# Глава 16.    Алгоритмы поиска и выбора (на примере массивов)

·        [Постановка задач поиска](http://abc.vvsu.ru/Books/ebooks_iskt/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA%D0%B8/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BD%D0%B0%20%D0%A1/glava16.htm#h0)

·        [Последовательный поиск](http://abc.vvsu.ru/Books/ebooks_iskt/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA%D0%B8/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BD%D0%B0%20%D0%A1/glava16.htm#h1)

·        [Бинарный поиск](http://abc.vvsu.ru/Books/ebooks_iskt/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA%D0%B8/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BD%D0%B0%20%D0%A1/glava16.htm#h2)

·        [М-блочный поиск](http://abc.vvsu.ru/Books/ebooks_iskt/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA%D0%B8/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BD%D0%B0%20%D0%A1/glava16.htm#h3)

·        [Массивы и функции хеширования, методы вычисления адреса](http://abc.vvsu.ru/Books/ebooks_iskt/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA%D0%B8/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BD%D0%B0%20%D0%A1/glava16.htm#h4)

·        [Постановка задач выбора и методы их решения](http://abc.vvsu.ru/Books/ebooks_iskt/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA%D0%B8/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F/%D0%A2%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BD%D0%B0%20%D0%A1/glava16.htm#h5)

## 16.1.  Постановка задач поиска

·        ***Задачи поиска.*** Пусть заданы линейные списки: список элементов В= { К1, К2, К3, ..., КN} и список ключей V= { V1, V2, ..., VM } (в простейшем случае это целые числа). Требуется для каждого значения Vi из V найти множество всех совпадающих с ним элементов из В.

Чаще всего встречается ситуация когда V содержит один элемент, а в В имеется не более одного такого элемента.

·        ***Эффективность некоторого алгоритма поиска*** А оценивается максимальным Max{А} и средним Avg{А} количествами сравнений, необходимых для нахождения элемента V в В. Если Pi - относительная частота использования элемента Кi в В, а Si - количество сравнений, необходимое для его поиска, то   
 Max{А} = max{ Si, i=1,N } ; Avg{А} = Pi Si .

## 16.2.  Последовательный поиск

·        ***Последовательный поиск*** предусматривает последовательный просмотр всех элементов списка В в порядке их расположения, пока не найдется элемент равный V. Если достоверно неизвестно, что такой элемент имеется в списке, то необходимо следить за тем, чтобы поиск не вышел за пределы списка, что достигается использованием стоппера.

·        Очевидно, что Max последовательного поиска равен N. Если частота использования каждого элемента списка одинакова, т.е. P=1/N, то Avg последовательного поиска равно N/2. При различной частоте использования элементов Avg можно улучшить, если поместить часто встречаемые элементы в начало списка.

Пусть во входном потоке задано 100 целых чисел К1, К2, ..., К100 и ключ V. Составим программу для последовательного хранения элементов Кi и поиска среди них элемента, равного V, причем такого элемента может и не быть в списке. Без использования стоппера (барьера, упрощающего сложное условие выхода из цикла) программа может быть реализована следующим образом:

// последовательный поиск без стоппера

#include <stdio.h>

#define N 100

void main()

{

int k[N],v,i;

for (i=0; i<N; i++) scanf("%d",&k[i]);

scanf("%d",&v);

i=0; while(k[i]!=v && i<N) i++;

if (k[i]==v) printf("%d %d",v,i);

else printf("%d не найден",v);

}

С использованием ***стоппера (барьера)*** программу можно записать в виде:

// последовательный поиск со стоппером

#include <stdio.h>

#define N 100

void main()

{

int k[N+1],v,i;

for (i=0; i<N; i++) scanf("%d",&k[i]); /\* ввод данных \*/

scanf("%d",&v);

k[N]=v; /\* стоппер \*/

i=0; while(k[i]!=v) i++;

if (i<N) printf ("%d %d",v,i);

else printf ("%d не найден",v);

}

## 16.3.  Бинарный поиск

Для ***упорядоченных линейных списков*** существуют более эффективные алгоритмы поиска, хотя и для таких списков применим последовательный поиск.

·        ***Бинарный поиск*** состоит в том, что ключ V сравнивается со средним элементом списка. Если эти значения окажутся равными, то искомый элемент найден, в противном случае поиск продолжается в одной из половин списка.

·        Нахождение элемента бинарным поиском осуществляется ***очень быстро***. Max бинарного поиска равен log2(N), и при одинаковой частоте использования каждого элемента Avg бинарного поиска равен log2(N). ***Недостаток бинарного поиска*** заключается в необходимости последовательного хранения списка, что усложняет операции добавления и исключения элементов .

Пусть, например, во входном потоке задано 101 число, К1, К2, ..., К100, V - элементы списка и ключ. Известно, что список упорядочен по возрастанию, и элемент V в списке имеется. Составим программу для ввода данных и осуществления бинарного поиска ключа V в списке К1, К2, ..., К100.

// Бинарный поиск

#include <stdio.h>

#define N 100

void main()

{

int k[N],v,i,j,m;

for (i=0;i<N;i++) scanf("%d",&k[i]);

scanf("%d",&v);

i=0; j=N; m=N/2;

while (k[m]!=v)

{

if (k[m] < v) i+=m;

else j=m-i;

m=(i+j)/2;

}

printf("%d %d",v,m);

}

## 16.4.  М-блочный поиск

***Этот способ удобен при индексном хранении списка***. Предполагается, что исходный упорядоченный список B длины N разбит на M подсписков B1, B2, ..., BM длины N1, N2, ..., NM, таким образом, что B={ B1, B2 , .., BM}.

·        Для нахождения ключа V, нужно сначала определить первый из списков Bi, i=1,M, последний элемент которого больше V, а потом применить последовательный поиск к списку Bi.

·        Хранение списков Bi может быть связным или последовательным. Если длины всех подсписков приблизительно равны и M= N, то Max М-блочного поиска равен 2 N. При одинаковой частоте использования элементов Avg М-блочного поиска равен N.

Описанный алгоритм усложняется, если не известно, действительно ли в списке имеется элемент, совпадающий с ключом V. При этом возможны случаи: либо такого элемента в списке нет, либо их несколько.

·        Если вместо ключа V имеется упорядоченный список ключей, то последовательный или М-блочный поиск ***может оказаться более удобным***, чем бинарный, поскольку не требуется повторной инициализации для каждого нового ключа из списка V.