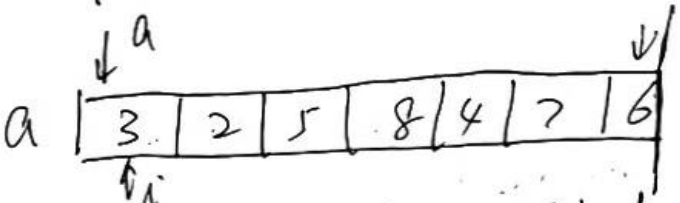
查找 { 静态查找表 { 顺序 / 二分(折半) / 索引 }
动态查找表 { 二叉排序树 / 二叉平衡树 / 红黑树 / B树 ⇒ B+树 } (边查边插)

1. 静态查找表 — 在一个固定的数据集合上完成查找.

① 顺序查找:



key=4
key=14

查找某个关键字key.
查找成功 — 返回下标.
查找失败: -1.

```
int Search_Seq(int a[], int n)
{
    for(int i=0; i<n; ++i)
        if(a[i]==key) break;
    if(i==n) i=-1;
    return i;
}
```

改进：

```
int Search_Seq(int a[], int n)
{
    for(int i=n-1; i>=-1; --i)
        if(a[i]==key) break;
    return i;
}
```

☆ 查找的性能：平均查找长度 $ASL=\sum_{i=1}^{n}P_iC_i$

（C_i：带入记录找到概率；P_i：需要比较的次数）

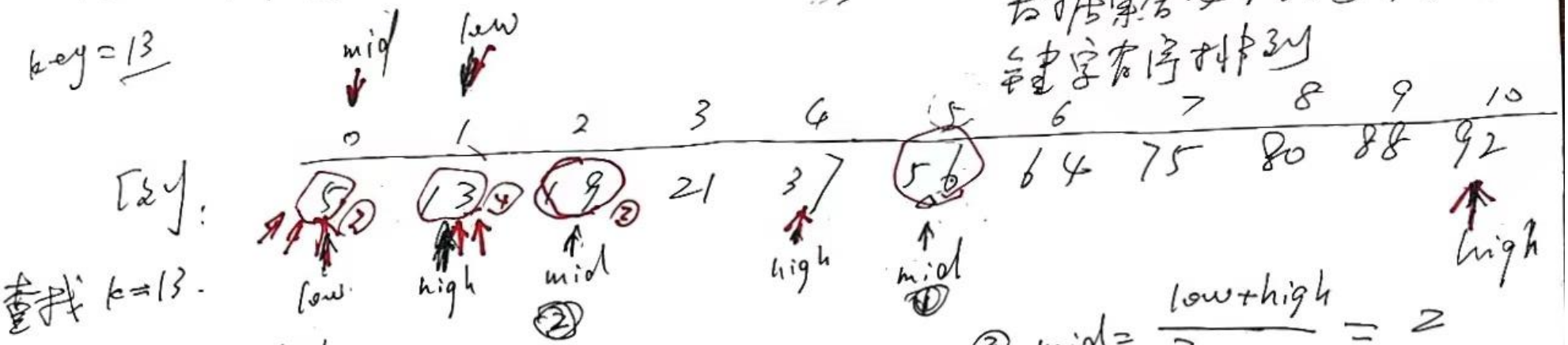
对概率：$P_i=\frac{1}{n}$

（查找成功）$ASL=\frac{1}{n}\sum_{i=1}^{n}(n-i+1)=\frac{n+1}{2}$

时间复杂度：$O(n)$　空间复杂度：$O(1)$

② 折半查找（二分查找）——只有对有序表查找

表对应集合必须是查找了关键字有序排列



查找 k=13.

① $mid=\frac{low+high}{2}=5$

a[mid]==key → Y → return mid.

