Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №9
дисциплины «Алгоритмизация»
Вариант___

Выполнил:
Репкин Александр Павлович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных
систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук,
доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Порядок выполнения работы:

1. Создана программа на основе поставленного задания – необходимо проанализировать два типа поиска элементов в массиве в худшем (когда элемента в массиве нет) и среднем (Элемент примерно в середине массива) случаях – линейный и бинарный. Также, если в используемом для выполнения работы языке (Выбран язык Java. В нём есть у Arrays метод binarySearch, которому передаётся массив и требуемый элемент) присутствуют встроенные методы бинарного поиска, то сравнить их с основными двумя типами. Согласно полученным данным, бинарный поиск гораздо лучше линейного. Метод бинарного поиска, встроенный в язык программирования Java, идентичен стандартному алгоритму бинарного поиска, в связи с чем результаты встроенного и самодельного бинарного поисков почти идентичны. Алгоритмы затрачивали: линейный поиск = $\theta(n)$, бинарный поиск (Как встроенный, так и самодельный) = $\theta(\log(n))$

```
static void linear() {
    long startTime = System.nanoTime();
    for (int i = 0; i < needed; i++) {
        if (numbers.get(i) == wanted) {
            break;
        }
    }
    long endTime = System.nanoTime();
    long spent_time = endTime-startTime; // Divide by 1000000 to get milliseconds.
    results_linear.append(spent_time / 1000000).append(",").append((spent_time / 10000) % 100).append(";");
}
```

Рисунок 1. Полученный код линейного поиска.

```
static void binary() {  2 usages
    long startTime = System.nanoTime();
    int first = 0;
    int last = wanted;
    while (first < last) {
        int mid = (first + last) / 2;
        if (wanted == numbers.get(mid)) {
            break;
        } else if (wanted < numbers.get(mid)) {
            last = mid - 1;
        } else {
            first = mid + 1;
        }
    }
    long endTime = System.nanoTime();
    long spent_time = endTime - startTime; // Divide by 1000000 to get milliseconds.
    results_binary.append(spent_time / 1000000).append(",").append((spent_time / 10000) % 100).append(";");
}
```

Рисунок 2. Полученный код бинарного поиска.

```java
static void built_in(){  2 usages
    // В Java есть метод binarySearch() у объекта Arrays.
    Object[] for_search = numbers.toArray();
    long startTime = System.nanoTime();
    int found_item = Arrays.binarySearch(for_search, wanted);
    long endTime = System.nanoTime();
    long spent_time = endTime-startTime; // Divide by 1000000 to get milliseconds.
    results_built_in.append(spent_time / 1000000).append(",").append((spent_time / 10000) % 100).append(";");
}
```

Рисунок 3. Полученный код встроенного бинарного поиска.

Размер массива	Линейный поиск (средний)	Линейный поиск (худший)	Бинарный поиск (средний)	Бинарный поиск (худший)	Встроенный бинарный поиск (средний)	Встроенный бинарный поиск (худший)
1000000	7,13	2,75	0	0	0,2	0,2
1025000	1,8	3,47	0	0	0,3	0,2
1050000	1,27	3,68	0	0	0,5	0,2
1075000	1,5	5,19	0	0	0,5	0,1
1100000	1,58	4,48	0,1	0	0,4	0,4
1125000	1,42	3,33	0	0	0,2	0,8
1150000	1,88	3,72	0	0	0,4	0,4
1175000	2,23	7,72	0,1	0	0,4	0,5
1200000	2,45	4,14	0	0	0,2	0,2
1225000	2,25	3,84	0	0	0,2	0,1
1250000	2,18	5,16	0,1	0,1	0,3	0,2
1275000	2,43	3,99	0	0	0,2	0,2
1300000	2,5	3,66	0	0	0,2	0,2
1325000	1,97	4,1	0	0	0,2	0,2
1350000	2,83	6,63	0	0	0,6	0,4
1375000	2,11	2,99	0	0	0,2	0,1
1400000	2,17	3,98	0	0	0,3	0,9
1425000	4,76	10,24	0,1	0	0,4	0,2
1450000	4,97	10,77	0,1	0,1	41,78	0,6
1475000	2,85	5,52	0	0	0,3	0,2
1500000	1,55	5,14	0	0	0,2	0,1
1525000	1,98	3,24	0	0	0,1	0,1
1550000	1,88	5,21	0	0	0,2	0,1

Рисунок 4. Полученные значения программы.

