно подключить solution\_tree.h  
  
Go:

type Node struct {    
value  int    
left   \*Node    
right  \*Node    
}

Ваша функция должна иметь сигнатуру func Solution(\*Node) int.  
  
JS:

class CNode {    
    constructor(value) {    
        this.value = value;    
        this.left = null;    
        this.right = null;    
    }    
}

Ваша функция должна иметь сигнатуру solution :: CNode -> Number.  
  
Java:  
Файл должен содержать public class Solution с функцией  
public static int treeSolution(Node head)

class Node {    
    int value;    
    Node left;    
    Node right;    
   
    Node(int value) {    
        this.value = value;    
        right = null;    
        left = null;    
    }    
}

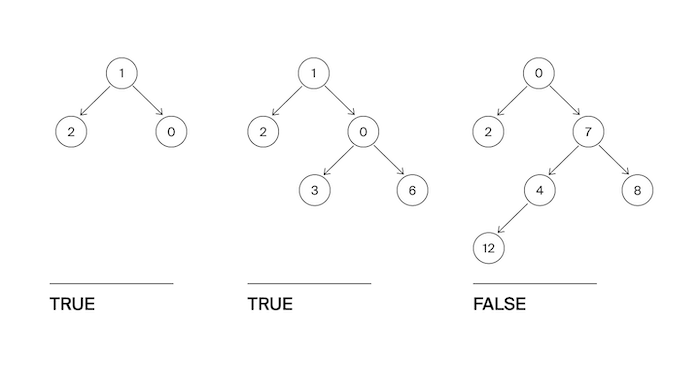
Формат вывода

Функция должна вернуть максимальное значение яркости в узле дерева.

B. Сбалансированное дерево

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Гоше очень понравилось слушать рассказ Тимофея про деревья. Особенно часть про сбалансированные деревья. Он решил написать функцию, которая определяет, сбалансировано ли дерево.

Дерево считается сбалансированным, если левое и правое поддеревья каждой вершины отличаются по высоте не больше, чем на единицу.  


Формат ввода

На вход функции подаётся корень бинарного дерева.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/njVBc1ltPjMT3Q).

Python:

class Node:    
    def \_\_init\_\_(self, value, left=None, right=None):    
        self.value = value    
        self.right = right    
        self.left = left

Ваша функция должна иметь сигнатуру solution(Node root) -> bool.  
  
С++:

struct Node{    
  int value;    
  const Node\* left = nullptr;    
  const Node\* right = nullptr;    
};    
bool Solution(const Node\* root);

Нужно подключить solution\_tree.h  
  
Go:

type Node struct {    
value int    
left  \*Node    
right \*Node    
}

Ваша функция должна иметь сигнатуру Solution(root \*Node) bool.  
  
JS:

class CNode {    
    constructor(value) {    
        this.value = value;    
        this.left = null;    
        this.right = null;    
    }    
}

Ваша функция должна иметь сигнатуру solution(root) :: CNode -> bool.  
  
Java:  
Файл должен содержать public class Solution с public static boolean treeSolution(Node head)

class Node {    
    int value;    
    Node left;    
    Node right;    
   
    Node(int value) {    
        this.value = value;    
        right = null;    
        left = null;    
    }    
}

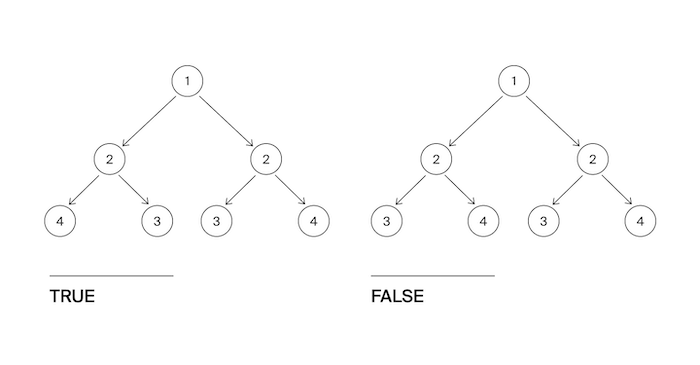
Формат вывода

Функция должна вернуть True, если дерево сбалансировано в соответствии с критерием из условия, иначе - False.

