Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им.И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление:710400 «Программная инженерия»

ОТЧЕТ

По дисциплине: «Алгоритмы и структуры данных»

Лабораторная работа №11

Тема: «Анализ алгоритмов поиска»

Выполнил: студент группы

ПИ(б)-2-19 Улан уулу Нурдин

Проверила: Валеева А. А.

Бишкек – 2020

**Практическое задание**

**I. Ответы на контрольные вопросы**

**1. Перечислите виды поиска.**

Линейный;

Бинарный;

Интерполяционный.

**2. Перечислите особенности последовательного поиска.**

Особенность последовательного поиска, в том, что он простой, и используется почти для любых типов списка.

**3. В каких случаях применяется последовательный поиск?**

При небольших списках, до N = 10 000 000.

**4. Перечислите достоинства и недостатки последовательного поиска.**

**Достоинства:** Не требует сортировки списка;

Не требует дополнительного анализа функции;

Не требует дополнительной оперативной памяти.

**Недостатки:** Занимает много времени;

Проверяет до каждого элемента до искомого элемента

**5. В чем суть двоичного поиска? В чем его особенности?**

Суть двоичного поиска в том, что алгоритм делит список на два, и проверяет в какой части от нынешнего элемента находится искомый элемент. Если искомый элемент меньше нынешнего, то он ищет в диапазоне от нынешнего элемента до первого, и так далее. Иначе если искомый элемент больше нынешнего, то он ищет в диапазоне от нынешнего элемента до последнего, и так далее.

Особенность двоичного поиска в том, что он быстрее от последовательного и совершает log2(N+1) проходов в отличии от последовательного поиска, который в среднем случае проходит (½ \* N).

**6. Соблюдение какого условия необходимо для применения к структуре данных двоичного поиска?**

Список должен быть отсортирован.

**7. Какими достоинствами обладает двоичный поиск?**

Быстрота по отношению к последовательному поиску.

**8. Сравните статистические характеристики последовательного и двоичного поисков.**

Двоичный поиск будет работать быстрее, из – за шаг, которым он проверяет список.

**9. В чем суть метода интерполяционного поиска? В чем его особенности?**

Интерполяционный поиск вместо сравнения каждого элемента с искомым, как при [последовательном поиске](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA), производит предсказание местонахождения элемента: поиск происходит подобно [двоичному поиску](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA), но вместо деления области поиска на две части, интерполирующий поиск производит оценку новой области поиска по расстоянию между ключом и текущим значением элемента. В среднем интерполирующий поиск производит log2(log2(*N*)) операций. Число необходимых операций зависит от равномерности распределения значений среди элементов. В плохом случае интерполяционный поиск может потребовать до O (*N*) операций.

Особенность в интерполяционного поиска в том, что он требует меньше времени и шагов в отличии от двух других поисков.

**II. Выполнить задания**

**1. Провести вычислительные исследования, анализ и оценку алгоритмов поиска (линейный, бинарный, интерполяционный), используя алгоритмы сортировки:**

a. алгоритм вставки;

b. алгоритм Шелла;

