МГТУ им. Баумана

Лабораторная работа №7

По курсу: "Анализ алгоритмов"

Алгоритмы поиска в словаре

Студент: Аникин И. А., ИУ7-51Б

Преподаватель: Волкова Л.Л.

Москва, 2020

Содержание

Введение 3

1. Аналитическая часть 4
- 1.1. Линейный поиск 4
- 1.2. Бинарный поиск 4
- 1.3. Частотный поиск 5

2. Конструкторская часть 6
- 2.1. Требования к ПО 6
- 2.2. Разработка алгоритмов 6

3. Технологическая часть 7
- 3.1. Листинги алгоритмов 8
- 3.2. Тестирование 9

4. Исследовательская часть 10
- 4.1. Сравнение временных характеристик алгоритмов 10

Заключение 13

Список литературы 14

Приложения 15

Введение

В жизни очень часто используются словари. Изначально словарь был придуман для перевода слов.

В программировании словарь представляет собой более абстрактную сущность. Словарь - это структура данных, которая хранит пары вида ключ-значение.

Основная задача словаря - обеспечивать быстрый поиск. В ходе лабораторной работы будет рассмотрено три алгоритма поиска в словаре.

1. Аналитическая часть

Рассмотрим цель и задачи данной лабораторной работы.

Цель лабораторной работы - разработать и провести сравнительный анализ алгоритмов поиска в словаре.

Для достижения данной цели можно выделить следующие задачи:

- разработать алгоритмы линейного, бинарного и частотного поиска;
- реализовать разработанные алгоритмы;
- провести тестирование разработанных алгоритмов по методу черного ящика;
- провести замеры времени работы реализаций алгоритмов;
- провести сравнительный анализ временной эффективности разработанных алгоритмов.

1.1. Линейный поиск

Самый простой вид поиска. Ключи просматриваются последовательно пока не найдется искомый или пока не закончится словарь.
Сложность - $O(n)$, где n - длина словаря.

1.2. Бинарный поиск

Эффективный вид поиска. Может применяться только на отсортированном списке ключей.

На каждой итерации берется середина между левой и правой границей поиска и сравнивается значение ключа по середине. Если он больше искомого, то искомый ключ находится в левой половине, если меньше, то в правой. Соответственно, размер рабочей области уменьшается в два раза за одну итерацию.

Сложность - $O(log(n))$, где n - длина словаря.

1.3. Частотный поиск

В этом способе используются дополнительные оптимизации для улучшения времени поиска в среднем случае. Понятно, что в худшем случае на отсортированных данных поиск будет занимать $log(n)$ от длины списка ключей.

Время в среднем случае можно улучшить засчет разделения данных на определенные категории и поиска ключа внутри соответствующей категории. Например, разбиение слов на категории по первой букве и поиск ключа внутри категории, соответствующей ключу[1].

В данной лабораторной работе в качестве ключей используются строки из русских букв. Данные разбиваются на 33 категории, которые соответствуют первым буквам слов. После определения нужной категории происходит бинарный поиск уже внутри категории. Это позволяет уменьшить начальное расстояние между границами поиска.

Сложность - $O(log(n))$, где n - длина категории.

2. Конструкторская часть

2.1. Требования к ПО

Разработанное ПО должно обладать следующей функциональностью:

- считывание заранее подготовленного списка ключей;
- выбор алгоритма поиска;
- вывод результата поиска;
- осуществление замеров процессорного времени работы реализаций алгоритмов.

2.2. Разработка алгоритмов

В ходе данной лабораторной работы будет реализовано 3 различных алгоритма поиска в словаре:

- линейный поиск;
- бинарный поиск;
- частотный поиск.

На рисунке 5 в приложении 1 представлена схема алгоритма линейного поиска.

На рисунке 6 в приложении 2 представлена схема алгоритма бинарного поиска.

На рисунках 7 в приложении 3 представлена схема алгоритма частотного поиска.

3. Технологическая часть

В ходе выполнения лабораторной работы были реализованы следующие модули:

- модуль main.py для работы с функциями поиска;
- модуль analysis.py для замера времени работы реализаций алгоритмов;
- модуль test.py для тестирования разработанных алгоритмов поиска;
- модуль search.py с алгоритмами поиска.

