

Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного
обеспечения и администрирования
информационных систем
Направление подготовки
математическое обеспечение и
администрирование
информационных систем
Форма обучения очная

Отчет

по лабораторной работе №3
«Сравнительный анализ алгоритмов поиска»

Выполнил:

студент группы 213

Тихонов Е.Е.

Проверил:

профессор кафедры ПОиАИС

Кудинов В.А.

Курск, 2021

Цель работы: изучение алгоритмов поиска элемента в массиве и закрепление навыков в проведении сравнительного анализа алгоритмов.

Задание

1. Изучить алгоритмы поиска:
 - 1) в неупорядоченном массиве:
 - а) линейный;
 - б) быстрый линейный;
 - 2) в упорядоченном массиве:
 - а) быстрый линейный;
 - б) бинарный;
 - в) блочный.
2. Разработать и программно реализовать средство для проведения экспериментов по определению временных характеристик алгоритмов поиска.
3. Провести эксперименты по определению временных характеристик алгоритмов поиска. Результаты экспериментов представить в виде таблиц 1.1 и 1.2. Клетки таблицы 1.1 содержат максимальное количество операций сравнения при выполнении алгоритма поиска, а клетки таблицы 1.2 – среднее число операций сравнения.
4. Построить графики зависимости количества операций сравнения от количества элементов в массиве.
5. Определить аналитическое выражение функции зависимости количества операций сравнения от количества элементов в массиве.
6. Определить порядок функций временной сложности алгоритмов поиска.

Сравнительный анализ алгоритмов

Временные затраты алгоритмов показаны в таблице 1.1 и 1.2.

Таблица 1.1

Максимальное количество операций сравнения

Алгоритмы поиска (см.задание)	Количество элементов в массиве								
	50	100	150	200	250	300	350	400	450
	Количество операций сравнения								
1.а	50	100	150	200	250	300	350	400	450
1.б	51	101	151	201	251	301	351	401	451
2.а	51	101	151	201	251	301	351	401	451
2.б	5	6	7	7	7	8	8	8	8
2.в	5	6	7	7	7	8	8	8	8

Таблица 1.2

Среднее количество операций сравнения

Алгоритмы поиска (см.задание)	Количество элементов в массиве								
	50	100	150	200	250	300	350	400	450
	Количество операций сравнения								
1.а	50	100	150	200	250	300	350	400	450
1.б	51	101	151	201	251	301	351	401	451
2.а	25	50	75	100	125	150	175	200	225
2.б	2	3	3	4	4	5	5	6	6
2.в	2	3	3	4	4	5	5	6	6

Общая оценка сложности алгоритмов показана в таблице 2.

Таблица 2

Поиск	Сложность
Линейный	N
Быстрый линейный	N
Бинарный	$\log_2 N$
Блочный	$X+N/X$ (X-количество блоков)

Текст программы

Текст файла linear_search.py

```
def linear_search(arr, elem, timer):  
    enter_inds = []  
    for i in range(len(arr)):  
        timer.inc()  
        if arr[i] == elem:  
            enter_inds.append(i)  
    return enter_inds
```

Текст файла quick_linear_search.py

```
def quick_linear_search(arr, elem, timer, is_sorted = False):  
    enter_inds = []  
    arr.append(elem)  
    i = 0  
    while(True):  
        if is_sorted and arr[i] > elem:  
            break  
        timer.inc()  
        if arr[i] == elem:  
            if i == len(arr)-1:  
                break  
            enter_inds.append(i)
```

```
        i+=1
    arr.pop()
    return enter_inds
```

Текст файла binary_search.py

```
def binary_search(arr, elem, timer):
    low = 0
    high = len(arr) - 1
    mid = int(high/2)

    while arr[mid] != elem and low < high:
        timer.inc()
        if elem > arr[mid]:
            low = mid+1
        else:
            high = mid-1
        mid = int((low+high)/2)

    if low > high:
        return None
    else:
        return mid
```

Текст файла block_search.py

```
def block_search(arr, elem, timer, block_count = 4):
    block_ends = []
    block_len = int(len(arr)/block_count)
    for i in range(len(arr)-1, 0, -block_len):
        block_ends.insert(0,i)

    block_start = 0
    for block_end in block_ends:
        timer.inc()
        if elem == arr[block_end]:
            return block_end
        if(elem < arr[block_end]):
            #linear search
            for i in range(block_start, block_end+1):
                timer.inc()
                if elem == arr[i]:
```

```
        return i

    block_start = block_end+1
    return None
```

Текст файла timer.py

```
class timer:
    def __init__(self):
        self.arr_len = 0
        self.iters = 0

    def set(self, arr):
        self.iters = 0
        self.arr_len = len(arr)

    def inc(self):
        self.iters+=1

    def get(self):
        print("len/iters = " + str(self.arr_len) + "/" + str(self.iters))
```