C. Дерево - анаграмма

Гоша и Алла играют в игру «Удивительные деревья». Помогите ребятам определить, является ли дерево, которое им встретилось, деревом-анаграммой?

Дерево называется анаграммой, если оно симметрично относительно своего центра.



Формат ввода

Напишите функцию, которая определяет, является ли дерево анаграммой.  
На вход подаётся корень дерева.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/e4lXsgBMX74rRg).

Python:

class Node:    
    def \_\_init\_\_(self, value, left=None, right=None):    
        self.value = value    
        self.right = right    
        self.left = left

Ваша функция должна иметь сигнатуру solution(root: Node) -> bool.  
  
С++:

struct Node{    
  int value;    
  const Node\* left = nullptr;    
  const Node\* right = nullptr;    
};    
bool Solution(const Node\* root);

Нужно подключить solution\_tree.h  
  
Go:

type Node struct {    
value int    
left  \*Node    
right \*Node    
}

Ваша функция должна иметь сигнатуру Solution(root \*Node) bool.  
  
JS:

class CNode {    
    constructor(value) {    
        this.value = value;    
        this.left = null;    
        this.right = null;    
    }    
}

Ваша функция должна иметь сигнатуру solution :: CNode -> bool.  
  
Java:  
Файл должен содержать public class Solution с  
public static boolean treeSolution(Node head)

class Node {    
    int value;    
    Node left;    
    Node right;    
   
    Node(int value) {    
        this.value = value;    
        right = null;    
        left = null;    
    }    
}

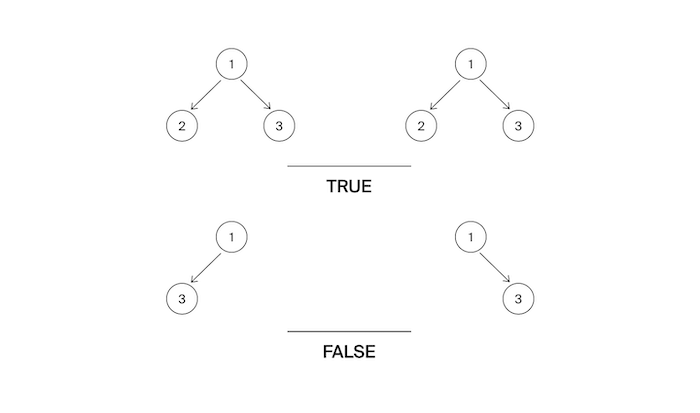
Формат вывода

Функция должна вернуть True если дерево является анаграммой. Иначе - False.

D. Деревья - близнецы

Гоше на день рождения подарили два дерева. Тимофей сказал, что они совершенно одинаковые. Но, по мнению Гоши, они отличаются.

Помогите разрешить этот философский спор!



Формат ввода

На вход подаются корни двух деревьев.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/RRvTegFORsPR2Q).

Класс, представляющий узел дерева.  
Python:

class Node:    
    def \_\_init\_\_(self, value, left=None, right=None):    
        self.value = value    
        self.right = right    
        self.left = left

Ваша функция должна иметь сигнатуру solution(root1: Node, root2: Node) -> bool.  
  
С++:

struct Node{    
  int value;    
  const Node\* left = nullptr;    
  const Node\* right = nullptr;    
};    
bool Solution(const Node\* root1, const Node\* root2);

Нужно подключить solution\_tree.h  
  
Go:

type Node struct {    
value int    
left  \*Node    
right \*Node    
}

Ваша функция должна иметь сигнатуру Solution(root1 \*Node, root2 \*Node) bool.  
  
JS:

class CNode {    
    constructor(value) {    
        this.value = value;    
        this.left = null;    
        this.right = null;    
    }    
}

Ваша функция должна иметь сигнатуру solution :: (CNode, CNode) -> bool.  
  
