Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«Алгоритмы и структуры данных: Алгоритмы поиска.»

Проверил:
Сентерев Ю.А. _____
«_____» _____ 201_г.

Выполнил:
Студент группы P3255
Кабардинов Д. В. _____

Оценка _____

Санкт-Петербург

2019

Задание

5. Вывести на экран все числа упорядоченного массива A кратные 4 (4,8,...) с помощью помощью линейного, бинарного и индексно-последовательного поиска.

Цель работы:

- исследовать и изучить основные процедуры, используемые при работе с бинарными (двоичными) деревьями;
- овладеть умениями и навыками написания программ по исследованию бинарных деревьев на языке программирования Python.

Ход выполнения работы

1. Последовательный поиск

Реализация алгоритма последовательного поиска на языке Python:

```
def linearSearch(numbers, search):
        for number in numbers:
                if number == search:
                        return number
        return None
```

Функция, использующая последовательный поиск и позволяющая найти все числа массива кратные 4:

```
def findAllMatchesUsingLinearSearch(numbers, searches):
        matches = []
        for search in searches:
                match = linearSearch(numbers, search)
                if (match != None):
                        matches.append(match)
        return matches
```

Проверка результата:

```
def testLinearSaerch(self):
                sortedNumbers = [1, 3, 5, 9, 11, 16, 24, 28, 35, 40, 89, 100]
                multiplesOfFour = [16, 24, 28, 40, 100]
                linearSearchResultList = findAllMatchesUsingLinearSearch(sortedNumbers, mul
                print('The result of linear search:')
                for num in linearSearchResultList:
                        print(num)
                self.assertListEqual(linearSearchResultList, multiplesOfFour)
```

Вывод команды

```
dmitrii@kdv:~/projects/TeX$ python3 algorithms\ and\ data\ structures/lab3/search.py
The result of linear search:
16
24
28
40
100
```

2. Бинарный поиск
Реализация алгоритма бинарного поиска на языке Python:

```
def binarySearch(numbers, search):
        left = 0
        right = len(numbers) - 1
        while left < len(numbers) and right >= 0:
                mid = (left + right)//2
                middleNumber = numbers[mid]
                if middleNumber < search:
                        left = mid + 1
                elif middleNumber > search:
                        right = mid
                else: # it's a match
                        return middleNumber
        return None
```

Функция, использующая бинарный поиск и позволяющая найти все числа массива кратные 4:

```
def findAllMatchesUsingBinarySearch(numbers, searches):
        matches = []
        for search in searches:
                match = binarySearch(numbers, search)
                if (match != None):
                        matches.append(match)
        return matches
```

Проверка результата:

```
def testBinarySearch(self):
                sortedNumbers = [1, 3, 5, 9, 11, 16, 24, 28, 35, 40, 89, 100]
                multiplesOfFour = [16, 24, 28, 40, 100]
                binarySearchResultList = findAllMatchesUsingBinarySearch(sortedNumbers, mul
                print('The result of binary search:')
                for num in binarySearchResultList:
                        print(num)
                self.assertListEqual(binarySearchResultList, multiplesOfFour)
```

Вывод команды

```
dmitrii@kdv:~/projects/TeX$ python3 algorithms\ and\ data\ structures/lab3/search.py
The result of binary search:
16
24
28
40
100
```

3. Индексно-последовательный поиск
Реализация алгоритма индексно-последовательного поиска на языке Python:

```
def linearIndexedSearch(numbers, index, search):
        start = 0
        end = len(numbers) - 1
        for entry in index:
                if entry['val'] < search:
                        start = entry['ind']
                if entry['val'] > search:
                        end = entry['ind']
                        break
                if entry['val'] == search:
                        return search
        for i in range(start, end + 1):
                if numbers[i] == search:
                        return search
        return None
```

Функция, использующая индексно-последовательный поиск и позволяющая найти все числа массива кратные 4:

```
def findAllMatchesUsingLinearIndexedSearch(numbers, index, searches):
        matches = []
        for search in searches:
                match = linearIndexedSearch(numbers, index, search)
                if (match != None):
                        matches.append(match)
        return matches
```

Проверка результата:

```
def testLinearIndexedSearch(self):
                index = [
                        {'val': 1,          'ind': 0},
                        {'val': 11, 'ind': 4},
                        {'val': 35, 'ind': 8}
                ]
                sortedNumbers = [1, 3, 5, 9, 11, 16, 24, 28, 35, 40, 89, 100]
                multiplesOfFour = [16, 24, 28, 40, 100]
                linearIndexedSearchResult = findAllMatchesUsingLinearIndexedSearch(
                sortedNumbers, index, multiplesOfFour)
                print('The result of linear indexed search:')
                for num in linearIndexedSearchResult:
                        print(num)
                self.assertListEqual(linearIndexedSearchResult, multiplesOfFour)
```

Вывод команды

```
dmitrii@kdv:~/projects/TeX$ python3 algorithms\ and\ data\ structures/lab3/search.py
The result of binary search:
16
24
28
40
100
```

Выводы
Мне удалось выполнить поставленную задачу двумя 3 способами:
1. Последовательный поиск
2. Бинарный поиск
3. Индексно-последовательный поиск
Таким образом, цель работы достигнута.

Список используемой литературы:

1. `https://docs.python.org/3/tutorial/introduction.html#lists` - Lists
2. `https://docs.python.org/3.5/tutorial/index.html` - The Python Tutorial
3. Стивен Скиена - Алгоритмы. Руководство по разработке
4. Никлаус Вирт - Алгоритмы и структуры данных
5. Томас Кормен - Алгоритмы. Построение и анализ
6. `https://www.pythoncentral.io/singly-linked-list-insert-node/` - Python Data Structures Tutorial