Алгоритм поиска наибольшего и наименьшего значения в дереве, представленном в виде списка поддеревьев

Автор: Малышев Антон Алексеевич, 143 группа

Научный руководитель: ст.пр. С.В. Григорьев

21 декабря 2015г.

Введение

- Дерево, представленное в виде списка поддеревьев (тип Tree):

```
type MyList<'a> =
    | Empty
    | Cons of 'a * MyList<'a>

type Tree =
    | Node of int * MyList<Tree>
    | Leaf of int
```

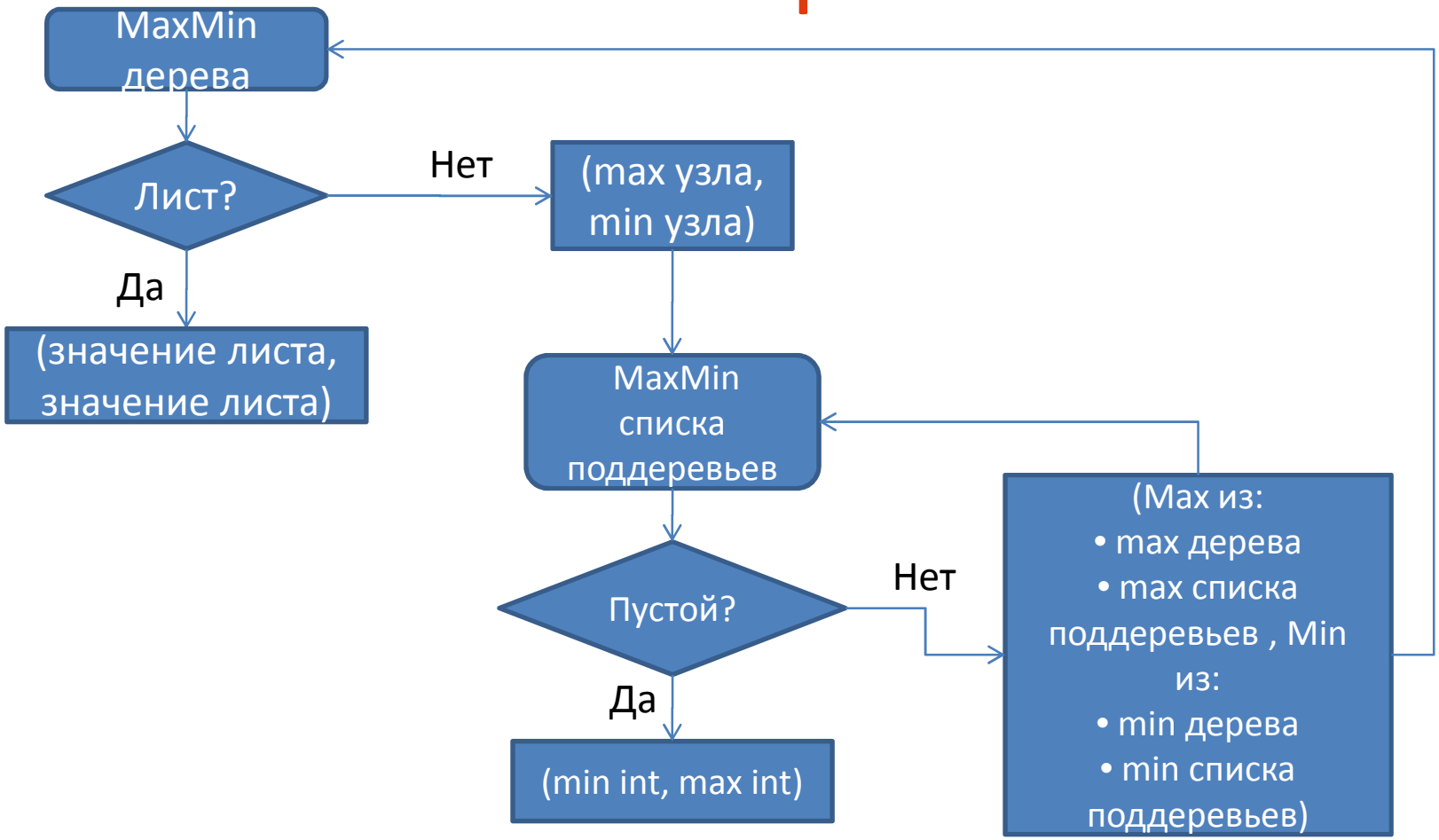
Постановка задачи

Целью работы является разработка алгоритма поиска наибольшего и наименьшего значения в дереве, представленном в виде списка поддеревьев