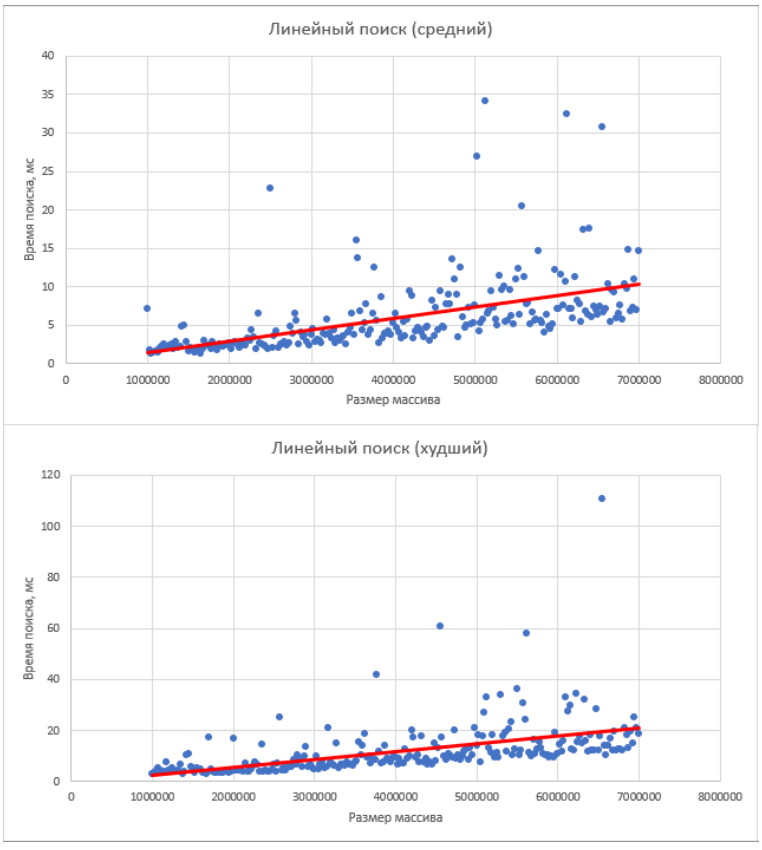

Рисунок 5. Полученный график линейного поиска

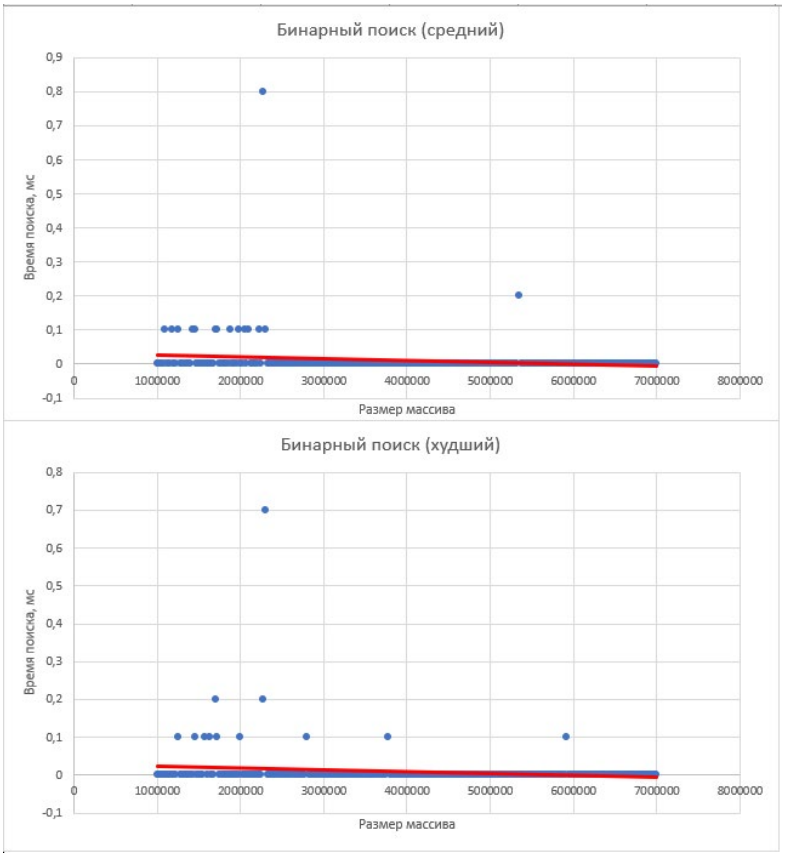

Рисунок 6. Полученный график бинарного поиска.

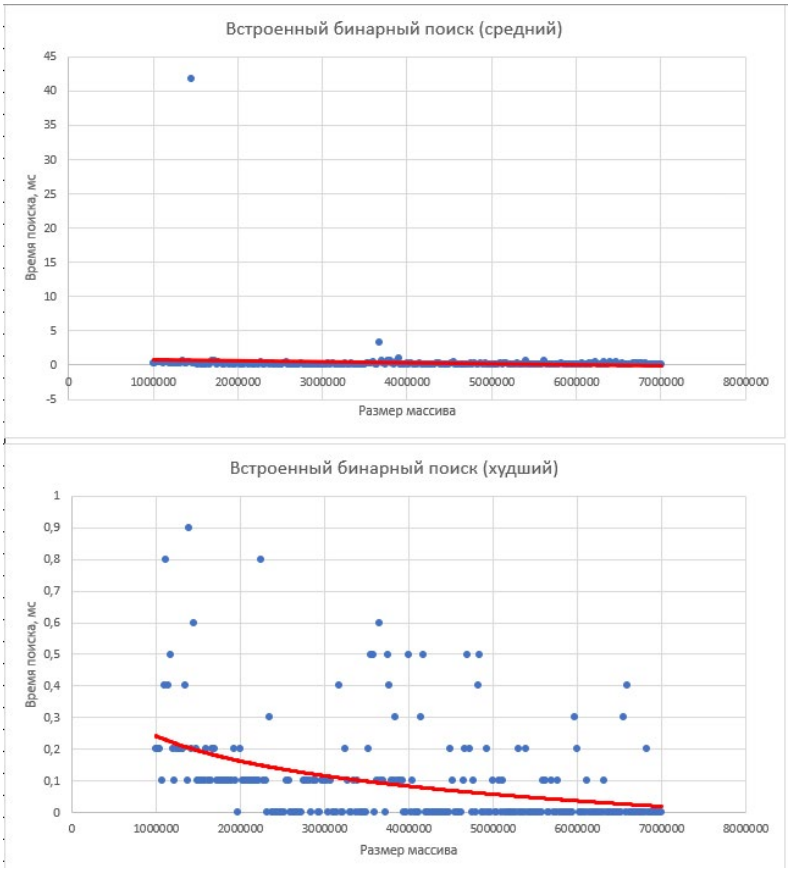

Рисунок 7. Полученный результат встроенного бинарного поиска.

Вывод: в ходе выполнения практической работы были рассмотрены методы линейного и бинарного поисков элементов в массиве. Из полученных данных выявлено безоговорочное преимущество бинарного поиска перед линейным.