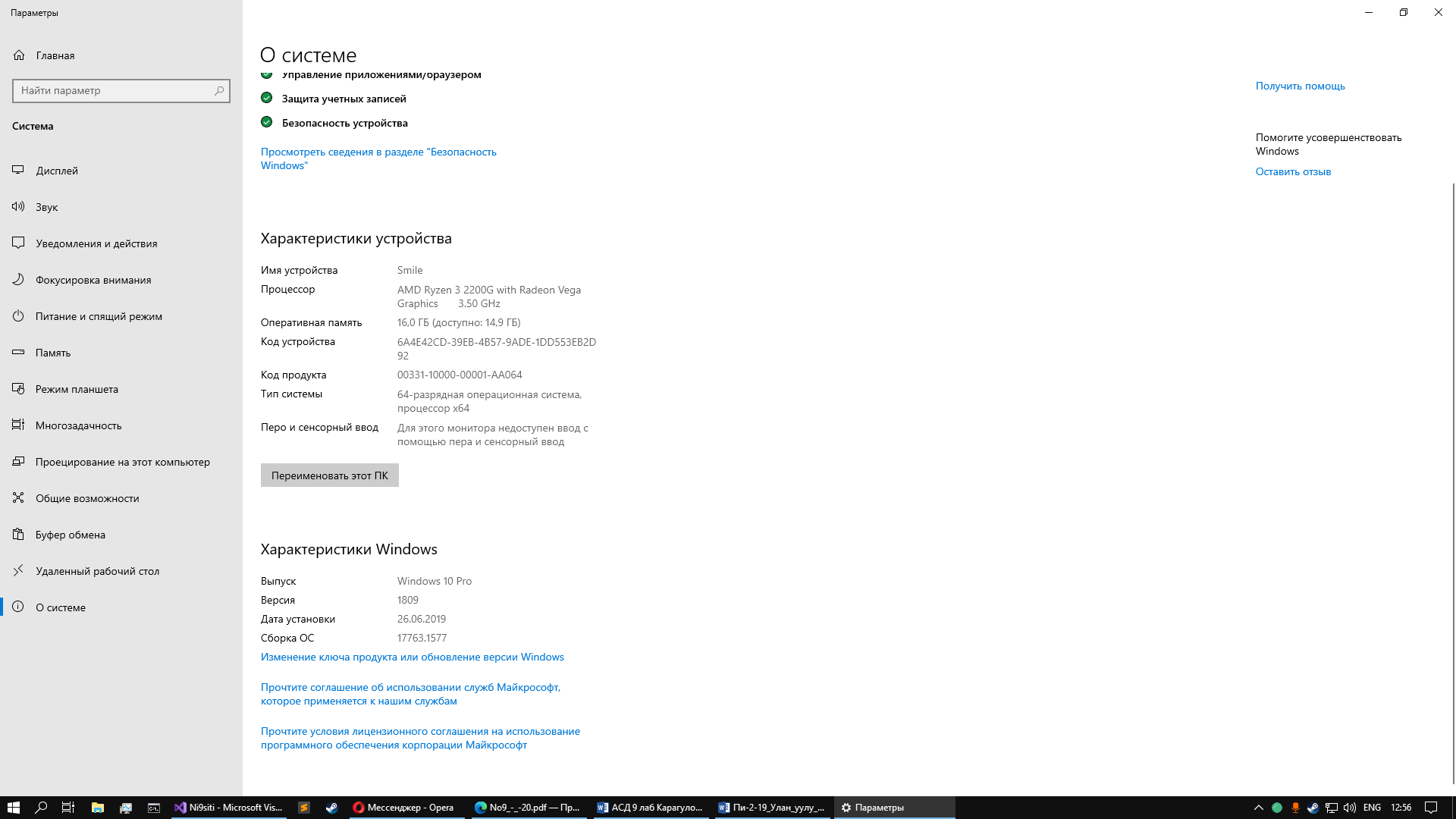