Java:  
Файл должен содержать public class Solution  
с public static boolean treeSolution(Node head1, Node head2)

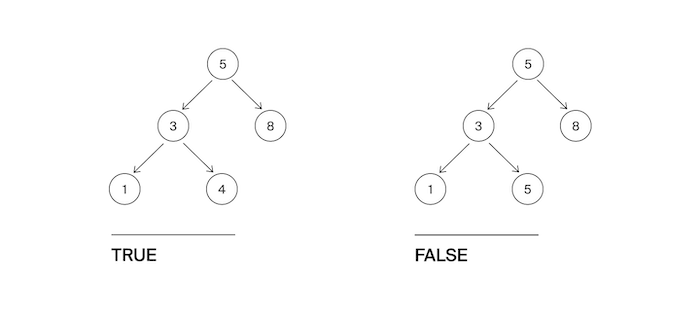
class Node {    
    int value;    
    Node left;    
    Node right;    
   
    Node(int value) {    
        this.value = value;    
        right = null;    
        left = null;    
    }    
}

Формат вывода

Функция должна вернуть True если деревья являются близнецами. Иначе - False.

E. Дерево поиска

Гоша понял, что такое дерево поиска, и захотел написать функцию, которая определяет, является ли заданное дерево деревом поиска. Значения в левом поддереве должны быть строго меньше, в правом —- строго больше значения в узле.

Помогите Гоше с реализацией этого алгоритма.  


Формат ввода

На вход функции подается корень бинарного дерева.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/ngvc3LXa7o0Gcw).

Python:

class Node:    
    def \_\_init\_\_(self, value, left=None, right=None):    
        self.value = value    
        self.right = right    
        self.left = left

Ваша функция должна иметь сигнатуру solution(root: Node) -> bool.  
  
С++:

struct Node{    
  int value;    
  const Node\* left = nullptr;    
  const Node\* right = nullptr;    
};    
bool Solution(const Node\* root);

Нужно подключить solution\_tree.h  
  
Go:

type Node struct {    
    value int    
    left  \*Node    
    right \*Node    
}

Ваша функция должна иметь сигнатуру Solution(root \*Node) bool.  
  
JS:

class CNode {    
    constructor(value) {    
        this.value = value;    
        this.left = null;    
        this.right = null;    
    }    
}

Ваша функция должна иметь сигнатуру solution :: CNode -> bool.  
  
Java:  
Файл должен содержать класс public class Solution с функцией  
public static boolean treeSolution(Node head)

class Node {    
    int value;    
    Node left;    
    Node right;    
   
    Node(int value) {    
        this.value = value;    
        right = null;    
        left = null;    
    }    
}

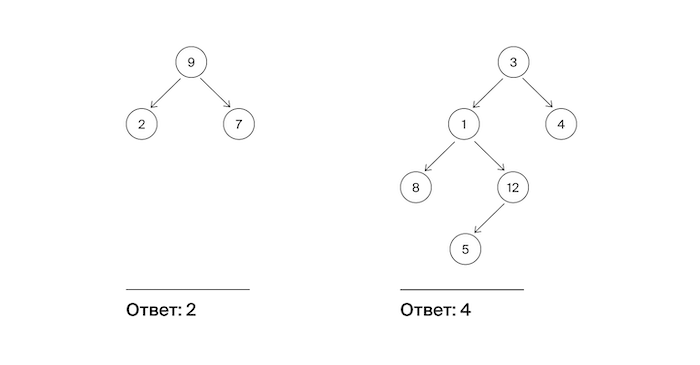
Формат вывода

Функция должна вернуть True, если дерево является деревом поиска, иначе - False.

F. Максимальная глубина

Алла хочет побывать на разных островах архипелага Алгосы. Она составила карту. Карта представлена в виде дерева: корень обозначает центр архипелага, узлы –— другие острова. А листья —– это дальние острова, на которые Алла хочет попасть.

Помогите Алле определить максимальное число островов, через которые ей нужно пройти для совершения одной поездки от стартового острова до места назначения, включая начальный и конечный пункты.



Формат ввода

На вход подается корень дерева.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/vmZ9QZzv8k4ZXw).

Python:

class Node:    
    def \_\_init\_\_(self, value, left=None, right=None):    
        self.value = value    
        self.right = right    
        self.left = left

Ваша функция должна иметь сигнатуру solution(root: Node) -> int.  
  
С++:

struct Node{    
  int value;    
  const Node\* left = nullptr;    
  const Node\* right = nullptr;    
};    
int Solution(const Node\* root);

Нужно подключить solution\_tree.h  
  
Go:

type Node struct {    
    value  int    
    left   \*Node    
    right  \*Node    
}

Ваша функция должна иметь сигнатуру Solution(root \*Node) int.  
  
