数据结构

查找

创客学院 小美老师

查找概念

- 设记录表L=(R_1 R_2……R_n)，其中R_i(l≤i≤n)为记录，对给定的某个值k，在表L中确定key=k的记录的过程，称为查找。
- 若表L中存在一个记录R_i的key=k，记为Ri.key=k，则查找成功，返回该记录在表L中的序号i(或Ri 的地址)，否则(查找失败)返回0(或空地址Null)。

查找方法

查找方法有顺序查找、折半查找、分块查找、Hash表查找等等。查找算法的优劣将影响到计算机的使用效率，应根据应用场合选择相应的查找算法。

查找-平均查找长度

对查找算法，主要分析其**T(n)**。查找过程是**key**的比较过程，时间主要耗费在各记录的**key**与给定**k**值的比较上。比较次数越多，算法效率越差（即T(n)量级越高），故用“比较次数”刻画算法的**T(n)**。

一般以“平均查找长度”来衡量**T(n)**。

查找-平均查找长度

- 平均查找长度**ASL**（Average Search Length）：对给定**k**，查找表**L**中记录比较次数的期望值(或平均值)，即：

$$ASL = \sum_{i=1}^{n} P_i C_i$$

P_i为查找R_i的概率。等概率情况下$P_i=1/n$；C_i为查找R_i时key的比较次数(或查找次数)。

顺序表的查找

顺序表，是将表中记录(R_1 R_2……R_n)按其序号存储于一维数组空间

记录Ri的类型描述如下：

```
typedef struct
 { keytype key;     //记录key//
                ……    //记录其他项//
 } Retype;
```

顺序表的查找

顺序表类型描述如下:

```
#define maxn 1024     //表最大长度//
typedef struct
{   Retype data[maxn];  //顺序表空间//
    int len;   //当前表长，表空时len=0//
} sqlist;
```

- 若说明：sqlist r，则（r.data[1]，……，r.data[r.len]）为记录表(R_1……R_n)， R_i.key为r.data[i].key，r.data[0]称为监视哨，为算法设计方便所设。

顺序查找算法及分析

算法思路 设给定值为k，在表(R_1 R_2……R_n)中，从R_n开始，查找key=k的记录。

```
int sqsearch(sqlist r, keytype k)
{   int i;
    r.data[0].key = k;  //k存入监视哨//
     i = r.len;  //取表长//
    while(r.data[i].key != k) i--;
    return (i);
}
```

顺序查找算法及分析

设$C_i(1 \le i \le n)$为查找第i记录的key比较次数(或查找次数)：

若r.data[n].key = k，　　$C_n=1$；

若r.data[n-1].key = k，　　$C_{n-1}=2$；

……

若r.data[i].key = k，　　$C_i=n-i+1$；

……

若r.data[1].key = k，　　$C_1=n$

顺序查找算法及分析

- 故ASL = O(n)。而查找失败时，查找次数等于n+l，同样为O(n)。
- 对查找算法，若ASL=O(n)，则效率是很低的，意味着查找某记录几乎要扫描整个表，当表长n很大时，会令人无法忍受。