В качестве языка разработки был выбран язык python, т. к. он имеет ряд преимуществ для достижения цели лабораторной работы:

- высокая скорость разработки программ;
- простой синтаксис языка;
- наличие большого количества библиотек, что позволяет легко анализировать полученные результаты и проводить тестирование.

3.1. Листинги алгоритмов

На листинге 1 приведена реализация алгоритма линейного поиска.

Листинг 1: линейный поиск

```
def lsearch(arr, key):
    for i in arr:
        if i == key:
            return True
    return False
```

На листинге 2 приведена реализация алгоритма бинарного поиска.

Листинг 2: бинарный поиск

```
def bsearch_wrap(arr, key):
    arr.sort()
    return bsearch(arr, key, 0, len(arr))


def bsearch(arr, key, l, r):
    left = l
    right = r

    mid = (left+right) // 2
    while left < right:
        if arr[mid] == key:
            return True
        elif arr[mid] > key:
            right = mid - 1
        else:
            left = mid + 1
        mid = (left+right) // 2
    return arr[mid] == key
```

На листинге 3 приведена реализация алгоритма частотного поиска. В качестве функции бинарного поиска используется функция из листинга 2.

Листинг 3: частотный поиск

```
def fsearch_wrap(arr, key):
    freq = [0] * 34
    for s in arr:
        freq[ord(s[0]) - ord('A')] += 1

    arr.sort()
    l = sum(freq[:ord(key[0]) - ord('A')])
    r = l + freq[ord(key[0]) - ord('A')]

```

```
10      return bsearch(arr, key, l, r)
```

3.2. Тестирование

Тестирование разработанных алгоритмов проводилось на наборе тестов, который представлен в таблице 1.
В качестве массива ключей взят массив строк ["арбуз "банан "виноград "груша "дыня"].

Таблица 1: Набор тестов

Ключ поиска	Ожидаемые выходные данные
"арбуз"	True
"банан"	True
"виноград"	True
"груша"	True
"дыня"	True
"вино"	False

Все тесты пройдены успешно.

4. Исследовательская часть

4.1. Сравнение временных характеристик алгоритмов

Проведем исследование временных характеристик реализованных алгоритмов. Замеры времени происходили с использованием ЦП Intel Xeon E3-1270 V2 и 16 ГБ ОЗУ.
Ниже представлены графики, показывающие соотношение времени работы алгоритмов поиска при различной длине используемого словаря. На вход подается ключ поиска.

На рисунке 1 изображены графики среднего времени поиска.