JS:

class CNode {    
    constructor(value) {    
        this.value = value;    
        this.left = null;    
        this.right = null;    
    }    
}

Ваша функция должна иметь сигнатуру solution :: CNode -> int.  
  
Java:  
Файл должен содержать public class Solution с функцией  
public static int treeSolution(Node head)

class Node {    
    int value;    
    Node left;    
    Node right;    
   
    Node(int value) {    
        this.value = value;    
        right = null;    
        left = null;    
    }    
}

Формат вывода

Функция должна вернуть число, равное максимальному числу островов в пути (включая начальный и конечный пункты).

Примечания

Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

G. Максимальный путь в дереве

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Тимофей устраивает соревнования по спортивному ориентированию в своём офисе. Схема офиса представлена в виде дерева. Посещая каждый пункт, можно зарабатывать или терять очки. Если в узле записано положительное число, это значение добавляется к текущему количеству очков участника. Если отрицательное —– очки вычитаются. Если 0 –— их количество не меняется.

Нужно определить, какое максимальное число очков можно заработать, пройдя по пути из какого-то пункта A в какой-то (возможно, тот же) пункт B.

Путь не обязательно должен проходить через корень или содержать лист. Путь должен содержать по крайней мере один узел.

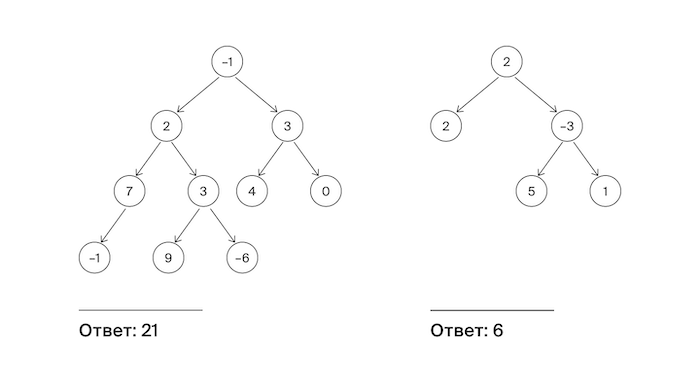
Пример 1:  
В дереве всего три вершины, во всех вершинах записаны положительные числа, поэтому объединим все три вершины в один путь. В ответе получим: 1+1+2=4.

Пример 2:  
Теперь в дереве есть вершина с отрицательным весом, через неё в данном случае проходить будет невыгодно. Оптимальный путь: 2+7+3=12.

Пример 3:  
Оптимальный путь: 7+2+3+9=21.

Пример 4:  
В этот раз имеет смысл пройти через вершину с отрицательным весом, так как в левом поддереве вершины −3 лежит 5. Оптимальный путь: 2+2−3+5=6.

**Требования к решению: передаваемое в качестве аргумента дерево** **нельзя менять. Не храните вспомогательную информацию в** **вершинах.**



Формат ввода

На вход подается корень дерева.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/sfnNhX5mCfNO2w).

Python:

class Node:    
    def \_\_init\_\_(self, value, left=None, right=None):    
        self.value = value    
        self.right = right    
        self.left = left

Ваша функция должна иметь сигнатуру solution(node: Node) -> int.  
  
С++:

struct Node{    
  int value;    
  const Node\* left = nullptr;    
  const Node\* right = nullptr;    
};

Ваша функция должна иметь сигнатуру int Solution(const Node\* root); Нужно подключить solution\_tree.h  
  
Go:

type Node struct {    
    value  int    
    left   \*Node    
    right  \*Node    
}

Ваша функция должна иметь сигнатуру Solution(node \*Node) int.  
  
JS:

class CNode {    
    constructor(value) {    
        this.value = value;    
        this.left = null;    
        this.right = null;    
    }    
}

Ваша функция должна иметь сигнатуру solution :: CNode -> Number.  
  
