[Tree\_1\_contest](https://contest.yandex.ru/contest/60222/enter/?retPage=)

B. postorder

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64.0 Мб |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Дана верхушка бинарного дерева. Вам нужно заполнить **List < Integer>**значениями из **TreeNode (node)** в порядке - "*postorder*".

Ограничения:

* Количество Nodes в дереве **TreeNode** в отрезке - [0, 100]
* -100 <= node.val <=100

Формат ввода

*1 2 3 null null 3 4*

Формат вывода

*2 3 4 3 1*

Примечания

Используйте этот подготовленный код и допишите Вашу логику в методе **toDo()**.так же Вы можете использовать другие языки из списка (документация использования: <https://yandex.com/support2/contest/en/examples-stdin-stdout>)

import java.io.\*;

import java.util.\*;

class TreeNode {

int val;

TreeNode left;

TreeNode right;

TreeNode() {}

TreeNode(int val) { this.val = val; }

TreeNode(int val, TreeNode left, TreeNode right) {

this.val = val;

this.left = left;

this.right = right;

}

}

public class Main

{

static void toDo(TreeNode node, List<Integer> list){

// TODO - Тут нужно дописать Ваш код <<<<<<<<<<<<<<<<

}

static void print(List<Integer> list) {

String output = "";

for(int x : list) {

output += x + " ";

}

System.out.print(output.trim());

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String terminal = br.readLine();

List<Integer> outputNode = new ArrayList<>();

toDo(getTreeTest(terminal), outputNode);

print(outputNode);

}

private static TreeNode treeTest1() {

TreeNode tree = new TreeNode(1);

tree.left = new TreeNode(2);

tree.right = new TreeNode(3);

tree.right.left = new TreeNode(3);

tree.right.right = new TreeNode(4);

return tree;

}

private static TreeNode getTreeTest(String str) {

switch (str) {

case "1 2 3 null null 3 4":

return treeTest1();

case "1 null 2 3":

return treeTest2();

case "":

return treeTest3();

case "1":

return treeTest4();

}

return null;

}

private static TreeNode treeTest2() {

TreeNode tree = new TreeNode(1);

tree.right = new TreeNode(2);

tree.right.left = new TreeNode(3);

return tree;

}

private static TreeNode treeTest3() {

return null;

}

private static TreeNode treeTest4() {

TreeNode tree = new TreeNode(1);

return tree;

}

}

C#

using System;

using System.Collections.Generic;

public class TreeNode

{

public int val;

public TreeNode left;

public TreeNode right;

public TreeNode(int val = 0, TreeNode left = null, TreeNode right = null)

{

this.val = val;

this.left = left;

this.right = right;

}

}

public class Program

{

public static void ToDo(TreeNode node, List<int> lst)

{

if (node == null)

return;

ToDo(node.left, lst);

ToDo(node.right, lst);

lst.Add(node.val);

}

public static void PrintList(List<int> lst)

{

string output = "";

foreach (int x in lst)

{

output += x.ToString() + " ";

}

Console.WriteLine(output.Trim());

}

public static TreeNode TreeTest1()

{

TreeNode tree = new TreeNode(1);

tree.left = new TreeNode(2);

tree.right = new TreeNode(3);

tree.right.left = new TreeNode(3);

tree.right.right = new TreeNode(4);

return tree;

}

public static TreeNode TreeTest2()

{

TreeNode tree = new TreeNode(1);

tree.right = new TreeNode(2);

tree.right.left = new TreeNode(3);

return tree;

}

public static TreeNode TreeTest3()

{

return null;

}

public static TreeNode TreeTest4()

{

return new TreeNode(1);

}

public static TreeNode GetTreeTest(string s)

{

if (s == "1 2 3 null null 3 4")

return TreeTest1();

else if (s == "1 null 2 3")

return TreeTest2();

else if (s == "")

return TreeTest3();

else if (s == "1")

return TreeTest4();

else

return null;

}

public static void Main(string[] args)

{

string terminal = Console.ReadLine();

List<int> outputNode = new List<int>();

ToDo(GetTreeTest(terminal), outputNode);

PrintList(outputNode);

}

}

C. inorder

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64.0 Мб |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Дана верхушка бинарного дерева. Вам нужно заполнить **List < Integer>**значениями из **TreeNode (node)** в порядке - "*inorder*".