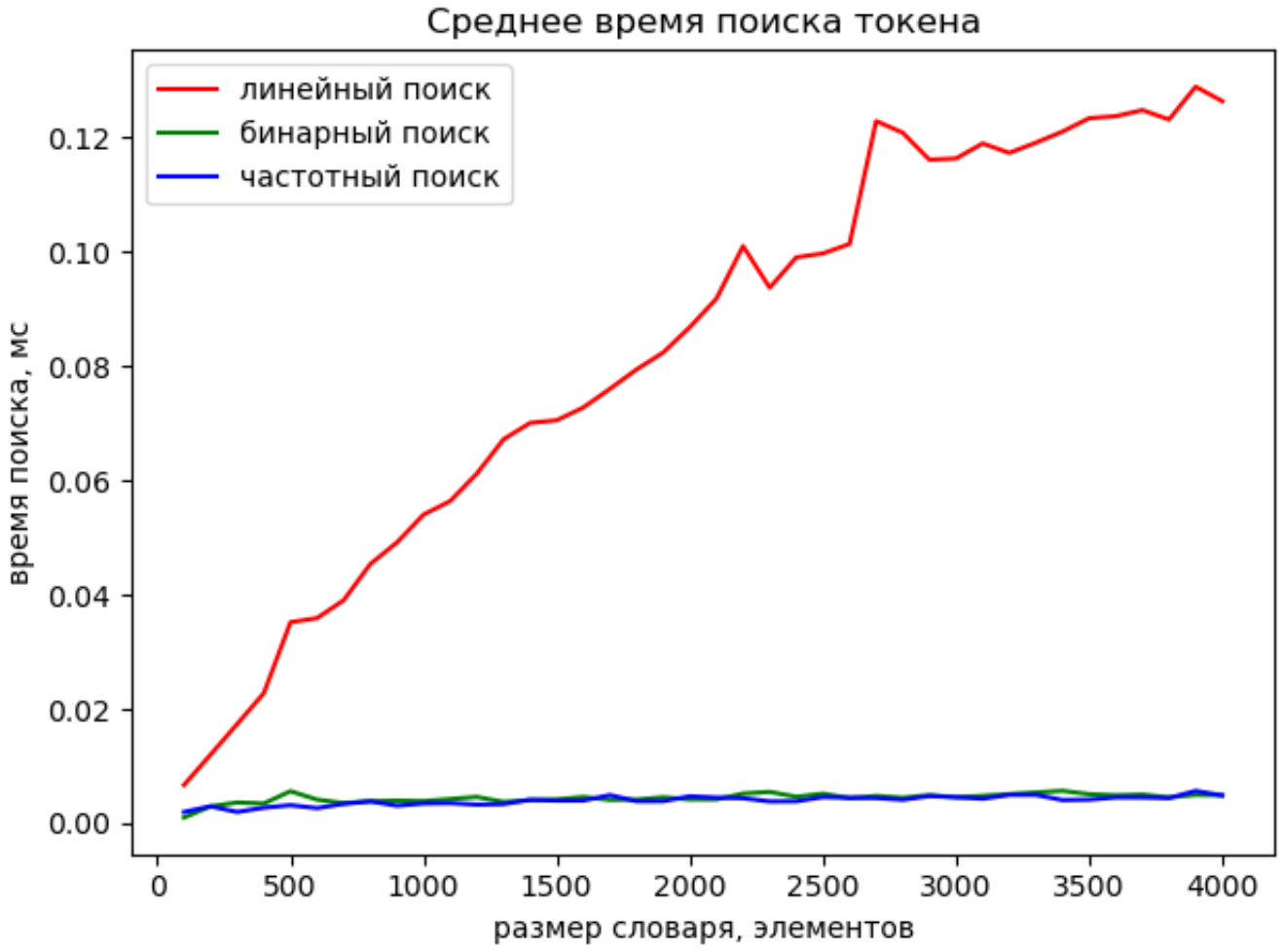

Рис. 1: Среднее время поиска в словаре для трех алгоритмов

На рисунке 2 изображены графики среднего времени поиска для бинарного и частотного поиска при большем масштабе.

