ПРАКТИЧЕСКОЕ ЗАДАНИЕ: **АЛГОРИТМЫ ПОИСКА**

ЦЕЛЬ РАБОТЫ: Приобрести навыки программной реализации алгоритмов поиска в различных структурах данных.

**Краткие теоретические сведения**

**Задача поиска в массивах**:

Пусть задано множество данных, состоящее из некоторого количества элементов. Элемент множества будет называться ключом, и обозначаться латинской буквой “**k**” с индексом, указывающим номер элемента - **{k1, k2, k3...kn}**.

Требуется проверить, входит ли заданный ключ **ki** в множество. Поиск может быть завершён в двух случаях:

1. Ключ во множестве отсутствует;
2. Ключ найден во множестве: определено первое вхождение заданного ключа (элемента) в исходном массиве. определен номер элемента массива.

**Общий алгоритм поиска данных:**

Шаг 1. Вычисление элемента (часто предполагает получение значения элемента, ключа элемента и т.д.).

Шаг 2. Сравнение элемента с эталоном или сравнение двух элементов (в зависимости от постановки задачи).

Шаг 3. Перебор элементов множества, то есть прохождение по элементам массива.

Основные идеи различных алгоритмов поиска сосредоточены в методах перебора и стратегии поиска. Алгоритмы поиска можно разделить на следующие группы:

1. Основанные на сравнении ключей

* бинарный;
* индексно – последовательный;
* интерполяционный;
* по бинарному дереву поиска;
* последовательный;
* Фибоначчи.

1. Основанные на числовых свойствах ключей

* по бору;
* хеширование.

**Основные определения**

**Бинарный (двоичный, дихотомический) поиск** – это поиск заданного элемента на упорядоченном множестве, осуществляемый путем неоднократного деления этого множества на две части таким образом, что искомый элемент попадает в одну из этих частей.

**Интерполирование** – нахождение промежуточных значений величины по имеющемуся дискретному набору известных значений.

**Ключ поиска** – это поле записи, по значению которого происходит поиск.

**Поиск** – это процесс нахождения конкретной информации в ранее созданном множестве данных.

**Последовательный (линейный) поиск** – это простейший вид поиска заданного элемента на некотором множестве, осуществляемый путем последовательного сравнения очередного рассматриваемого значения с искомым до тех пор, пока эти значения не совпадут.

**Последовательный поиск**

В последовательном поиске исходное множество не упорядоченно, т.е. имеется произвольный набор ключей **{k1, k2, k3...kn}**. Метод заключается в том, что отыскиваемый ключ **ki** *последовательно* сравнивается со всеми элементами множества. Поиск заканчивается досрочно, если ключ найден.

**Интерполяционный поиск**

Интерполяционный поиск работает также с упорядоченными массивами и похож на бинарный. Но в отличие от него, набор данных не делится на две равные части, а, используя операцию интерполирования, определяется приблизительное расположение ключа (искомого элемента), ориентируясь на расстояние (шаг) между искомым и текущим значением элемента.

Первоначальное сравнение осуществляется на расстоянии шага **d** , который определяется по формуле:

|  |  |
| --- | --- |
| **d=,** где: | i – номер первого рассматриваемого элемента  j – номер последнего рассматриваемого элемента  K – отыскиваемый ключ  значения ключей в i и j позициях  [квадратные скобки] – целая часть от числа. |

Шаг **d** меняется после каждого этапа. Метод заканчивает работу при **d=0**, при этом анализируются соседние элементы, после чего делается окончательный вывод - о наличии / отсутствии ключа **K** в наборе данных.

Пример3.Дано множество ключей:

{2,9,10,12,20,24,28,30,37,40,45,50,51,60,65,70,74,76}.

Найти ключ K=70.

**Шаг 1**. Определим шаг d для исходного множества ключей:

D =[ (18-1)(70-2)/(76-2) ]=15

Сравниваем ключ, стоящий под шестнадцатым порядковым номером в данном множестве с искомым ключом: K16~K 70=70. Результат поиска: ключ найден.

Интерполяционный поиск хорошо работает на последовательности, элементы которой распределены равномерно относительно друг друга. Интерполяционный поиск в эффективности, как правило, превосходит двоичный, в среднем требуя log(log(N)) операций, время его работы составляет O(log(log(N))). Но на больших объемах данных увеличивается до O(N).

На практике при поиске в больших файлах оказывается выгодным на ранних стадиях применять интерполяционный поиск, а затем, когда диапазон существенно уменьшится, переходить к двоичному.

**Индексно - последовательный метод поиска**

Для увеличения эффективности поиска в отсортированном наборе используют индексно – последовательный метод; порядок его эффективности - √ n, но применение приводит к увеличению требуемого пространства.

Исходный отсортированный набор данных размером n элементов разобьем на блоки из m элементов. В дополнение к исходному набору создадим вспомогательную таблицу, называемую индексом (Index); в неё помещаем *максимального представителя* каждого блока исходной таблицы. Каждый элемент Index состоит из ключа Kindex и указателя Pindex на запись в исходном наборе, соответствующую ключу Kindex. они так же как и элементы в исходном наборе, должны быть отсортированы по тому же ключу. Количество элементов в Index (размер таблицы):

Size=|(n-1)/m|+1

Сначала производится *последовательный* поиск в таблице индексов по заданному ключу. Находим первый элемент в Index, ключ которого не меньше ключа элемента поиска: K < Index.Kindex[i], т.е установили нижнюю границу поиска в основном наборе – Index.Pindex[i].

