Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Институт информационных технологий

Факультет компьютерных технологий

Кафедра информационных систем и технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

«АНАЛИЗ АЛГОРИТМА БИНАРНОГО ПОИСКА»

Студент:

гр. 381574 Жгуновский О.Б.

Руководитель:

ассистент кафедры ИСиТ ИИТ БГУИР Потоцкий Д.С.

Минск 2024

Оглавление

[Введение 3](#_Toc159068080)

[1. Моделирование программного продукта 5](#_Toc159068081)

[2. Проектирование программного продукта 6](#_Toc159068082)

[3. Оценка работы (тестирование) программного средства и анализ результатов 7](#_Toc159068083)

[Выводы 8](#_Toc159068084)

[Список используемой литературы 9](#_Toc159068085)

[Приложение А 10](#_Toc159068086)

# Введение

Бинарный поиск является одним из наиболее эффективных алгоритмов поиска в упорядоченном массиве данных. Этот метод, основанный на принципе деления пополам, широко применяется в различных областях информатики и программирования, где требуется быстрый поиск элементов в больших объемах данных.

Цель данной курсовой работы заключается в анализе алгоритма бинарного поиска, его моделировании, проектировании, а также оценке работы программного средства, реализующего данный алгоритм на языке программирования C#. В ходе работы будут рассмотрены основные принципы функционирования алгоритма, его преимущества и недостатки, а также проведено тестирование разработанного программного продукта с последующим анализом полученных результатов.

Задачами курсовой работы являются:

* Рассмотреть общие принципы работы алгоритма бинарного поиска;
* Реализация алгоритма бинарного поиска на языке программирования C#;
* Тестирование реализованного алгоритма;
* Понять суть работы алгоритма бинарного поиска.

Особое внимание будет уделено оценке работы программного средства и анализу результатов. Это включает в себя тестирование программы на различных наборах данных и анализ времени выполнения.

В конечном итоге, данная работа поможет получить глубокое понимание принципов работы алгоритма бинарного поиска, его применимости и эффективности, а также обеспечит практические навыки реализации и использования данного алгоритма для решения различных задач в области программирования и информационных технологий.

# Моделирование программного продукта

Алгоритм бинарного поиска — эффективный метод поиска элемента в отсортированном массиве данных. Данный алгоритм работает путём деления массива пополам, и последующего сравнения искомого значения с элементом в середине массива. Если значение совпадает с искомым — алгоритм завершается. Если же нет — алгоритм, путём сравнения искомого значения со значениями по бокам центрального элемента массива, определяет, в какой половине массива содержится искомое значение, и повторяет вышеописанные действия до тех пор, пока искомый элемент не будет найден или не останется больше элементов для проверки.

Если описывать алгоритм по шагам, это можно сделать следующим образом:

1. Получаем массив данных;
2. Сортируем массив данных;
3. Определяем левую и правую границы поиска;
4. Находим средний элемент, между границами поиска;
5. Сравниваем средний элемент с искомым значением;
6. Если искомое равно среднему элементу, заканчиваем поиск;
7. Если искомое больше среднего элемента, устанавливаем левую границу на значение среднего элемента;
8. Если искомое меньше среднего элемента, устанавливаем правую границу на значение среднего элемента;
9. Повторяем пункты 4-8 до тех пор, пока значение среднего элемента не будет совпадать с искомым (что отражено в пункте 6), либо же, когда левая граница становится больше правой (это означает, что элемент не найден в массиве);
10. Возвращаем результат.