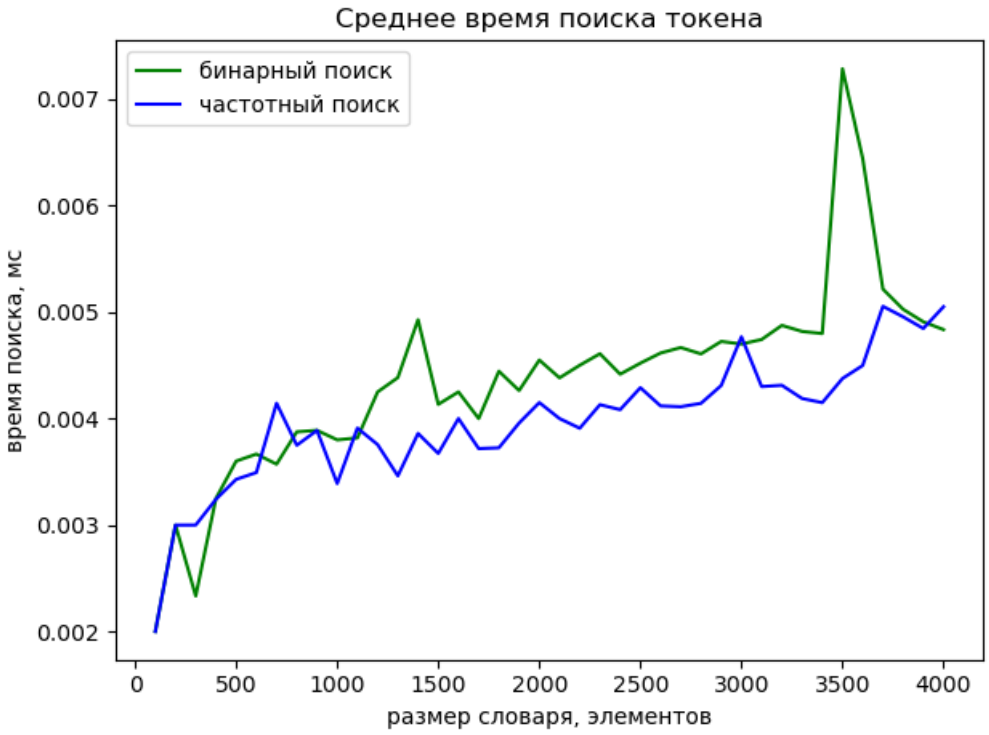

Рис. 2: Среднее время поиска в словаре для бинарного и частотного поиска

На рисунке 3 изображены графики худшего времени поиска.

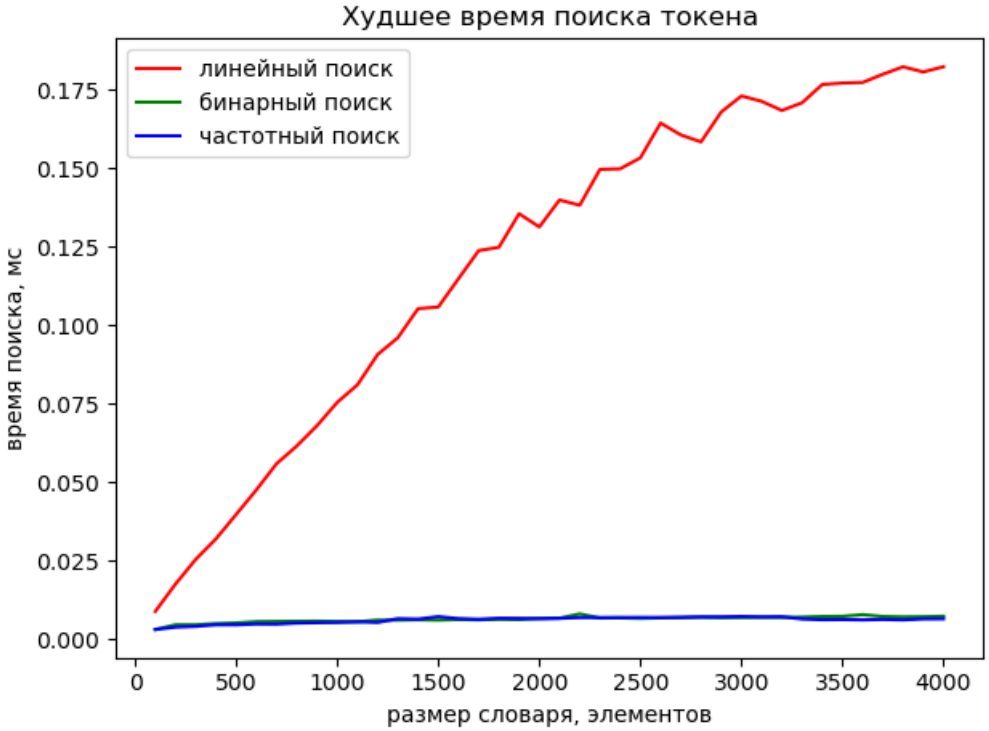

Рис. 3: Худшее время поиска в словаре для трех алгоритмов

На рисунке 4 изображены графики худшего времени поиска для бинарного и частотного поиска при большем масштабе.