N → a[mid]>key: 左二分. low=low, high=mid−1

a[mid]<key: 右二分. low=mid+1, high=high

② $mid=\frac{low+high}{2}=2$

a[mid]>key: low=low, high=mid−1

③ $mid=\frac{low+high}{2}=0$

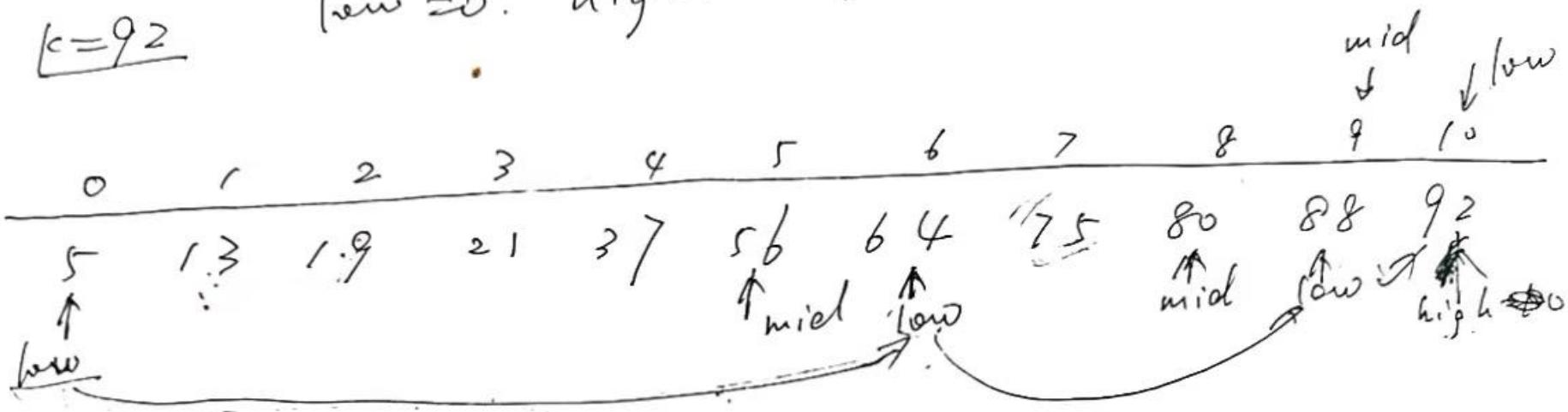
a[mid]<key: low=mid+1

④ $mid=\frac{low+high}{2}=1$

a[mid]==key ⇒ return mid

☆ low、high 不断地缩减找

k=92 low=0. high=n-1



① $mid = \frac{low+high}{2} = 5$

$a[mid] < key \xrightarrow{左=5} low = mid+1$

② $mid = \frac{low+high}{2} = 8$

$a[mid] < key \xrightarrow{左=5} low = mid+1$

③ $mid = \frac{low+high}{2} = \frac{9+10}{2} = 9$

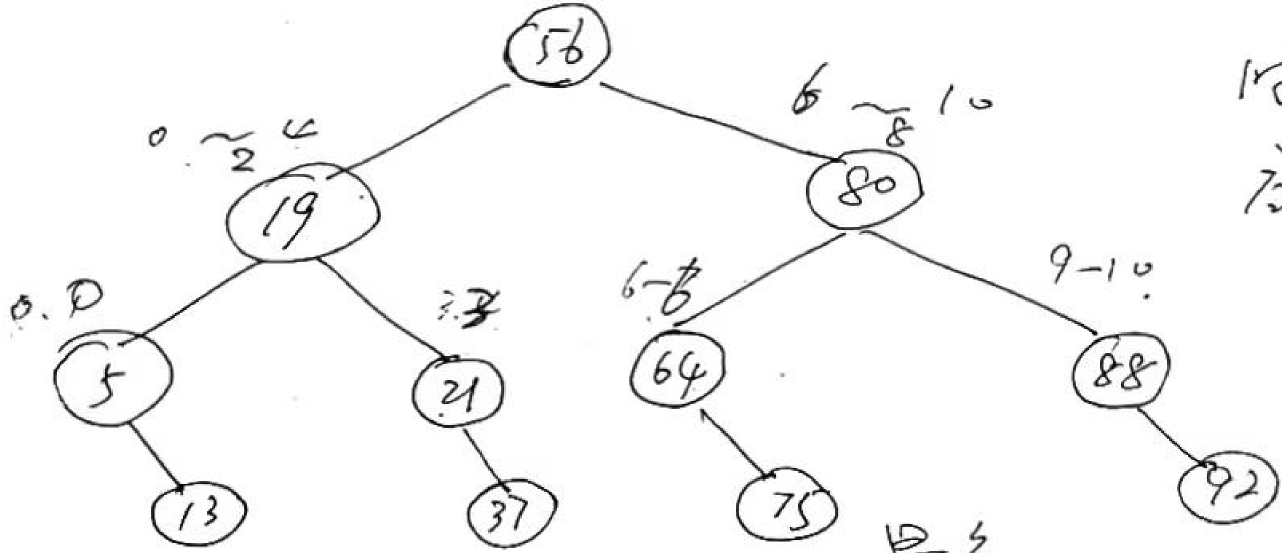
$a[mid] < key \xrightarrow{左=5} low = mid+1$

④ $mid = \frac{low+high}{2} = 10$

$a[mid] == key \Rightarrow return\ mid$

★ 二分查找可借助二叉查找树

0	1	2	3	4	5	6	7	8	9	10
5	13	19	21	37	56	64	75	80	88	92



时间：$O(\log_2 n)$

空间：$O(1)$

小结：

对任何一个关键字key的查找次数：最多 树的深度

$\lfloor \log n \rfloor + 1$.