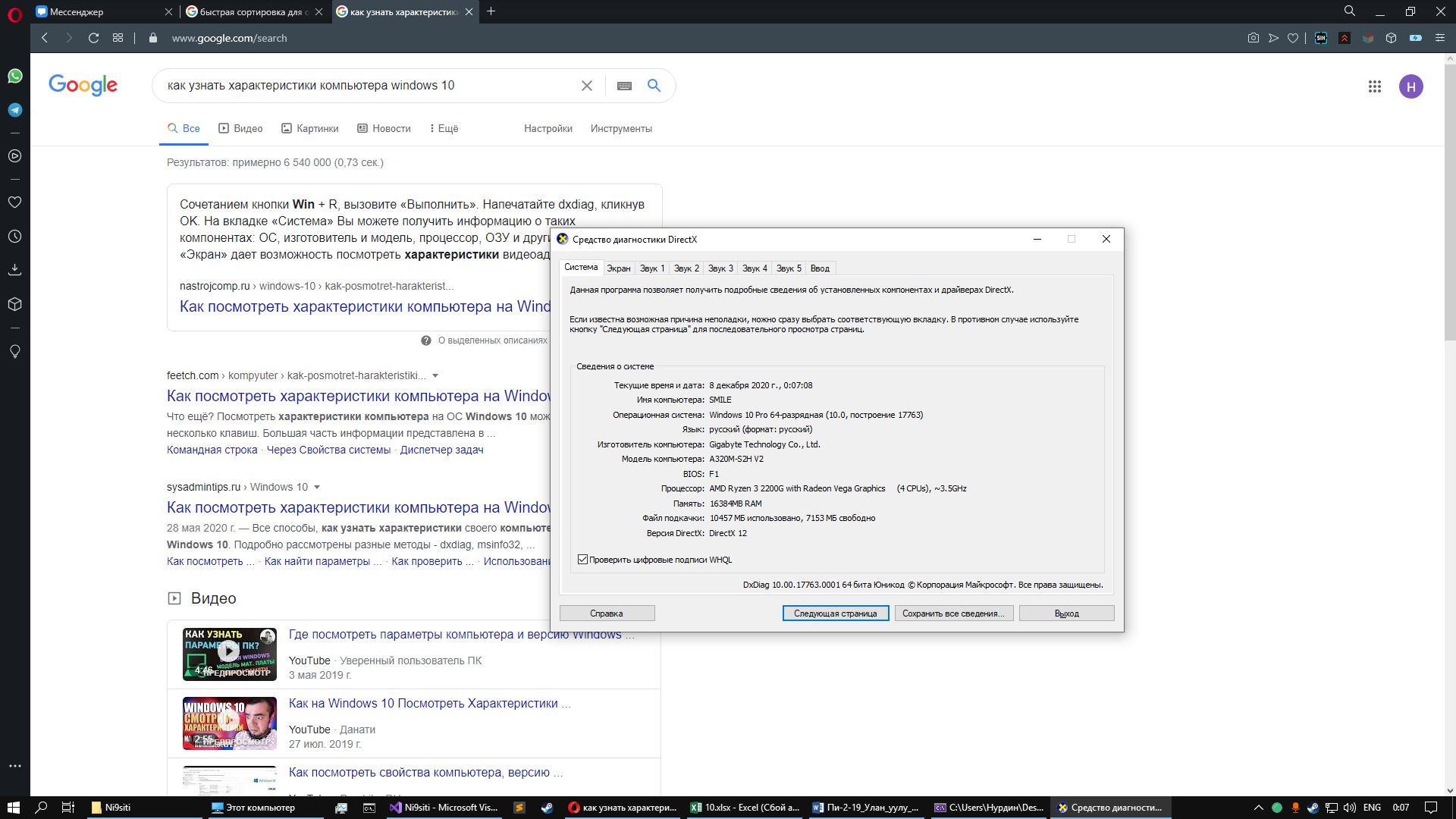
c. алгоритм Хоара;