Текст файла main.py

```
from linear_search import linear_search
from quick_linear_search import quick_linear_search
from binary_search import binary_search
from block_search import block_search

from timer import timer
import random

def rnd_arr(len, min=-100, max=100, step=1):
    arr = []
    for i in range(len):
        arr.append(random.randrange(min, max, step))
    return arr

def srt_arr(min=1, max=100, step=1):
    arr = []
    for i in range(min, max+1, step):
        arr.append(i)
    return arr

def main():
```

```
tmr = timer()

arr_len = 100

no_sort_arr = rnd_arr(len=arr_len, min=1, max=9)
print("random arr")
print(no_sort_arr)

print("linear search")
tmr.set(no_sort_arr)
idxs = linear_search(no_sort_arr, 6, tmr)
print(idx)
tmr.get()

print("quick linear search(random array)")
tmr.set(no_sort_arr)
idxs = quick_linear_search(no_sort_arr, 6, tmr)
print(idx)
tmr.get()

sort_arr = srt_arr(max=arr_len)
print("sort arr")
print(sort_arr)

print("quick linear search(sorted array)")
tmr.set(sort_arr)
idxs = quick_linear_search(sort_arr, 6, tmr, is_sorted=True)
print(idx)
tmr.get()

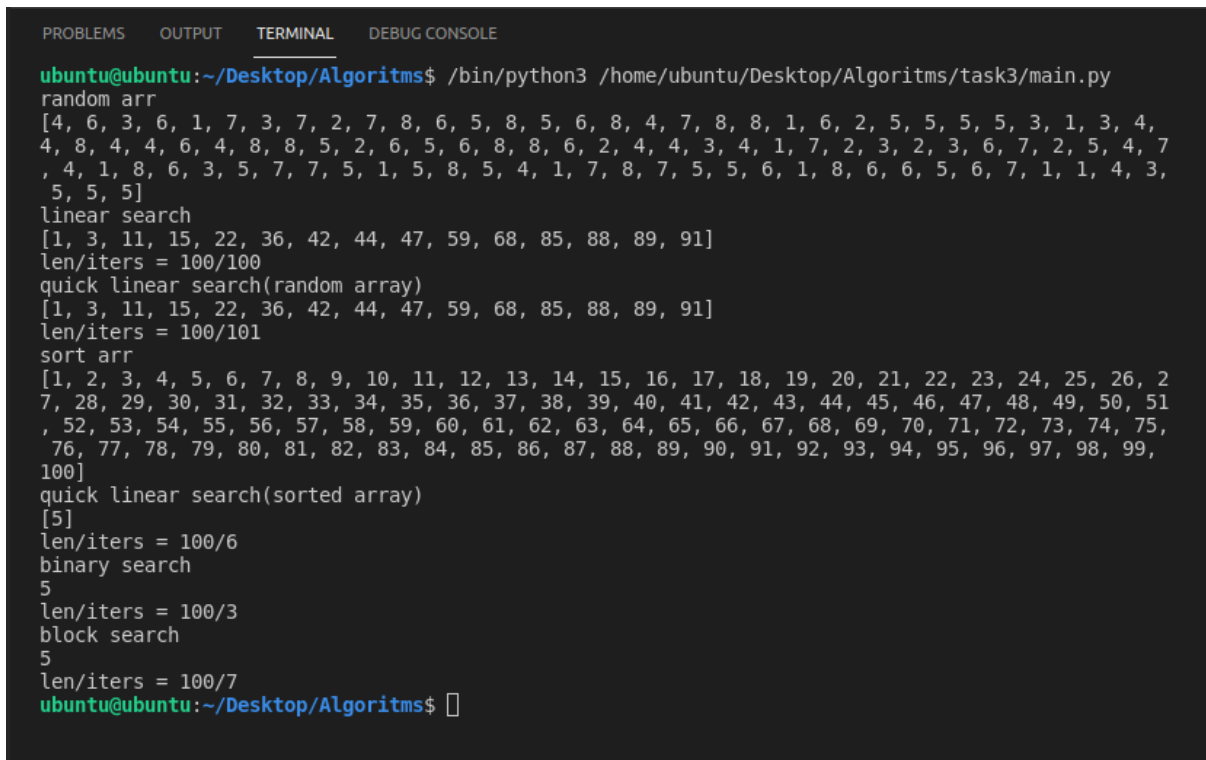
print("binary search")
tmr.set(sort_arr)
idxs = binary_search(sort_arr, 6, tmr)
print(idx)
tmr.get()

print("block search")
tmr.set(sort_arr)
idxs = block_search(sort_arr, 6, tmr)
print(idx)
tmr.get()

main()
```

Тест программы

Вывод программы представлен на рисунке 1.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
ubuntu@ubuntu:~/Desktop/Algoritms$ /bin/python3 /home/ubuntu/Desktop/Algoritms/task3/main.py
random arr
[4, 6, 3, 6, 1, 7, 3, 7, 2, 7, 8, 6, 5, 8, 5, 6, 8, 4, 7, 8, 8, 1, 6, 2, 5, 5, 5, 5, 3, 1, 3, 4,
4, 8, 4, 4, 6, 4, 8, 8, 5, 2, 6, 5, 6, 8, 8, 6, 2, 4, 4, 3, 4, 1, 7, 2, 3, 2, 3, 6, 7, 2, 5, 4, 7
, 4, 1, 8, 6, 3, 5, 7, 7, 5, 1, 5, 8, 5, 4, 1, 7, 8, 7, 5, 5, 6, 1, 8, 6, 6, 5, 6, 7, 1, 1, 4, 3,
5, 5, 5]
linear search
[1, 3, 11, 15, 22, 36, 42, 44, 47, 59, 68, 85, 88, 89, 91]
len/iters = 100/100
quick linear search(random array)
[1, 3, 11, 15, 22, 36, 42, 44, 47, 59, 68, 85, 88, 89, 91]
len/iters = 100/101
sort arr
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 2
7, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51
, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75,
76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99,
100]
quick linear search(sorted array)
[5]
len/iters = 100/6
binary search
5
len/iters = 100/3
block search
5
len/iters = 100/7
ubuntu@ubuntu:~/Desktop/Algoritms$
```

Рисунок 1 - Вывод программы.