折半查找算法及分析

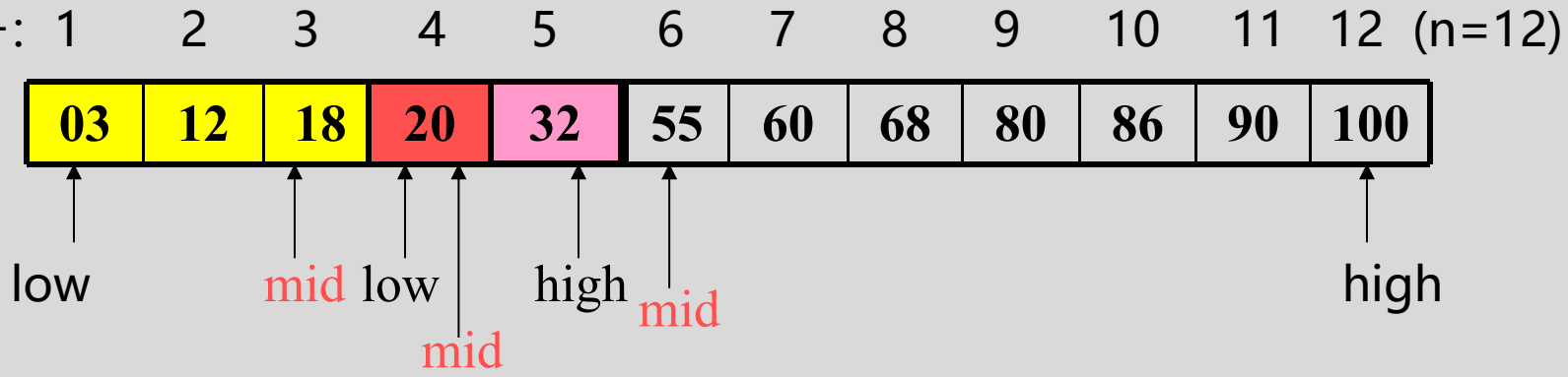
算法思路

对给定值k，逐步确定待查记录所在区间，每次将搜索空间减少一半(折半)，直到查找成功或失败为止。

设两个游标low、high，分别指向当前待查找表的上界(表头)和下界(表尾)。mid指向中间元素。

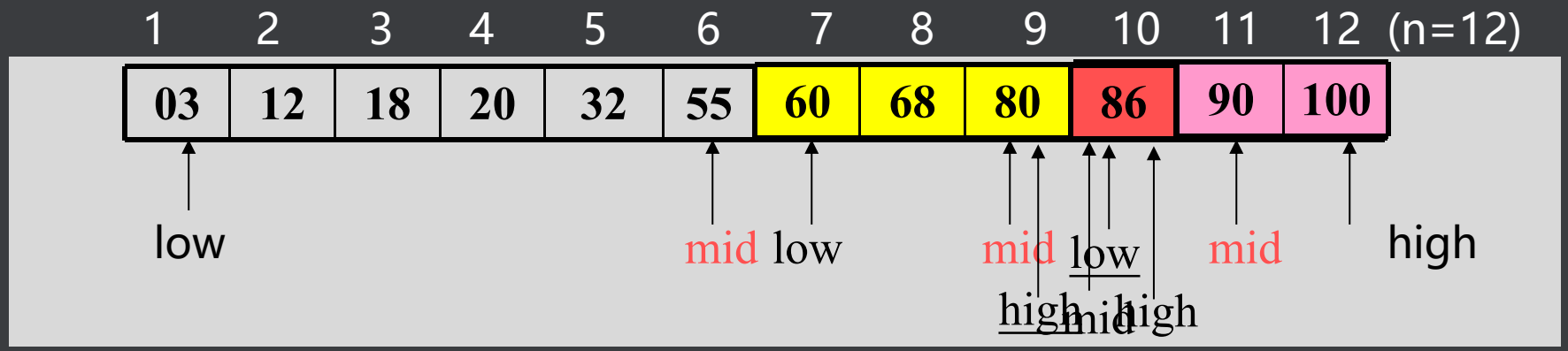
折半查找算法及分析

设记录表的key序列如下:



现查找k=20的记录。

再看查找失败的情况，设要查找k=85的记录。

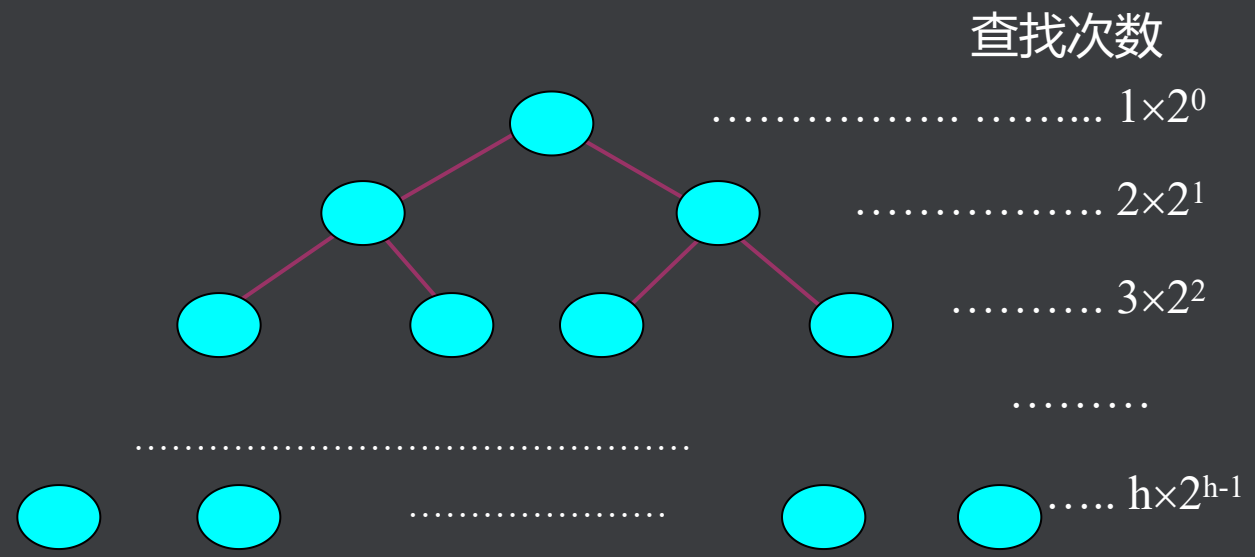


折半查找算法及分析

```
int Binsearch(sqlist r, keytype k)    //对有序表r折半查找的算法//
{  int low, high, mid;  low = 1;high = r.len;
    while (low <= high)
    {  mid = (low+high) / 2;
        if (k == r.data[mid].key)  return (mid);
        if (k < r.data[mid].key) high = mid-1;
        else low = mid+1;
    }
     return(0);
 }
```

折半查找算法及分析

不失一般性，设表长$n=2^h-1$，$h=\log_2(n+1)$。记录数n恰为一棵**h**层的满二叉树的结点数。得出表的判定树及各记录的查找次数如图所示。



$$ASL=\sum_{i=1}^{n}P_iC_i=\frac{1}{n}\sum_{i=1}^{h}i\cdot 2^{i-1}$$

令$S=\sum_{i=1}^{h}i\cdot 2^{i-1}=1\cdot 2^0+2\cdot 2^1+3\cdot 2^2+\ldots\ldots+(h-1)2^{h-2}+h\cdot 2^{h-1}$

$$2S=1\cdot 2^1+2\cdot 2^2+3\cdot 2^3+\ldots\ldots+(h-1)2^{h-1}+h\cdot 2^h$$

$$S=2S-S=h\cdot 2^h-(2^0+2^1+2^2+\ldots\ldots+2^{h-1})=h\cdot 2^h-(2^h-1)=(n+1)\log_2(n+1)-n$$

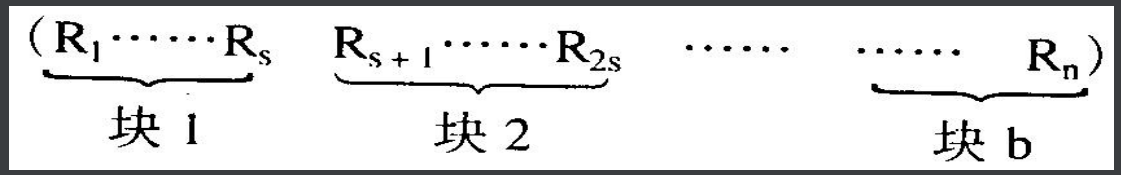
$$故ASL=\frac{1}{n}((n+1)\log_2(n+1)-n)=\frac{n+1}{n}\log_2(n+1)-1$$

$n\to\infty$ 时，$ASL=O(\log_2(n+1))$，大大优于$O(n)$。

分块查找算法及分析

分块

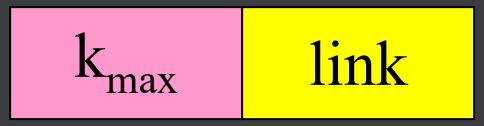
设记录表长为n，将表的n个记录分成b=$\lceil n/s \rceil$个块，每块s个记录（最后一块记录数可以少于s个），即：