Java:  
Файл должен содержать public class Solution с public static int treeSolution(Node head)

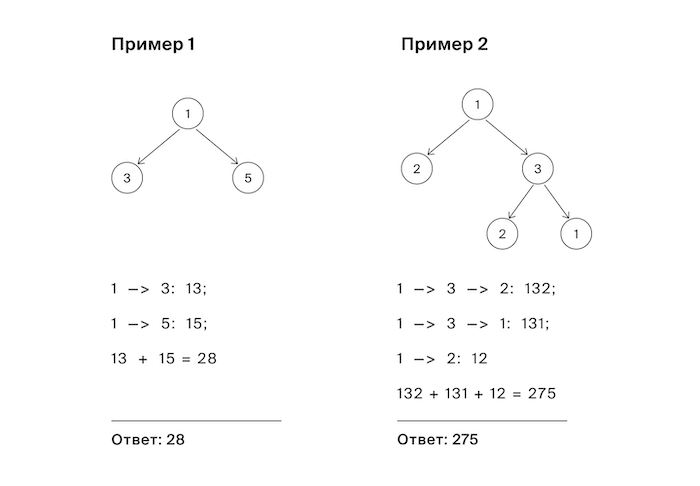
class Node {    
    int value;    
    Node left;    
    Node right;    
   
    Node(int value) {    
        this.value = value;    
        right = null;    
        left = null;    
    }    
}

Формат вывода

Функция должна вернуть число, равное максимальному количеству очков, которое можно заработать, попав из города А в город В.

H. Числовые пути

Вася и его друзья решили поиграть в игру. Дано дерево, в узлах которого записаны цифры от 0 до 9. Таким образом, каждый путь от корня до листа содержит число, получившееся конкатенацией цифр пути (склеиванием цифр пути в одно число). Нужно найти сумму всех таких чисел в дереве.

Гарантируется, что ответ не превосходит 20 000.  


Формат ввода

На вход подается корень дерева.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/0DxM4gXOC5x3Pg).

Python:

class Node:    
    def \_\_init\_\_(self, value, left=None, right=None):    
        self.value = value    
        self.right = right    
        self.left = left

Ваша функция должна иметь сигнатуру solution(root: Node) -> int.  
  
С++:

struct Node{    
  int value;    
  const Node\* left = nullptr;    
  const Node\* right = nullptr;    
};    
int Solution(const Node\* root);

Нужно подключить solution\_tree.h  
  
Go:

type Node struct {    
    value  int    
    left   \*Node    
    right  \*Node    
}

Ваша функция должна иметь сигнатуру Solution(root \*Node) int.  
  
JS:

class CNode {    
    constructor(value) {    
        this.value = value;    
        this.left = null;    
        this.right = null;    
    }    
}

Ваша функция должна иметь сигнатуру solution :: CNode -> int.  
  
Java:  
Файл должен содержать public class Solution с функцией  
public static int treeSolution(Node head)

class Node {    
    int value;    
    Node left;    
    Node right;    
   
    Node(int value) {    
        this.value = value;    
        right = null;    
        left = null;    
    }    
}

Формат вывода

Функция должна вернуть число, равное сумме чисел всех путей в дереве.

# I. Разные деревья поиска

Ребятам стало интересно, сколько может быть различных деревьев поиска, содержащих в своих узлах все уникальные числа от 1 до n. Помогите им найти ответ на этот вопрос.

## Формат ввода

В единственной строке задано число n. Оно не превосходит 20.

## Формат вывода

Нужно вывести число, равное количеству различных деревьев поиска, в узлах которых могут быть размещены числа от 1 до n включительно.

### Пример 1

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 2 | 2 |

### Пример 2

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 3 | 5 |

### Пример 3

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| 4 | 14 |

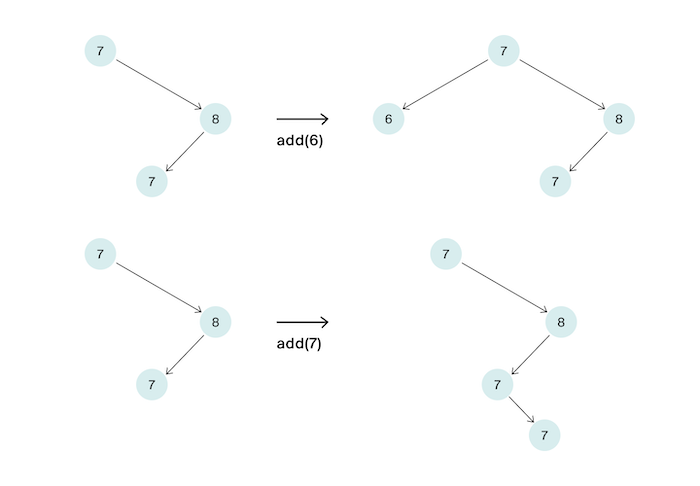
J. Добавь узел

Дано BST. Надо вставить узел с заданным ключом. Ключи в дереве могут повторяться.

