Антон Макеев

Яндекс. Тренировки по алгоритмам 2.0, занятие 8 (B)

9 окт 2021, 12:43:04
старт: 23 сен 2021, 12:00:00
начало: 23 сен 2021, 12:00:00

A. Бинарное дерево (вставка, поиск, обход)

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

Напишите программу, которая будет реализовывать действия в бинарном дереве поиска «вставить» и «найти» (по значению). Программа должна обрабатывать запросы трёх видов:
ADD n — если указанного числа еще нет в дереве, вставлять его и выводить слово «DONE», если уже есть — оставлять дерево как было и выводить слово «ALREADY».

SEARCH — следует выводить слово «YES» (если значение найдено в дереве) или слово «NO» (если не найдено). Дерево при этом не меняется.

PRINTTREE — выводить все дерево, обязательно используя алгоритм, указанный в формате вывода результатов.

Формат ввода

В каждой строке входных данных записан один из запросов ADD n или SEARCH n или PRINTTREE. Гарантируется, что запросы PRINTTREE будут вызываться только в моменты, когда дерево не пустое. Общее количество запросов не превышает 1000, из них не более 20 запросов PRINTTREE.

Формат вывода

Для каждого запроса выводите ответ на него. Для запросов ADD и SEARCH — соответствующее слово в отдельной строке. На запрос PRINTTREE надо выводить дерево, обязательно согласно такому алгоритму:
1) Распечатать левое поддерево

2) Вывести количество точек, равное глубине узла

3) Вывести значение ключа

4) Распечатать правое поддерево

Пример

Ввод	Вывод
ADD 2 ADD 3 ADD 2 SEARCH 2 ADD 5 PRINTTREE SEARCH 7	DONE DONE ALREADY YES DONE 2 .3

```
..5
NO
```

Язык Swift 5.3

Набрать здесь | Отправить файл

```
import Foundation

class TreeNode {
    var key: Int?
    var left: TreeNode?
    var right: TreeNode?

    init(key: Int) {
        self.key = key
    }

    init() {}

    @discardableResult func add(_ key: Int) -> Bool {
        if let nodeKey = self.key {
           if nodeKey == key {
                return false
           } else if key < nodeKey {
               if let leftNode = left {
                   return leftNode.add(key)
               } else {
                   let newNode = TreeNode(key: key)
                   left = newNode
                   return true
               }
           } else {
               if let rightNode = right {
                   return rightNode.add(key)
               } else {
                   let newNode = TreeNode(key: key)
                   right = newNode
                   return true
               }
            }
        } else {
            self.key = key
            return true
        }
```

Отправить

Предыдущая Следующая