|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ | | | | | | | | |
|  | | |  | | |  | | |
| ФГБОУ ВО «Пермский государственный  национальный исследовательский университет» | | | | | | | | |
|  | | |  | | |  | | |
|  | | Алгоритмы и анализ сложности  *Практическая работа*  **«Поиск числа в массиве»** | | | | |  | |
|  | | |  | | |  | | |
|  | Работу выполнили  студенты группы ПМИ 3-4  Гришин Н.А.  Вотинова Е.С. | | |  | Проверила  Постаногова И. Ю.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 | | |  |
|  |  | | |  |  | | |  |
| Пермь 2020 | | | | | | | | |

Оглавление

[1. Постановка задачи 3](#_Toc56357446)

[2. Краткое описание фактически реализованных алгоритмов 4](#_Toc56357447)

[3. Теоретическая оценка сложности реализованных алгоритмов 5](#_Toc56357448)

[4. Средства разработки 6](#_Toc56357449)

[5. Полученные результаты 7](#_Toc56357450)

[6. Данные для подведения статистики 8](#_Toc56357451)

[7. Вывод 9](#_Toc56357452)

1. Постановка задачи

Дано: в первой строке записано количество чисел в массиве (*n*) и искомый элемент (*x*). Во второй строке перечислено *n* чисел ­ элементы массива.

Получить: индекс первого элемента, равного *x*, либо значение «–1», если такого элемента нет.

Алгоритмы:

линейный поиск;

бинарный поиск.

1. Краткое описание фактически реализованных алгоритмов

В ходе выполнения работы нами были использованы следующие алгоритмы поиска элемента в массиве:

**Линейный поиск**

Начиная с первого, все элементы массива последовательно просматриваются и сравниваются с искомым. Если на каком-то шаге текущий элемент окажется равным искомому, тогда элемент считается найденным, и в качестве результата возвращается индекс этого элемента, и сразу выходим из цикла. Иначе, возвращаем значение -1.

**Бинарный поиск**

Бинарный поиск производится в упорядоченном массиве. Для начала нужно отсортировать массив (алгоритм быстрой сортировки). При бинарном поиске искомый ключ сравнивается с ключом среднего элемента в массиве. Если они равны, то поиск успешен, выводим индекс элемента. В противном случае поиск осуществляется аналогично в левой или правой частях массива. Если элемент не найден, возвращаем -1.

1. Теоретическая оценка сложности реализованных алгоритмов

|  |  |  |
| --- | --- | --- |
|  | **Линейный поиск** | **Бинарный поиск** |
| **Максимальная** |  |  |
| **Минимальная** |  |  |

1. Средства разработки

Программа была реализована с помощью языка высокого уровня С++ и использовалась среда разработки Visual Studio 2019.

Характеристики процессора:

* Intel core i5
* Тактовая частота 1.19 GHz
* ОЗУ 8 Гб

1. Полученные результаты

Экспериментально определено (168.322 с в случае бинарного поиска, с учетом сортировки и 0.422 с в случае линейного поиска).

На следующих графиках представлена зависимость времени реальных вычислений и теоретическим временем. Одна простая операция = 0,00000353 мс.

Рисунок - Соотношение времени в случае бинарного поиска.

Рисунок - Соотношение времени в случае линейного поиска

Вывод: теоретическая оценка очень близка с фактическими результатами.

1. Данные для подведения статистики

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Линейный поиск** | | **Тест1** | **Тест2** | **Тест3** | **Тест4** | **Тест5** |
| **50млн** | **Время** | 0,088 | 0,103 | 0 | 0,134 | 0 |
| **Решение** | -1 | -1 | 69297 | -1 | 15159 |
| **100млн** | **Время** | 0,178 | 0,001 | 0,171 | 0,181 | 0,179 |
| **Решение** | -1 | 100652 | -1 | -1 | -1 |
| **150млн** | **Время** | 0,251 | 0,255 | 0,35 | 0 | 0 |
| **Решение** | -1 | -1 | -1 | 16429 | 10982 |
| **200млн** | **Время** | 0,304 | 0,334 | 0 | 0 | 0,001 |
| **Решение** | -1 | -1 | 15765 | 55586 | 55051 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Линейный поиск** | **50млн** | **100млн** | **150млн** | **200млн** |
| **Среднее время** | 0,049 | 0,093 | 0,147 | 0,185 |
| **Время лучшего случая** | 0 | 0 | 0 | 0 |
| **Время худшего случая** | 0,088 | 0,180 | 0,286 | 0,363 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Бинарный поиск** | | **Тест1** | **Тест2** | **Тест3** | **Тест4** | **Тест5** |
| **50млн** | **Время** | 0 | 0 | 0 | 0 | 0 |
| **Решение** | 6467664 | -1 | -1 | -1 | 4050181 |
| **100млн** | **Время** | 0 | 0 | 0 | 0 | 0 |
| **Решение** | 40995206 | 17173100 | -1 | -1 | -1 |
| **150млн** | **Время** | 0 | 0 | 0 | 0 | 0 |
| **Решение** | 22825065 | -1 | -1 | 14904957 | 35958622 |
| **200млн** | **Время** | 0 | 0 | 0 | 0 | 0 |
| **Решение** | -1 | -1 | 9538594 | 55586 | 34371081 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Бинарный поиск** | **50млн** | **100млн** | **150млн** | **200млн** |
| **Среднее время** | 0,006 | 0,007 | 0,008 | 0,009 |
| **Время лучшего случая** | 0 | 0 | 0 | 0 |
| **Время худшего случая** | 0,011 | 0,014 | 0,015 | 0,017 |

1. Вывод

В ходе работы трудностей не возникало. Алгоритмы просты в реализации. Тестирование проходило долго только в бинарном поиске, так как для сортировки массива требовалось время.

Алгоритм линейного поиска прост в использовании, не требует упорядоченности массива и нет необходимости в каких-либо упорядоченных элементах. Алгоритм двоичного поиска, однако, сложнее, и элементы обязательно располагаются по порядку.

Оба алгоритма точные, работают очень быстро даже при больших размерах входных данных. Несмотря на то, что бинарный алгоритм быстрее, при работе с массивом большого размера лучше использовать линейный поиск, так как не надо сортировать массив, что достаточно затратно по времени.

1. Приложение

Файлы исходного кода прилагаются: searchElemInMass.cpp