d. алгоритм сортировки пирамидой.

**2. Провести анализ и оценку алгоритма бинарного дерева поиска**

**3. Провести обобщённый анализ для всех выполненных алгоритмов**

**Указать характеристики ПК:**

**Замеры времени выполнения производятся при помощи функции clock();**

**Поиск элемента, который есть в массиве лучший случай**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Метод поиска | N | 10 | 50 | 100 | 1 000 | 10 000 | 100 000 | 1 000 000 | 10 000 000 | 100 000 000 |
|
| Линейный | Искомый элемент в лучшем случае | 1 | 25 | 23 | 700 | 5 000 | 27 342 | 123 125 | 7 657 545 | 53 465 654 |
| Время в Секунды | 0 | 0 | 0 | 0 | 0 | 0 | 0,03 | 0,09 | 0,3 |
| Кол-во сравнений | 5 | 4 | 91 | 446 | 8 590 | 29 989 | 914 745 | 1 986 557 | 77 833 154 |
| Кол-во шагов | 5 | 4 | 91 | 446 | 8 590 | 29 989 | 914 745 | 1 986 557 | 77 833 154 |
| Бинарный | Искомый элемент в лучшем случае | 1 | 25 | 23 | 700 | 5 000 | 27 342 | 123 125 | 7 657 545 | 53 465 654 |
| Время в Секунды | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Кол-во сравнений | 4 | 6 | 11 | 24 | 14 | 65 | 77 | 82 | 189 |
| Кол-во шагов | 0 | 4 | 10 | 23 | 13 | 64 | 76 | 81 | 189 |
| Интерполяционный | Искомый элемент в лучшем случае | 1 | 25 | 23 | 700 | 5 000 | 27 342 | 123 125 | 7 657 545 | 53 465 654 |
| Время в Секунды | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Кол-во сравнений | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Кол-во шагов | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Поиск элемента, который есть в массиве худший случай**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Метод поиска | N | 10 | 50 | 100 | 1 000 | 10 000 | 100 000 | 1 000 000 | 10 000 000 | 100 000 000 |
|
| Линейный | Искомый элемент в худшем случае | 10 | 50 | 100 | 1 000 | 10 000 | 100 000 | 1 000 000 | 10 000 000 | 100 000 000 |
| Время в Секунды | 0 | 0 | 0 | 0 | 0 | 0 | 0,04 | 0,4 | 0,5 |
| Кол-во сравнений | 9 | 49 | 49 | 999 | 9 999 | 99 999 | 999 999 | 9 999 999 | 99 999 999 |
| Кол-во шагов | 9 | 49 | 49 | 999 | 9 999 | 99 999 | 999 999 | 9 999 999 | 99 999 999 |
| Бинарный | Искомый элемент в худшем случае | 10 | 50 | 100 | 1 000 | 10 000 | 100 000 | 1 000 000 | 10 000 000 | 100 000 000 |
| Время в Секунды | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Кол-во сравнений | 4 | 6 | 7 | 10 | 14 | 17 | 20 | 24 | 27 |
| Кол-во шагов | 3 | 5 | 6 | 9 | 13 | 16 | 19 | 23 | 26 |
| Интерполяционный | Искомый элемент в худшем случае | 10 | 50 | 100 | 1 000 | 10 000 | 100 000 | 1 000 000 | 10 000 000 | 100 000 000 |
| Время в Секунды | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Кол-во сравнений | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Кол-во шагов | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Поиск элемента которого нет в массиве**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Метод поиска | N | 10 | 50 | 100 | 1 000 | 10 000 | 100 000 | 1 000 000 | 10 000 000 | 100 000 000 |
|
| Линейный | Искомый элемент | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| Время в Секунды | 0 | 0 | 0 | 0 | 0 | 0 | 0,04 | 0,041 | 0,41 |
| Кол-во сравнений | 10 | 50 | 100 | 1 000 | 10 000 | 100 000 | 1 000 000 | 10 000 000 | 100 000 000 |
| Кол-во шагов | 10 | 50 | 100 | 1 000 | 10 000 | 100 000 | 1 000 000 | 10 000 000 | 100 000 000 |
| Бинарный | Искомый элемент | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| Время в Секунды | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Кол-во сравнений | 4 | 6 | 7 | 10 | 14 | 17 | 20 | 24 | 27 |
| Кол-во шагов | 4 | 6 | 7 | 10 | 14 | 17 | 20 | 24 | 27 |
| Интерполяционный | Искомый элемент | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| Время в Секунды | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Кол-во сравнений | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Кол-во шагов | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Анализ по поискам**

Сложность алгоритма для **линейного** поиска: **O(N)**.

При малых размерностях массива, **линейный** поиск хорошо работает во всех случаях. В лучшем случае выполняется лишь 1 одно сравнение, в худшем наоборот, количество сравнений равно N, так как искомый элемент - в конце. В среднем случае выполняется N / 2 сравнений, так как в этом случае искомый элемент находится в середине.

Количество сравнений для **Бинарного** поиска: Лучший случай: О(1);

Средний случай: О(log2N);

Худший случай: О(log2N).

Данный метод поиска работает отлично. По сравнению с **линейным** поиском, **бинарный** поиск совершил меньшее количество сравнений, и нашел искомый элемент за короткое время. Единственным недостатком данного метода поиска, по сравнению с **линейным** поиском, является то, что он дает хорошие результаты лишь в отсортированных массивах.

В среднем **интерполирующий** поиск производит log2(log2(*N*)) операций. Число необходимых операций зависит от равномерности распределения значений среди элементов.

**Вывод**

Исходя из результатов поиска, можно сказать, что для неотсортированных данных небольших размеров до N = 10 000 000 подойдет **линейный** поиск, так как он не требует предварительной упорядоченности элементов. Для больших размерностей где N >= 10 000 000 лучше использовать, как и **бинарный**, так и **интерполяционный** поиск.