На вход функции подаётся корень корректного бинарного дерева поиска и ключ, который надо вставить в дерево. Осуществите вставку этого ключа. Если ключ уже есть в дереве, то его дубликаты уходят в правого сына. Таким образом вид дерева после вставки определяется однозначно. Функция должна вернуть корень дерева после вставки вершины.

Ваше решение должно работать за O(h), где h –— высота дерева.

На рисунках ниже даны два примера вставки вершин в дерево.



Формат ввода

Ключи дерева – натуральные числа, не превосходящие 109. Число вершин в дереве не превосходит 105.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение. Ниже приведены сигнатуры функций, которые надо реализовать.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/lVil2dE14f0nUg).

**Python**

*# do not declare Node in your submit-file*   
**class** Node:   
    **def** \_\_init\_\_(self, left=None, right=None, value=0):   
        self.value = value   
        self.right = right   
        self.left = left   
   
**from** node **import** Node *# Attention!*   
**def** insert(root: Node, key: **int**) -> Node

**C++**

*// do not declare Node in your submit-file*   
**struct** Node {   
    Node∗ left;   
    Node∗ right;   
    **int** value;   
};   
**#include** "solution.h" *// Attention!*   
Node∗ insert(Node∗ root, **int** key);

**Java**

*// do not declare Node in your submit-file*   
**public** **class** Node {   
    **private** Node left;   
    **private** Node right;   
    **private** **int** value;   
    **public** **int** getValue();   
    **public** Node getRight();   
    **public** Node getLeft();   
    **public** **void** setValue(**int** value);   
    **public** **void** setRight(Node right);   
    **public** **void** setLeft(Node left);   
    **public** Node(Node left, Node right, **int** value);   
}   
**public** **class** Solution {   
        **public** **static** Node insert(Node root, **int** key);   
}

**Go**

*// do not declare Node in your submit-file*   
type Node **struct** {   
        value    **int**   
        left   ∗Node   
        right  ∗Node   
}   
package main   
   
func insert(root ∗Node, key **int**) ∗Node

**NodeJs**

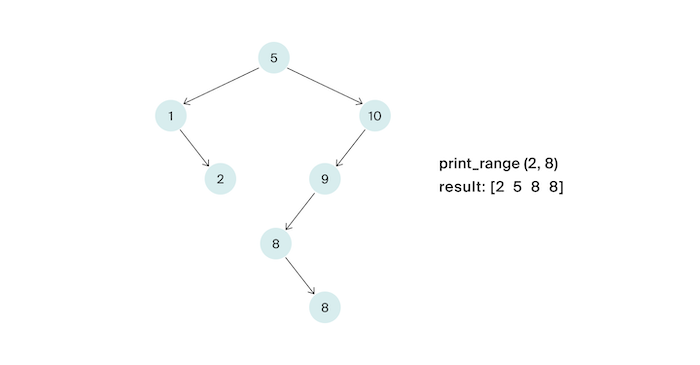
*// do not declare Node in your submit-file*   
**class** Node {   
    constructor(value) {   
        **this**.value = value;   
        **this**.left = **null**;   
        **this**.right = **null**;   
    }   
}   
function insert(node, key) {   
*// Your code*   
}

K. Выведи диапазон

Напишите функцию, которая будет выводить по неубыванию все ключи от L до R включительно в заданном бинарном дереве поиска.

Ключи в дереве могут повторяться. Решение должно иметь сложность O(h+k), где h –— глубина дерева, k — число элементов в ответе.

В данной задаче если в узле содержится ключ x, то другие ключи, равные x, могут быть как в правом, так и в левом поддереве данного узла. (Дерево строил стажёр, так что ничего страшного).



