查找长度：查找到目标元素的比较次数。

平均查找长度：在一个基于某种数据结构的元素集合中，各个元素被查找的概率和其查找长度的积的 和，

衡量一个查找算法优劣的指标：平均查找长度。

1. 顺序表查找 x

1

2

3

4

5

6

7

int

{

find\_value(int \*a, int x,int frist, int last)

int i

while if (i

= frist;

(i <= last && a[i] != x) i++;

== last + 1) 查无此数据;

else return i;

}

2.第一次改良：哨兵查找

1

2

3

4

5

6

7

8

9

10

int

{

find\_value(int \*a, int x,int frist, int last)

int tmp = a[last + 1];

a[last + 1] = x; int i = frist;

while (a[i] != x) i++; a[last + 1] = tmp;

if (i == last + 1) 查无此数据;

else return i;

}

当数组数据大于1000时，能节约一半时间！ 3.第二次改良：经常查找的数据放到前面 方法：增加frequency，每访问一次，frequency++；移动该元素到适当位置

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

int

{

find\_value(int \*a, int x,int frist, int last)

int i

while if (i else

{

= frist;

(i <= last && a[i].data != x) i++;

== last + 1) 查无此数据;

a[i].frequency++;

int j = i;

while (j <= frist + 1)

{

if (a[j].frequency > a[j ‐ 1].frequency) swap(a[j], a[j ‐ 1]);

j‐‐; return i;

}

|  |  |
| --- | --- |
| 16  17 | }  } |
|  |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

有序数组，数组中的元素查找概率相等 采用二分查找

1 Status Binsearch(int a[], int len, int e, int \*pos) 2 {

1. if (!a) return error;//特殊值
2. int low = 0, high = len ‐ 1;
3. if(e < a[low] || e > a[high])
4. return error;
5. while (low <= high)

8 {

9 if (a[low] == e)

10 {

1. \*pos = low + 1;
2. return ok;

13 }

14 if (a[high] == e)

15 {

1. \*pos = high + 1;
2. return ok;

18 }

19 int mid = (int)(low + (high ‐ low) / 2);

20 //用mid = (low+high)/2不好，容易溢出，超出int表示的范围

21 if (a[mid] == e)

22 {

1. \*pos = mid + 1;
2. return ok;

25 }

26 else if (a[mid] < e)

27 {

28 low = mid + 1; 29 }

30 else

31 {

32 high = mid ‐ 1; 33 }

34 }

1. if(low>high)
2. return error;//找不到 37 }

【拓展】对 1 2 3 3 3 5 6 8

①如果查找 2 ：查找结果是哪个3是随机的

②如果查找 7：循环结束，low = 6的位置，high = 5的位置

根据①②可发散以下算法：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

静态次优查找树

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| low |  | i |  |  |  |  |  |  | high |

自定义 sw[i] = w[low] + w[low+1] + ......+w[i­1] + w[i] i 位置两侧元素被查找概率和的差可表示成：

△p =| sw[high] ­ sw[i] ­ (sw[i­1] ­sw[low­1] | = | sw[high] + sw[low­1] ­ sw[i] ­ sw[i­1] |