Блок-схема алгоритма бинарного поиска, представлена на рисунке 1.1:

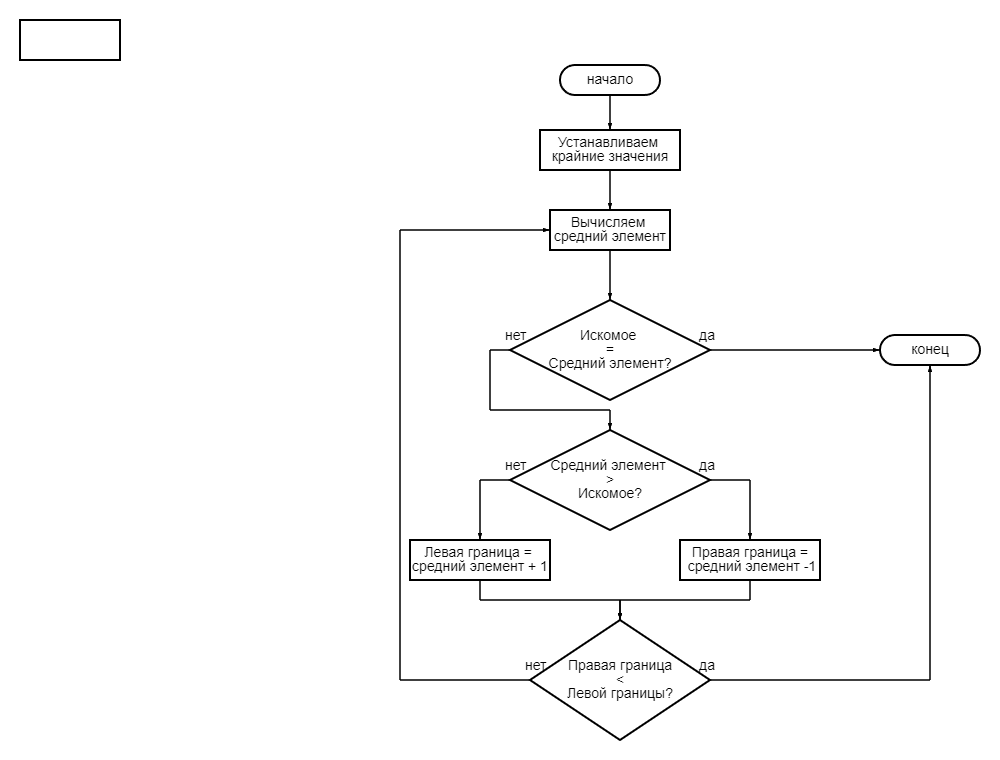


Рисунок 1.1 – блок-схема алгоритма бинарного поиска

Важной особенностью данного алгоритма является то, что он требует отсортированный массив. Соответственно, без выполнения данного условия, работать он вообще не будет.

Алгоритм обладает сложностью O(log n) в худшем и среднем случаях[1], где n — количество элементов в массиве. Что делает данный алгоритм, одним из самых эффективных при работе с большими объемами данных (если они, разумеется, отсортированы).

Данный алгоритм относительно прост в реализации, и тратит мало памяти — O(1).

Для реализации данного алгоритма, мною было выбран тип данных «массив» (array). Это наиболее простой в плане понимания тип данных, включённый в стандартную библиотеку C#. Он обеспечивает прямой доступ к элементам по индексу, что делает его подходящим для реализации бинарного поиска. Массивы имеют фиксированную длину, которая задаётся при их инициализации. Однако, этого достаточно для реализации данного алгоритма. Тем более, что новых элементов добавлять во время исполнения мною не планируется.

Для того, чтобы можно было провести анализа, необходимо выбрать другой алгоритм поиска. И мною был выбран алгоритм линейного поиска.

Линейный поиск — это простой алгоритм поиска элемента в списке, массиве или последовательности. Работает он, путем последовательного перебора каждого элемента в структуре данных, и сравнения его с целевым значением. Если элемент совпадает с целевым значением, то алгоритм возвращает его позицию, если нет - продолжает. Эффективность сильно зависит от размера списка: время выполнения линейного поиска пропорционально количеству элементов в списке. Иными словами, его сложность будет составлять О(n)[2].