Задачи:

- Разработать алгоритм нахождения наибольшего и наименьшего значения в сконструированном типе Tree
- Реализовать алгоритм нахождения наибольшего и наименьшего значения в сконструированном типе Tree
- Протестировать реализованный алгоритм

Алгоритм

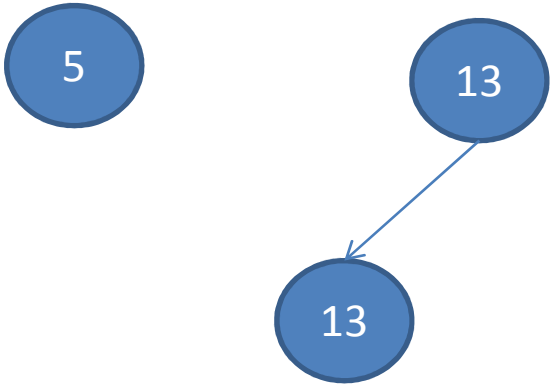


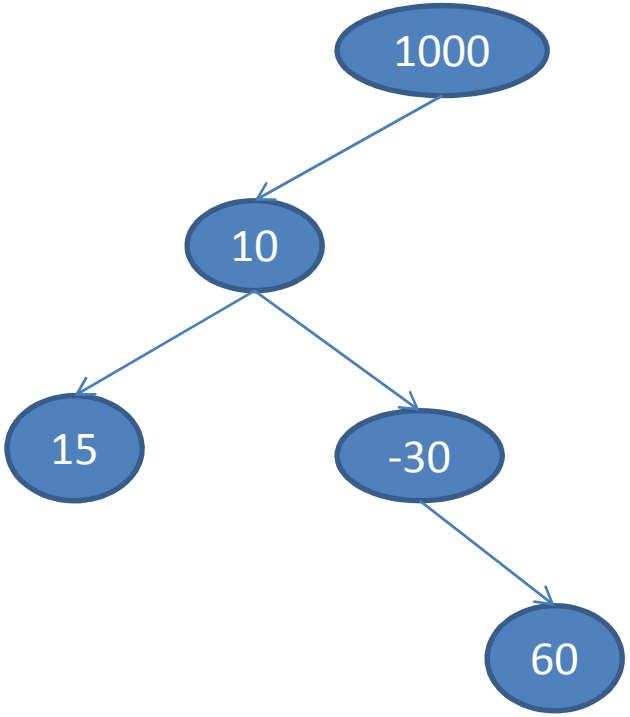
Минимум узла – это минимальное значение из двух: элемента узла и минимума списка поддеревьев (Максимум аналогично)

Тестирование

Было проведено три теста:

- Тест на дереве из одного элемента
- Тест на дереве из двух одинаковых элементов
- Тест на «сложном» дереве





Результаты

- Разработан алгоритм нахождения наибольшего и наименьшего значения в сконструированном типе Tree
- Реализован алгоритм нахождения наибольшего и наименьшего значения в сконструированном типе Tree
- Реализованный алгоритм протестирован

Код программы

```
type MyList<'a> =
    | Empty
    | Cons of 'a * MyList<'a>

type Tree =
    | Node of int * MyList<Tree>
    | Leaf of int

let rec main (tree: Tree) =

    let rec maxminList (list: MyList<Tree>) =
```

```
        match list with
        | Cons(tree, subList) ->
            let (maximal, minimal) = maxminList subList
            (max (fst (main tree)) maximal, min (snd (main tree)) minimal)
        | Empty -> (System.Int32.MinValue, System.Int32.MaxValue)


match tree with
| Leaf(a) -> (a, a)
| Node(a, list) ->
    let (maximal, minimal) = maxminList  list
    (max a maximal, min a minimal)
```

Код тестов

```
[<Test>]
let ``Тест сложного дерева`` () =
  let result = main (Node(1000, Cons(Node(10, Cons(Leaf 15, Empty)), Cons(Leaf -30,
      Cons(Leaf 60, Empty)))))
  printfn "%A" result
  Assert.AreEqual((1000, -30), result)

[<Test>]
let ``Тест дерева из двух одинаковых элементов`` () =
  let result = main (Node(13, Cons(Leaf 13, Empty)))
  printfn "%A" result
  Assert.AreEqual((13, 13), result)

[<Test>]
let ``Тест дерева из одного элемента`` () =
  let result = main (Leaf 5)
  printfn "%A" result
  Assert.AreEqual((5, 5), result)
```