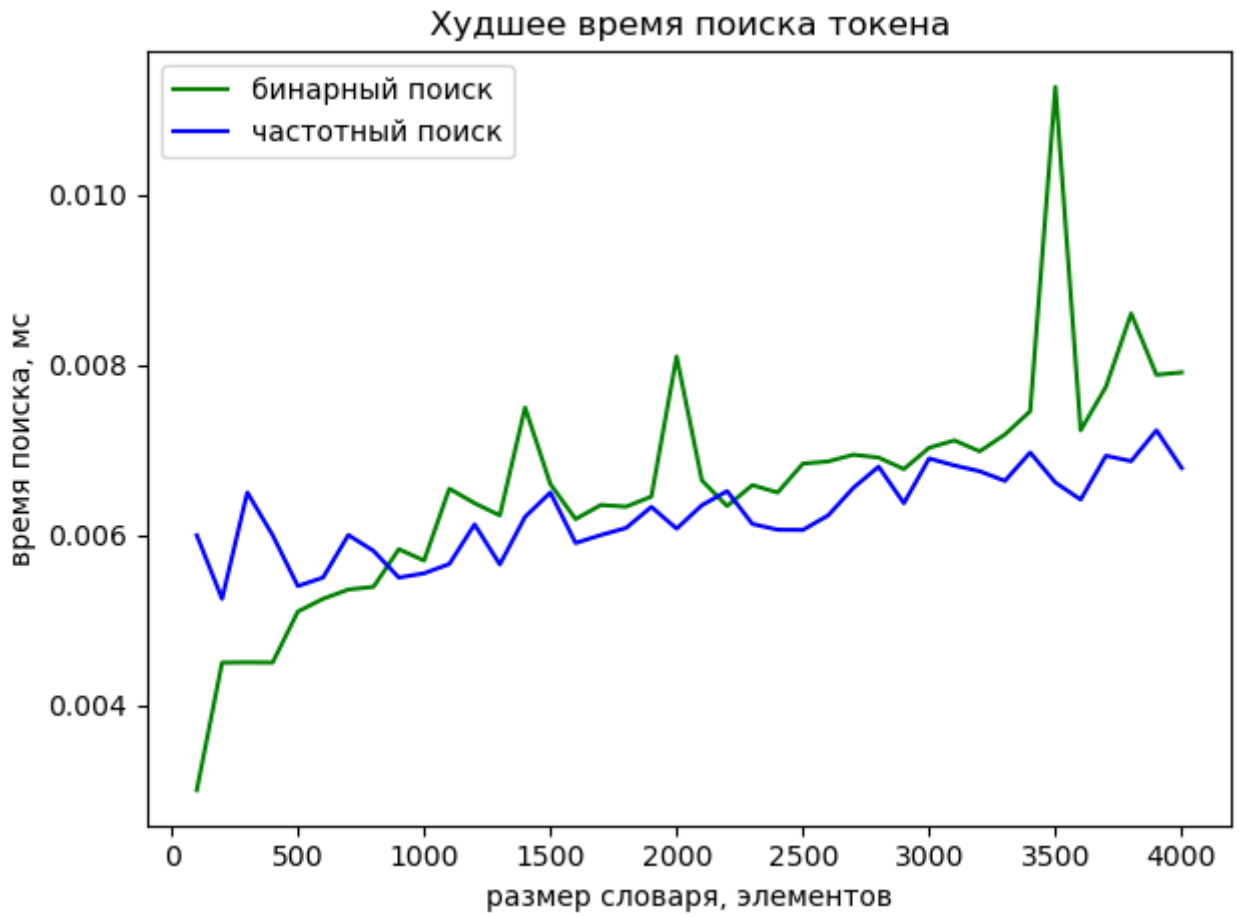

Рис. 4: Худшее время поиска в словаре для бинарного и частотного поиска

Исходя из данных графиков, можно сделать следующий вывод. Наиболее быстрое среднее и худшее время поиска обеспечивает алгоритм частотного поиска, т.к. он использует дополнительные оптимизации, которые позволяют уменьшить размер зоны поиска, в отличие от линейного и бинарного поисков, которые изначально рассматривают список ключей целиком.

К применению рекомендуется частотный алгоритм поиска, как наименее затратный по времени.

Заключение

В ходе выполнения лабораторной работы были выполнены следующие задачи:

- разработаны алгоритмы линейного, бинарного и частотного поиска;
- реализованы разработанные алгоритмы;
- проведено тестирование разработанных алгоритмов по методу черного ящика;
- проведены замеры времени работы реализаций алгоритмов;
- проведен сравнительный анализ временной эффективности разработанных алгоритмов.

Цель лабораторной работы достигнута, все задачи выполнены.

Даны рекомендации по использованию исследованных алгоритмов.

По итогам достигнутых задач можно сделать следующие выводы.

К применению рекомендуется частотный алгоритм поиска, как наименее затратный по времени. Этот алгоритм использует дополнительные оптимизации, которые позволяют уменьшить размер зоны поиска, в отличие от линейного и бинарного поисков, которые изначально рассматривают список ключей целиком.

Список литературы

[1] Кнут, Д. Э. Искусство программирование, том 3. Сортировка и поиск. -Изд. 2. -М.: ООО "И. Д. Вильямс 2007. -832с.