Затем организуем поиск ключа K в основной таблице на участке Index.Pindex[i]..Index.Pindex[i+1], если K нет здесь, его нет в этом наборе данных.

Пример 4. Дано множество ключей:

{2,5,6,9,12,16,18,20,21,28,32,39,44,46,51,55,60}.

Найти ключ K=44.

Решение: {2,5,6,9,11, | 16,18,20,21,25, | 30,39,44,46,50, | 55,60}

Index 11 | 25 | 50 | 60

4 | 9 | 14 | 16

Поиск от 10-го до 14-го элемента в основной таблице.

ЗАДАНИЕ

1. Разработать программу решения задачи поиска в массиве с использованием алгоритмов ***Вставка А***.

***Вставка А***

1. Индексно-последовательный поиск, Бинарный поиск
2. **Индексно-последовательный поиск, Интерполяционный поиск**
3. Индексно-последовательный поиск, Поиск Фибоначчи
4. Сравнить теоретическую и практическую сложность реализованных алгоритмов поиска (например, количество этапов).

ВХОДНЫЕ ДАННЫЕ:

1. Бинарный поиск, Интерполяционный поиск, Поиск Фибоначчи реализовывать на массиве чисел (размерность - 100, 500 элементов).
2. Индексно – последовательный поиск реализовывать на массиве записей (структур) (размерность - 30 элементов).

Структура массива записей и диапазон чисел приведены в Приложении. Для реализации индексно – последовательного поиска добавить поле Код записи, по этому полю выполняется сортировка.

Входные данные генерируются случайным образом; если требуются отсортированные данные, использовать реализацию сортировки из ЛР 1

## **ПРИЛОЖЕНИЕ** ВАРИАНТЫ ЗАДАНИЙ

При реализации индексно-последовательного поиска в структуру записи добавить поле Код записи, он должен быть уникальным; по этому полю выполняется сортировка.

| **№  варианта** | **Структура (имя, поля)** | **Массив** |
| --- | --- | --- |
|  | STUDENT   * + фамилия и инициалы;   + номер группы;   + успеваемость (массив из 5 элементов). | [-20, 50] |
|  | PLANE   * + название пункта назначения;   + номер рейса;   + тип самолета. | [-50, 50]. |
|  | WORKER   * + фамилия и инициалы работника;   + название занимаемой должности;   + год поступления на работу. | [-120, 50] |
|  | TRAIN   * + название пункта назначения;   + номер поезда;   + время отправления. | [-5, 100] |
|  | TRACK   * + название начального пункта маршрута;   + название конечного пункта маршрута;   + номер маршрута. | [-10, 25] |
|  | NOTE   * + фамилия и имя;   + номер телефона;   + дата рождения (день, месяц, год) | [-10, 40]. |
|  | PRICE   * + название товара;   + название магазина, в котором продается товар;   + стоимость товара. | [-20, 100] |
|  | DISK   * + название альбома;   + исполнитель;   + год выхода. | [-10, 75]. |
|  | ORDER   * + расчетный счет плательщика;   + расчетный счет получателя;   + перечисляемая сумма в рублях. | [-20, 10]. |
|  | CLIP   * + название видео;   + автор;   + количество просмотров. | [-20, 70] |
|  | CLIP   * + название композиции;   + автор;   + длительность в сек. | [-45, 30] |
|  | GAME   * + название игры;   + жанр игры;   + рейтинг. | [-20, 70]. |
|  | SPORT   * + название вида;   + фамилия рекордсмена;   + дата установление рекорда (день, месяц, год) | [-100, 50] |
|  | RASP   * + направление поезда;   + время отправления поезда;   + общая протяженность пути; | [-100, 100] |
|  | DOGS   * + кличка собаки;   + порода собаки;   + стоимость. | [-10, 85] |
|  | MOVIE   * + название;   + жанр;   + режиссёр;   + год выхода картины. | [-65, 150] |
|  | BLOCKBASTER   * + название;   + жанр;   + режиссёр;   + кассовые сборы. | [-25, 70]. |
|  | DOC   * + номер строки документа;   + текст строки;   + количество символов. | [-20, 50] |
|  | LIBRARY   * + фамилия и инициалы автора книги;   + название книги;   + год издания;   + жанр. | [-5, 10] |
|  | BOOKS   * + фамилия и инициалы автора книги;   + название книги;   + жанр;   + количество страниц. | [-10, 45]. |
|  | GAMER   * + название игры;   + жанр игры;   + год выпуска. | [-30, 95]. |
|  | CATS   * + кличка кошки;   + порода кошки;   + окрас;   + стоимость. | [-35, 100] |
|  | TRADE   * + вид товара;   + наименование товара;   + выручка (в руб.); | [-50, 60] |
|  | PROFIT   * + вид товара;   + наименование товара;   + количество продаж (в шт.). | [-35, 45] |