Ограничения:

* Количество Nodes в дереве **TreeNode** в отрезке - [0, 100]
* -100 <= node.val <=100

Формат ввода

*1 2 3 null null 3 4*

Формат вывода

**2 1 3 3 4**

Примечания

Используйте этот подготовленный код и допишите Вашу логику в методе **toDo()**.так же Вы можете использовать другие языки из списка (документация использования: <https://yandex.com/support2/contest/en/examples-stdin-stdout>)

import java.io.\*;

import java.util.\*;

class TreeNode {

int val;

TreeNode left;

TreeNode right;

TreeNode() {}

TreeNode(int val) { this.val = val; }

TreeNode(int val, TreeNode left, TreeNode right) {

this.val = val;

this.left = left;

this.right = right;

}

}

public class Main

{

static void toDo(TreeNode node, List<Integer> list){

// TODO - Тут нужно дописать Ваш код <<<<<<<<<<<<<<<<

}

static void print(List<Integer> list) {

String output = "";

for(int x : list) {

output += x + " ";

}

System.out.print(output.trim());

}

public static void main(String[] args) throws Exception {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String terminal = br.readLine();

List<Integer> outputNode = new ArrayList<>();

toDo(getTreeTest(terminal), outputNode);

print(outputNode);

}

private static TreeNode treeTest1() {

TreeNode tree = new TreeNode(1);

tree.left = new TreeNode(2);

tree.right = new TreeNode(3);

tree.right.left = new TreeNode(3);

tree.right.right = new TreeNode(4);

return tree;

}

private static TreeNode getTreeTest(String str) {

switch (str) {

case "1 2 3 null null 3 4":

return treeTest1();

case "1 null 2 3":

return treeTest2();

case "":

return treeTest3();

case "1":

return treeTest4();

}

return null;

}

private static TreeNode treeTest2() {

TreeNode tree = new TreeNode(1);

tree.right = new TreeNode(2);

tree.right.left = new TreeNode(3);

return tree;

}

private static TreeNode treeTest3() {

return null;

}

private static TreeNode treeTest4() {

TreeNode tree = new TreeNode(1);

return tree;

}

}

C#

using System;

using System.Collections.Generic;

public class TreeNode

{

public int val;

public TreeNode left;

public TreeNode right;

public TreeNode(int val = 0, TreeNode left = null, TreeNode right = null)

{

this.val = val;

this.left = left;

this.right = right;

}

}

public class Program

{

public static void ToDo(TreeNode node, List<int> lst)

{

if (node == null)

return;

ToDo(node.left, lst);

lst.Add(node.val);

ToDo(node.right, lst);

}

public static void PrintList(List<int> lst)

{

string output = "";

foreach (int x in lst)

{

output += x.ToString() + " ";

}

Console.WriteLine(output.Trim());

}

public static TreeNode TreeTest1()

{

TreeNode tree = new TreeNode(1);

tree.left = new TreeNode(2);

tree.right = new TreeNode(3);

tree.right.left = new TreeNode(3);

tree.right.right = new TreeNode(4);

return tree;

}

public static TreeNode TreeTest2()

{

TreeNode tree = new TreeNode(1);

tree.right = new TreeNode(2);

tree.right.left = new TreeNode(3);

return tree;

}

public static TreeNode TreeTest3()

{

return null;

}

public static TreeNode TreeTest4()

{

return new TreeNode(1);

}

public static TreeNode GetTreeTest(string s)

{

if (s == "1 2 3 null null 3 4")

return TreeTest1();

else if (s == "1 null 2 3")

return TreeTest2();

else if (s == "")

return TreeTest3();

else if (s == "1")

return TreeTest4();

else

return null;

}

public static void Main(string[] args)

{

string terminal = Console.ReadLine();

List<int> outputNode = new List<int>();

ToDo(GetTreeTest(terminal), outputNode);

PrintList(outputNode);

