***Лабораторная работа 4.* Линейный и бинарный поиск**

***Цель работы —* изучение эффективности основных методов поиска в массивах.**

# 1. АЛГОРИТМЫ ПОИСКА

Алгоритмы поиска, наряду с алгоритмами сортировки, являются основными алгоритмами обработки данных как системных, так и прикладных задач.

Задача поиска состоит в отыскании записи , имеющей ключ . Этот ключ называется аргументом поиска.

Алгоритмы поиска применяются для нахождения, например, в массиве элемента с нужными свойствами. Если множество данных задано в виде массива A= из n целых или вещественных чисел, то поиск состоит в нахождении числа , для которого , если –ключ поиска. Обычно различают постановки задачи поиска для первого и последнего вхождения элемента.

## 1.1 ЛИНЕЙНЫЙ ПОИСК

Для массива, данные в котором не упорядочены сортировкой или другим способом, единственный путь для поиска заданного элемента состоит в сравнении каждого элемента массива с заданным. При совпадении некоторого элемента массива с ключом, его позиция в массиве фиксируется, и поиск заканчивается успешно. В противном случае будут просмотрены все элементы, а поиск окажется безуспешным. Этот подход называется последовательным или линейным поиском.

**Алгоритм линейного поиска имеет вид:**

****

**пока ( И ) выполнить**

****

**всё-пока**

Равенство  означает отсутствие в массиве A ключа поиска.

Эффективность этого метода очень низкая, так как для отыскания одного элемента в массиве размера N в среднем нужно сделать N/2 сравнений.

Линейный поиск осуществляется циклом с двойным условием. Первое условие контролирует индекс на принадлежность массиву, например, (i<=N). Второе условие - это условие поиска. В цикле **while** это условие продолжения поиска: (A[i]<>X). В теле цикла обычно пишется только один оператор: изменение индекса в массиве.

После выхода из цикла необходимо проверить, по какому из условий мы вышли. В операторе **if** обычно повторяют первое условие цикла. Можно говорить об успешном поиске с циклом **while** при выполнении этого условия.

## 

## 1.2 ПОИСК С БАРЬЕРОМ

Единственно возможное убыстрение алгоритма линейного поиска заключается в упрощении логического условия. Для этого следует увеличить количество элементов массива A на один элемент со значением ключа поиска , то есть . Этот дополнительный элемент называется «барьером». Его введение позволяет алгоритм линейного поиска представить в виде:



**пока  выполнить**

****

**всё-пока**

Введение барьера позволяет не проверять каждый раз в цикле условие, связанное с границами массива.

Выход из цикла, в котором теперь остается только условие совпадения с ключом поиска, может произойти либо на найденном элементе, либо на барьере. Таким образом, после выхода из цикла проверяется, не барьер ли мы нашли? Вычислительная сложность поиска с барьером меньше, чем у линейного поиска, но также является величиной того же порядка, что и N - количество элементов массива.

Существует два способа установки барьера: дополнительным элементом или вместо крайнего элемента массива.

## 1.3 ДВОИЧНЫЙ (БИНАРНЫЙ) ПОИСК

Алгоритм двоичного поиска можно использовать для поиска элемента только в упорядоченном массиве . Упорядоченность массива позволяет при поиске использовать условия в виде неравенств.

**Принцип двоичного поиска:**

* Исходный массив делится пополам и для сравнения выбирается средний элемент. Если он совпадает с искомым, то поиск заканчивается.
* Если средний элемент меньше искомого, то все элементы левее его также будут меньше искомого. Следовательно, их можно исключить из процесса дальнейшего поиска, оставив только правую часть массива. Аналогично, если средний элемент больше искомого, то отбрасывается правая часть, а остаётся левая.
* На втором этапе выполняются аналогичные действия над оставшейся половиной массива. В результате после второго этапа останется ¼ массива.
* Процесс продолжается до тех пор, пока или элемент будет найден, или длина зоны поиска станет равной нулю. В последнем случае поиск будет безуспешен.

**Алгоритм двоичного поиска имеет вид:**

****

**пока  выполнить**

****

**если  то **

**иначе **

**всё-если**

**всё-пока**

Существуют две модификации этого алгоритма для поиска первого и последнего вхождения. Все зависит от того, как выбирается средний элемент: округлением в меньшую или большую сторону. Например, округление c:=(L+R+1) **div** 2 даёт середину с округлением в большую сторону. В случае округления в меньшую сторону средний элемент относится к левой части массива, а при округлении в большую сторону – к правой.

В процессе работы алгоритма двоичного поиска размер фрагмента, где этот поиск должен продолжаться, каждый раз уменьшается примерно в два раза. Это обеспечивает вычислительную сложность алгоритма порядка , где N - количество элементов массива.

**1.4. ЗАДАНИЕ**

1. **Сравнить время выполнения линейного поиска в массиве случайных чисел  при  = 100, 500, 1000, 3000, 10000. В каждом случае использовать не менее трёх массивов одного размера с заданным числом элементов. Построить диаграмму зависимости времени поиска и числа сравнений от числа элементов в массиве.**
2. **Провести сравнение времени поиска и количества повторений цикла при линейном и двоичном поиске в предварительно отсортированном массиве.**
3. **Исследовать эффективность введения «барьера» при линейном поиске. Представить таблицы, иллюстрирующие сделанные выводы.**
4. **В отчёте привести тексты программ с необходимыми комментариями, позволяющими пояснить суть выполняемых действий.**