Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Сибирский государственный университет
телекоммуникаций и информатики»

Кафедра ВС

Лабораторная работа №3
Тема: «Генерация случайного дерева»

Выполнил:
студент группы МГ-172
Суходоева Н.Н.

Проверил:

д.т.н. Родионов А.С.

Новосибирск, 2018

Задание: Реализовать генерацию случайного дерева с заданными ограничениями:

1. Ограничение на количество потомком узла;
2. Ограничение на количество уровней в дереве.

Решение:

```
import random as r

class Node(object):
    """Узел дерева
    root - номер узла
    level - уровень в дереве
    children - список дочерних узлов"""
    def __init__(self,  root, level):
        self.root = root
        self.level = level
        self.children = []

    def __str__(self):
        tmp = str(self.root)
        return tmp

    def countCh(self):
        return len(self.children)

class Tree(object):
    """Описание дерева
    node - корень
    children - дочерние узлы
    """
    def __init__(self, node, root=None):
        if root is None:
            self.node = Node(node,0)
        else:
            self.node = Node(node,root)

    def __repr__(self):
        return str(self.node.root)
    def __str__(self):
        return str(self.node.root)

    def addChild(self, root, node):
        tmp = self.find(root)
        if tmp is not None:
```

```
            print("addChild", tmp, type(tmp), "add", node)
            new = Tree(node,tmp.level+1)
            tmp.children.append(new)

    def printTree(self, root, lv = 0):
        tmp = self.node
        print("---"*lv, tmp.root, "lv", tmp.level)
        if tmp.children is not []:
            for i in tmp.children:
                i.printTree(i, lv+1)

    def find(self, root):
        tmp = self.node
        print("find", root, type(root), "->", tmp)
        if int(str(root),10) == int(str(tmp.root),10):
            return tmp
        else:
            print("find rek in", tmp.children)
            for i in tmp.children:
                r = i.find(root)
                if r is not None:
                    return r
                else:
                    continue

    def countChild(self, root=None):
        tmp = self.node

        if root is None:
            print("Vertex",str(tmp.root),"have",tmp.countCh(),"child")
            if tmp.children is not []:
                for i in tmp.children:
                    i.countChild()

        if root is not None:
            if str(tmp.root) == str(root):
                return tmp.countCh()
            else:
                if tmp.children is not []:
                    res = 0
                    for i in tmp.children:
                        res += i.countChild(root)
                    return res

def randTree(n, countNode = None, countLevel = None):
```

```
    """Случайное дерево
    Аргументы:
    n - количество элементов дерева
    countNode - ограничение на количество дочерних
    countLevel - ограничение на количество уровней
    Возвращает:
    head - голова дерева
    tr - случайное дерево
    """
    left = list(range(1,n+1))
    right = []

    head = r.choice(left)
    tr = Tree(head)
    right.append(left.pop(left.index(head)))

    #print('left =',left, ' right =',right,'\n')
    while left:
        print("Подготовка")
        tmp = r.choice(left)
        ind = r.choice(right)
        if (countNode is None or tr.countChild(ind) < countNode) \
        and (countLevel is None or tr.find(ind).level < countLevel):
            print("lv",tr.find(ind).level)
            right.append(left.pop(left.index(tmp)))
        else:
            print("right",right)
            right.pop(right.index(ind))
            continue
        print("Добавление")
        print("add vertex", ind, tmp)
        tr.addChild(ind, tmp)

        print()
        tr.printTree(head)
        print()
        #print('left =',left, ' right =',right,'ind =',ind,'\n')
        #tr.printTree (head)
    return head,tr

head, tr = randTree(10, countLevel = 3)
tr.printTree (head)
print()
```