Стоит отметить, что, он гораздо проще в реализации нежели бинарный поиск, и тратит тоже минимум памяти — O(1).

Так же, стоит отметить, что в отличии от алгоритма поиска, нет необходимости получать отсортированный массив. Однако, так как проверки теряют всякий смысл при разных входных данных, мы будем проводить поиск в отсортированном массиве.

Тип данных следует использовать тот же, что и для алгоритма бинарного поиска. По тем же причинам, а так же, для корректности результатов самого теста

Если описывать его по шагам, то выйдет следующее:

1. Получаем массив данных;
2. Сортируем массив данных;
3. Ставим счётчик в i = 0;
4. Проверяем, чтобы элемент массива с индексом i, был равен искомому числу;
5. Если элемент массива с индексом i равен искомому числу, завершаем алгоритм и выводим результат;
6. Если элемент массива с индексом i не равен искомому числу, то убеждаемся, что мы не дошли до конца массива, и увеличиваем значение i на 1, после чего, возвращаемся к шагу 4;
7. Если же мы дошли до конца массива, то возвращаем информацию о том, что искомого элемента нету в массиве.

Блок-схема алгоритма линейного поиска представлена на рисунке 1.2:

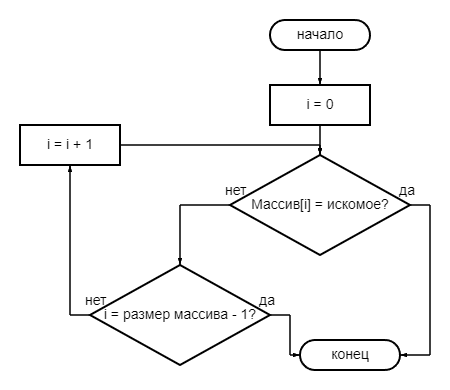


Рисунок 1.2 – блок-схема алгоритма линейного поиска

Подводя итоги. Для выполнения алгоритмов, программа должна получить следующие данные:

* Отсортированные массивы;
* Значения, которое необходимо найти.

А так же, если значение, которое необходимо найти в массиве отсутствует, то алгоритм должен это указать после своего выполнения.

Так как бинарный поиск может показать свою результативность только на больших массивах, и так как для получения каких-либо результатов, которые имеют значение требуется большой набор данных, предоставлять пользователю ручной ввод массивов просто отсутствует. А значит, стоит предоставить пользователю возможность выбрать размер массивов, и искомые значения.

Заходя дальше, стоит отметить тот факт, что реальной необходимости рандомизировать значения в массивах – нету. А значит, стоит заполнить массив числами от 1 до n, где n – размер массива, введённый пользователем.

Так как «сила» бинарного поиска в поиске по большим отсортированным массивам, а линейного – в маленьких и неотсортированных, стоит сделать массивы разной длины. Таким образом, будет создаваться набор массивов, с минимальным размеров в 10, а максимальным- введённым пользователем. Шаг будет составлять 10 позиций.

Так же, стоит пользователю дать возможность выбрать массив чисел для поиска, а не просто одно число.

После отработки алгоритма, необходимо показать пользователю как минимум:

* Минимальное, максимальное и среднее время для каждого алгоритма;
* Минимальное, максимальное и среднее количество «проходов» по массиву.

# Проектирование программного продукта

# Оценка работы (тестирование) программного средства и анализ результатов

# Выводы

# Список используемой литературы

[1] Бинарный поиск. [Электронный ресурс]. - 2024. - Режим доступа: https://blog.skillfactory.ru/glossary/binarnyj-poisk/

[2] Какая временная сложность поиска в обычном массиве. [Электронный ресурс]. - 2024. - Режим доступа: https://uchet-jkh.ru/i/kakaya-vremennaya-sloznost-poiska-v-obycnom-massive/#:~:text=Сложность%20алгоритма%20линейного%20поиска%20составляет,является%20его%20простота%20и%20универсальность.

# Приложение А