МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

АЛГОРИТМ ДЕРЕВА БИНАРНОГО ПОИСКА

ЛАБОРАТОРНАЯ РАБОТА

студента 3 курса 331 группы
специальности 100501 — Компьютерная безопасность
факультета КНиИТ
Окунькова Сергея Викторовича

Проверил
доцент _____ А. Н. Гамова

Саратов 2022

СОДЕРЖАНИЕ

1 Описание алгоритма

Бинарное дерево — это иерархическая структура данных, в которой каждый узел имеет значение (оно же является в данном случае и ключом) и ссылки на левого и правого потомка. Узел, находящийся на самом верхнем уровне (не являющийся чьим либо потомком) называется корнем. Узлы, не имеющие потомков (оба потомка которых равны NULL) называются листьями.статком данного алгоритма.

Бинарное дерево поиска — это бинарное дерево, обладающее дополнительными свойствами: значение левого потомка меньше значения родителя, а значение правого потомка больше значения родителя для каждого узла дерева. То есть, данные в бинарном дереве поиска хранятся в отсортированном виде. При каждой операции вставки нового или удаления существующего узла отсортированный порядок дерева сохраняется. При поиске элемента сравнивается искомое значение с корнем. Если искомое больше корня, то поиск продолжается в правом потомке корня, если меньше, то в левом, если равно, то значение найдено и поиск прекращается.

2 Эффективность алгоритма

Так как для добавления элемента в дерево бинарного поиска нам не нужно проходить все дерево, а нужно только найти элемент, для которого мы определим его как потомка, то операция вставки в бинарное дерево поиска имеет сложность $O(\log n)$, где n - размер дерева.

Чтобы создать дерево из массива, нужно n раз вызвать операцию вставки в дерево. Из всего выше сказанного очевидно, что вычислительная сложность создания дерева бинарного поиска определяется как $O(n \log n)$.

За счет свойств дерева бинарного поиска сам поиск работает по такому принципу, что на каждом шаге мы спускаемся на один уровень дерева вниз. Очевидно что в худшем случае рекурсия пройдет по всем уровням, колличество которых равняется $\log_2 n$. Отсюда следует, что вычислительная сложность поиска определяется как $O(\log n)$.

3 Реализация

```
class Node(object):
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

class Tree(object):
    def __init__(self):
        self.root = None

def insert(v, x):
    if v is None:
        return Node(x)
    if x < v.key:
        v.left = insert(v.left, x)
    elif x > v.key:
        v.right = insert(v.right, x)
    # v.key == x
    return v

def search(v, x):
    if v is None or v.key == x:
        return v
    elif x < v.key:
        return search(v.left, x)
    else:  # x > v.key
        return search(v.right, x)

def find_min(v):
    if v.left is not None:
        return find_min(v.left)
    else:
        return v

def delete(v, x):
    if v is None:
        return None

    if x < v.key:
        v.left = delete(v.left, x)
```

```
            return v
        elif x > v.key:
            v.right = delete(v.right, x)
            return v

        # v.key == x
        if v.left is None:
            return v.right
        elif v.right is None:
            return v.left
        else:
            # both subtrees are present
            min_key = find_min(v.right).key
            v.key = min_key
            v.right = delete(v.right, min_key)
            return v

tree = Tree()
print('Enter your array: ')
a = list(map(int, input().split()))
for x in a:
    tree.root = insert(tree.root, x)

print('Enter the element you want to find')
find = int(input())
if search(tree.root, find) == None:
    print('This element does not exist in the given tree')
else:
    print('The element is part of the given tree')
```

4 Тестирование программы

```
sokunkov@C11310 complexity_calculation % python3 lab2.py
Enter your array:
-1 7 0 -100 12 4 2 9 1000
Enter the element you want to find
-1
The element is part of the given tree
sokunkov@C11310 complexity_calculation % python3 lab2.py
Enter your array:
-1 7 0 -100 12 4 2 9 1000
Enter the element you want to find
10
This element does not exist in the given tree
sokunkov@C11310 complexity_calculation %
```

Рисунок 1 – Тест1

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Скиена Стивен "Алгоритмы. Руководство по разработке 2018 год. Яз. рус.

2 Николаус Вирт "Алгоритмы и структуры данных 2008 год. Яз. рус.