}

}

A. Показ всех nodes на каждой глубине начиная с низу вверх

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64.0 Мб |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Вам дано дерево (**node**) нужно вернуть **list** в котором хранятся все **nodes** каждого уровня. Но надо показать их в порядке с низу вверх



т.е. рассмотрим дереве выше. Где:

1) Первый уровень - 15, 7

2) Второй уровень - 9, 20

3) Третий уровень - 3

в итоге получим результат

[

[15,7],

[9,20],

[3]

]

Ограничения:

* Количество Nodes в дереве **TreeNode** в отрезке - [1, 100]
* -1000 <= node.val <=1000

Формат ввода

3 9 20 null null 15 7

Формат вывода

[[3],[9,20],[15,7]]

B. Показ всех nodes на каждой глубине в виде zigzag

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64.0 Мб |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Вам дано дерево (**node**) нужно вернуть **list** в котором хранятся все **nodes** каждого уровня.



т.е. рассмотрим дереве выше. Где:

1) Первый уровень - 3

2) Второй уровень - 9, 20

3) Третий уровень - 15, 7

в итоге получим результат

[

[3],

[20,9],

[15,7]

]

Ограничения:

* Количество Nodes в дереве **TreeNode** в отрезке - [1, 100]
* -1000 <= node.val <=1000

Формат ввода

3 9 20 null null 15 7

C. Показ всех nodes на каждой глубине

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64.0 Мб |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Вам дано дерево (**node**) нужно вернуть **list** в котором хранятся все **nodes** каждого уровня.



т.е. рассмотрим дереве выше. Где:

1) Первый уровень - 3

2) Второй уровень - 9, 20

3) Третий уровень - 15, 7

в итоге получим результат

[

[3],

[9,20],

[15,7]

]

Ограничения:

* Количество Nodes в дереве **TreeNode** в отрезке - [1, 100]
* -1000 <= node.val <=1000

Формат ввода

3 9 20 null null 15 7

Формат вывода

[[3],[9,20],[15,7]]

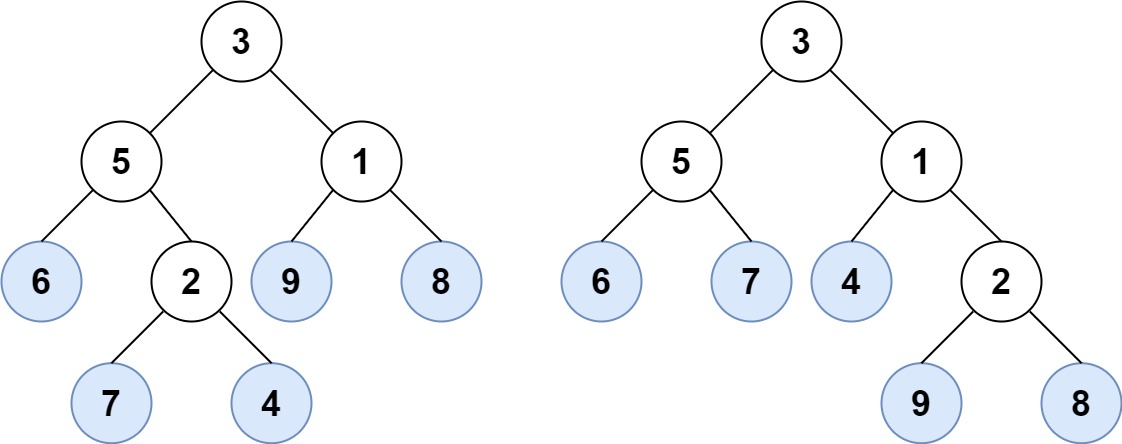
# D. Одинаковые листья дерева

|  |  |
| --- | --- |
| Ограничение времени | 1 секунда |
| Ограничение памяти | 64.0 Мб |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Вам даны два дерева **node1** и **node2**. Вам нужно выявить являются ли у деревьев конечные **nodes** одинаковыми. Конечными **node**-ами являются **node** у которого нет дочерних элементов.

Важно: Порядок листьев ВАЖНО !

## Формат ввода



3 5 1 6 2 9 8 null null 7 4

3 5 1 6 7 4 2 null null null null null null 9 8

## Формат вывода

true