且表分块有序，即第i（1≤i≤b-1）块所有记录的key小于第i+1块中记录的key，但块内记录可以无序。

分块查找算法及分析

- **建立索引**
- 每块对应一索引项：
- 其中k_{max}为该块内记录的最大key；link为该块第一记录的序号（或指针）。



分块查找(索引顺序查找)

设表长n=19，取s=5，b=$\lceil 19/5 \rceil$ = 4，

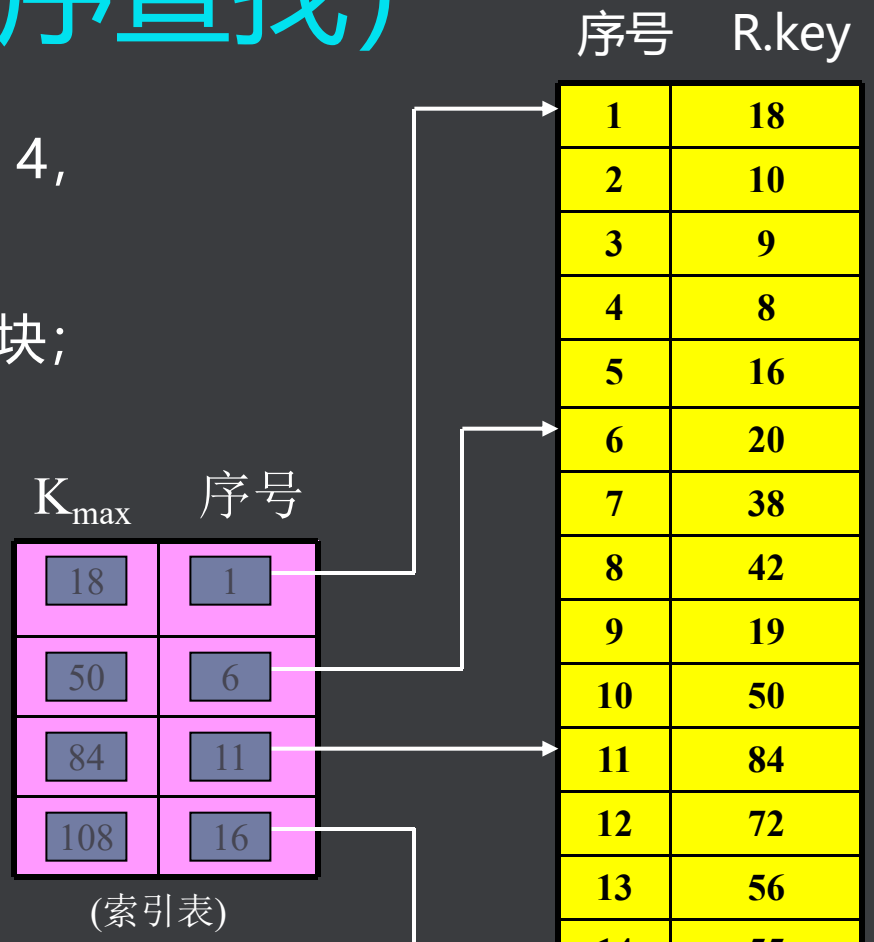
分块索引查找分两步进行：

(1)由索引表确定待查找记录所在的块；

(2)在块内顺序查找。

如查找k=19的记录

索引表是按照k_{max}有序的，可对其折半查找。而块内按顺序方法查找。



序号	R.key
1	18
2	10
3	9
4	8
5	16
6	20
7	38
8	42
9	19
10	50
11	84
12	72
13	56
14	55
15	76
16	100
17	90
18	88
19	108

总结

- 顺序、折半、分块查找和树表的查找中，其ASL的量级在O(n)~O(log2n)之间。
- 不论ASL在哪个量级，都与记录长度n有关。随着n的扩大，算法的效率会越来越低。
- ASL与n有关是因为记录在存储器中的存放是随机的，或者说记录的key与记录的存放地址无关，因而查找只能建立在key的“比较”基础上。

扫一扫，获取更多信息

THANK YOU