Приложения

Приложение 1

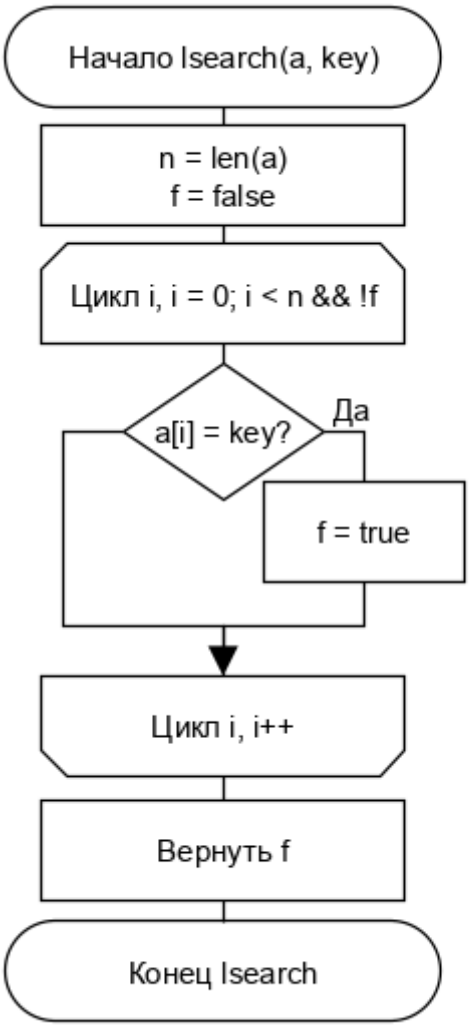

Рис. 5: Схема алгоритма линейного поиска

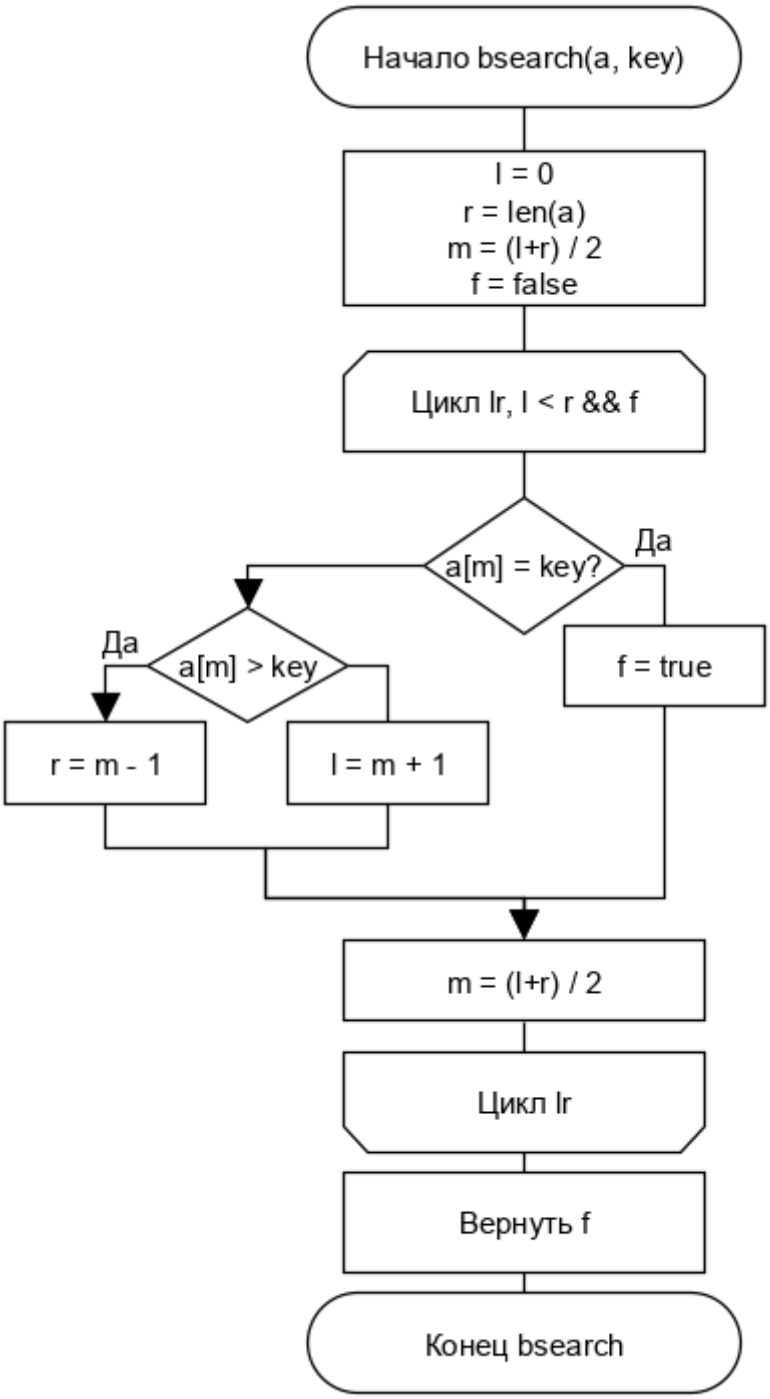