Формат ввода

На вход функции подаётся корень дерева и искомый ключ. Число вершин в дереве не превосходит 105. Ключи – натуральные числа, не превосходящие 109. Гарантируется, что L≤R.

В итоговом решении не надо определять свою структуру / свой класс, описывающий вершину дерева.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/_lfcXwO2RyVlxQ).

**Python**

*# do not declare Node in your submit-file*   
**class** Node:   
    **def** \_\_init\_\_(self, left=None, right=None, value=0):   
        self.value = value   
        self.right = right   
        self.left = left   
*#################*   
**def** print\_range(node: Node, l: **int**, r: **int**) -> **pass**

**C++**

*// do not declare Node in your submit-file*   
**struct** Node {   
    Node∗ left;   
    Node∗ right;   
    **int** value;   
};   
*//*   
**#include** "solution.h" *// Attention!*   
**void** print\_range(Node∗ root, **int** L, **int** R);

**Java**

*// do not declare Node in your submit-file*   
**public** **class** Node {   
    **private** Node left;   
    **private** Node right;   
    **private** **int** value;   
    **public** **int** getValue();   
    **public** Node getRight();   
    **public** Node getLeft();   
    **public** **void** setValue(**int** value);   
    **public** **void** setRight(Node right);   
    **public** **void** setLeft(Node left);   
    **public** Node(Node left, Node right, **int** value);   
}   
*//*   
**public** **class** Solution {   
    **public** **static** **void** printRange(Node root, **int** L, **int** R, BufferedWriter writer) {   
*// Your code*   
    }   
}

**Go**

*// do not declare Node in your submit-file*   
type Node **struct** {   
        value    **int**   
        left   ∗Node   
        right  ∗Node   
}   
*//*   
package main   
func printRange(root ∗Node, left **int**, right **int**)

**NodeJs**

*// do not declare Node in your submit-file*   
**class** Node {   
    constructor(value) {   
        **this**.value = value;   
        **this**.left = **null**;   
        **this**.right = **null**;   
    }   
}   
*//*   
function printRange(root, left, right)

Формат вывода

Функция должна напечатать по неубыванию все ключи от L до R по одному в строке.

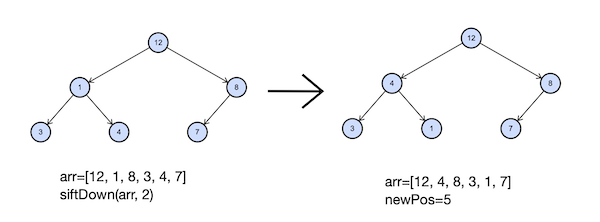
L. Просеивание вниз

|  |  |
| --- | --- |
| Ограничение времени | 2 секунды |
| Ограничение памяти | 80Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Напишите функцию, совершающую просеивание вниз в куче на максимум. Гарантируется, что порядок элементов в куче может быть нарушен только элементом, от которого запускается просеивание.

Функция принимает в качестве аргументов массив, в котором хранятся элементы кучи, и индекс элемента, от которого надо сделать просеивание вниз. Функция должна вернуть индекс, на котором элемент оказался после просеивания. Также необходимо изменить порядок элементов в переданном в функцию массиве.

Индексация в массиве, содержащем кучу, начинается с единицы. Таким образом, сыновья вершины на позиции v это 2v и 2v+1. Обратите внимание, что нулевой элемент в передаваемом массиве фиктивный, вершина кучи соответствует 1-му элементу.



Формат ввода

Элементы кучи —– целые числа, лежащие в диапазоне от −109 до 109. Все элементы кучи уникальны. Передаваемый в функцию индекс лежит в диапазоне от 1 до размера передаваемого массива. В куче содержится от 1 до 105 элементов.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/iYqbFMlTZe2L2A).