Рис. 6: Схема алгоритма бинарного поиска

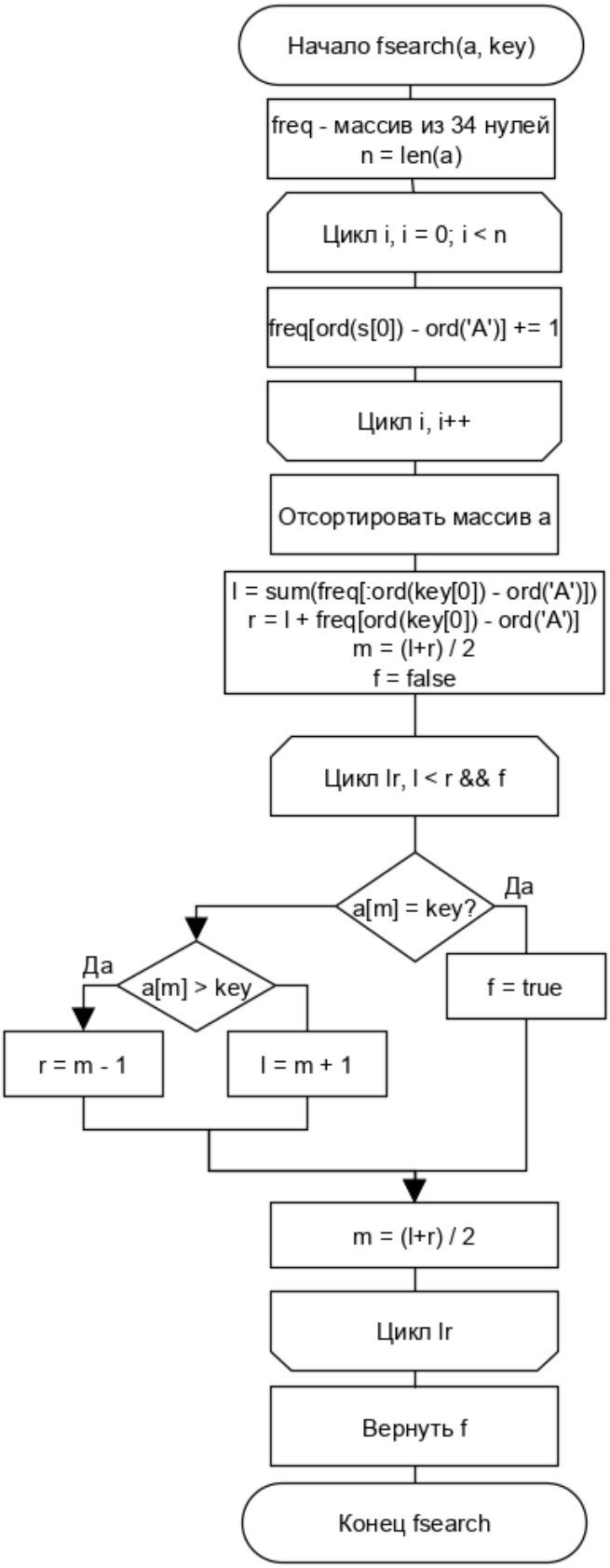

Рис. 7: Схема алгоритма частотного поиска