**Python**

**def** sift\_down(heap: **list**, idx: **int**) -> **int**

**C++**

**#include** "solution.h" *// Attention!*   
**int** siftDown(std::vector<**int**>& heap, **int** idx);

**Java**

**public** **class** Solution {   
    **public** **static** **int** siftDown(**int**[] heap, **int** idx) {   
*// your code*   
    }   
}

**GO**

package main   
func siftDown(heap []**int**, idx **int**) **int**;

**NodeJS**  
Пожалуйста, не используйте стрелочную нотацию при определении функции

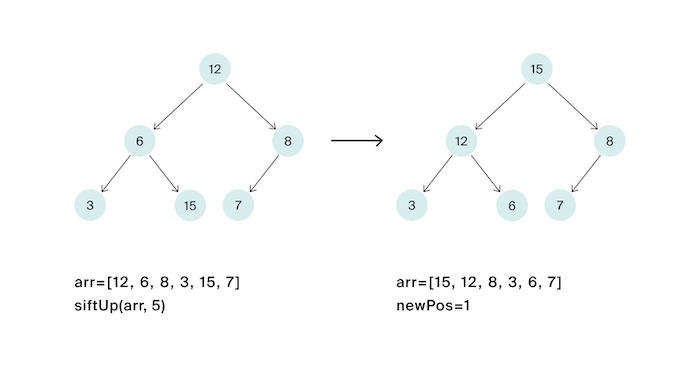
function siftDown(hep, idx) {   
*// Your code*   
}

M. Просеивание вверх

Напишите функцию, совершающую просеивание вверх в куче на максимум. Гарантируется, что порядок элементов в куче может быть нарушен только элементом, от которого запускается просеивание.

Функция принимает в качестве аргументов массив, в котором хранятся элементы кучи, и индекс элемента, от которого надо сделать просеивание вверх. Функция должна вернуть индекс, на котором элемент оказался после просеивания. Также необходимо изменить порядок элементов в переданном в функцию массиве.

Индексация в массиве, содержащем кучу, начинается с единицы. Таким образом, сыновья вершины на позиции v это 2v и 2v+1. Обратите внимание, что нулевой элемент в передаваемом массиве фиктивный, вершина кучи соответствует 1-му элементу.



Формат ввода

Элементы кучи —– целые числа, лежащие в диапазоне от −109 до 109. Все элементы кучи уникальны. Передаваемый в функцию индекс лежит в диапазоне от 1 до размера передаваемого массива. В куче содержится от 1 до 105 элементов.

**Замечания про отправку решений**  
Выберите компилятор make. Решение нужно отправлять в виде файла с расширением, которое соответствует вашему языку программирования. Если вы пишете на Java, имя файла должно быть Solution.java, для C# – Solution.cs. Для остальных языков назовите файл my\_solution.ext, заменив ext на необходимое расширение.

Мы рекомендуем воспользоваться заготовками кода для данной задачи, расположенными по [ссылке](https://disk.yandex.ru/d/XtYa7EqfPz6bmA).

**Python**

**def** sift\_up(heap: **list**, idx: **int**) -> **int**

**C++**

**#include** "solution.h" *// Attention!*   
**int** siftUp(std::vector<**int**>& heap, **int** idx);

**Java**

**public** **class** Solution {   
    **public** **static** **int** siftUp(**int**[] heap, **int** idx) {   
*// your code*   
    }   
}

**GO**

func siftUp(heap []**int**, idx **int**) **int**;

**NodeJS**  
Пожалуйста, не используйте стрелочную нотацию при определении функции

function siftUp(heap, idx) {   
*// Your code*   
}

N. Разбиение дерева

|  |  |
| --- | --- |
| Ограничение времени | 2.5 секунд |
| Ограничение памяти | 128Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Дано бинарное дерево поиска, в котором хранятся целые числа. От этого дерева надо отделить k самых маленьких элементов.

Реализуйте функцию, которая принимает корень дерева и число k, а возвращает два BST — в первом k наименьших элементов из исходного дерева, а во втором — оставшиеся вершины BST.

В вершинах дерева уже записаны корректные размеры поддеревьев (точное название поля смотрите в заготовках кода). После разбиения размеры должны остаться корректными — вам придётся пересчитывать их на ходу.

Ваше решение должно иметь асимптотику O(h), где h — высота исходного дерева.

Выберите компилятор Make и отправьте решение в виде файла с разрешением, соответствующим вашему языку. Сигнатуры для всех языков приведены в заготовках на [диске](https://disk.yandex.ru/d/dosbzqSafoevbg).

Формат ввода

Числа, записанные в вершинах дерева, лежат в диапазоне от 0 до 109. Дерево не содержит одинаковых ключей.

Число вершин в дереве не превосходит 105.