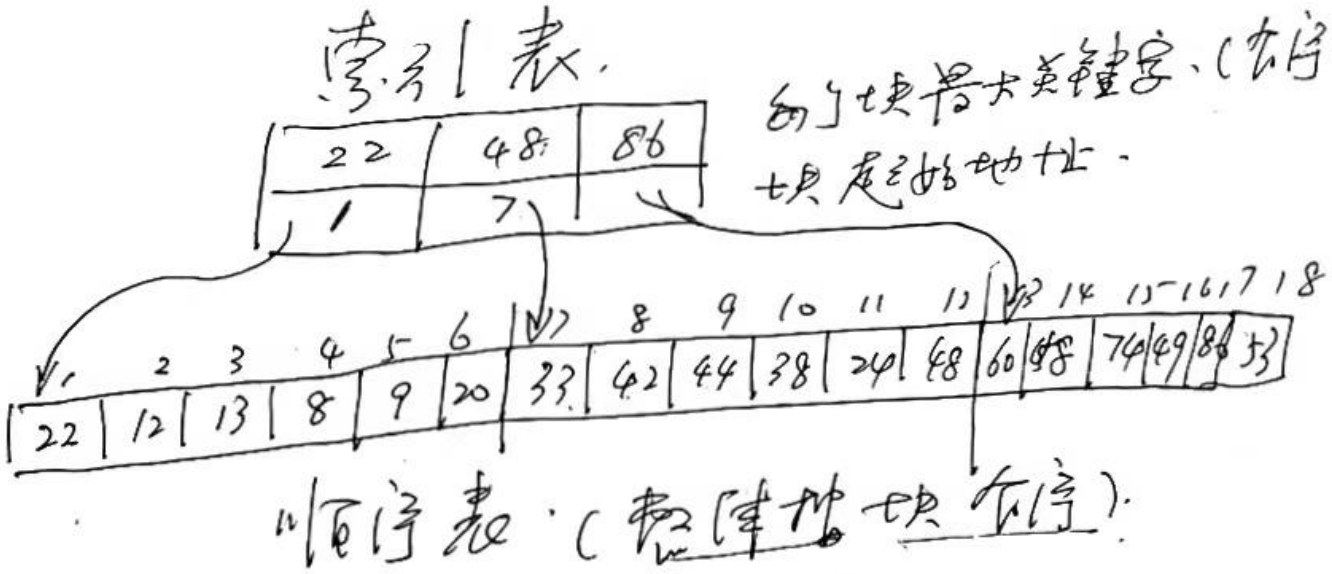
成功查找.

```
int half_search (int a[], int n) //折半查找
    {   int low=0, high=n-1;
        while(low<=high){
            int mid=(low+high)/2;
            if(a[mid]==key) return mid;
            else if(a[mid]>key) high=mid-1;
                 else low=mid+1;
        }
        return -1;
    }
```

half_search(a, 0, n-1)

```
int half_search (int a[], int low, int high)
{   if(low > high) return -1;
    else { int mid;
           mid=((low+high)/2;
           if(a[mid]==key) return mid;
           else if(a[mid]>key){
                   return half_search(a, low, mid-1);
                }
           else {
                   return half_search(a, mid+1, high);
                }
         }
}
```

索引查找（索引顺序表查找）



先在索引表以顺序查 → 确定块（索引2个）

在块内以顺序查

哈希表查找

哈希表：将关键字映射到内存中一块连续的存储空间中

① 散列表 —— 哈希表

② 哈希函数（散列函数）—— 将关键字存入散列空间的存储地址的算法

☆ 主要方法：除留余数法

$Hash(key) = key \ MOD$ 表长 ← 可以是散列表长，也可不是散列表长

地址

☆ ② 装填因子：$\alpha = \frac{元素个数}{散列表长}$ ← 访问表长

例：散列表长 17，元素个数：11

$\alpha = \frac{11}{17} = 0.65$

④ 构建哈希（散列）函数的要素：

A> 计算简单，较低的复杂度。

B> 地址空间分布均匀，降冲突抑制在最小范围。

（关键字在进行地址映射时分散程度）。

⟹ 装填因子。

⑤ 方法：除留余法 ＜ 装填因子 / 关键字复合

⑤ 散列查找：以某个组织形式的表层时

用顺序查找法可进查找散列空间

冲突
- 同义词冲突：两个关键字一个Hash值
- 异义词冲突：

解决冲突
- 线性探查法：缺点：容易产生局部堆积
- 拉链法

★ 平均查找长度是衡量哈希表查找效率的指标.

需要比较的关键字值

影响因素：⟹ 哈希函数、装填因子、处理冲突的方法

★ 处理冲突的方法相同的哈希表，平均查找长度依赖于装填因子α. α越小——发生冲突可能性越小

★ 平均查找长度 ≈ $\frac{1}{\alpha}\ln(1-\alpha)$，是α的函数而不是n的函数.

即α影响平均查找长度，n不影响，

——无论n多大，都可以找到一个合适的α ——使AVL限定在一个范围内

哈希表查找 —— 动态查找表.(也查也插)

查找集合: {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}

哈希函数: H(key) = key MOD 13

哈希空表:

0	1	2	3	4	5	6	7	8	9	10	11	12
	14	1	68	27	55	19	20	84	79	23	11	10

19%13=6. (1)　14%13=1 (1).　23%13=10 (1)

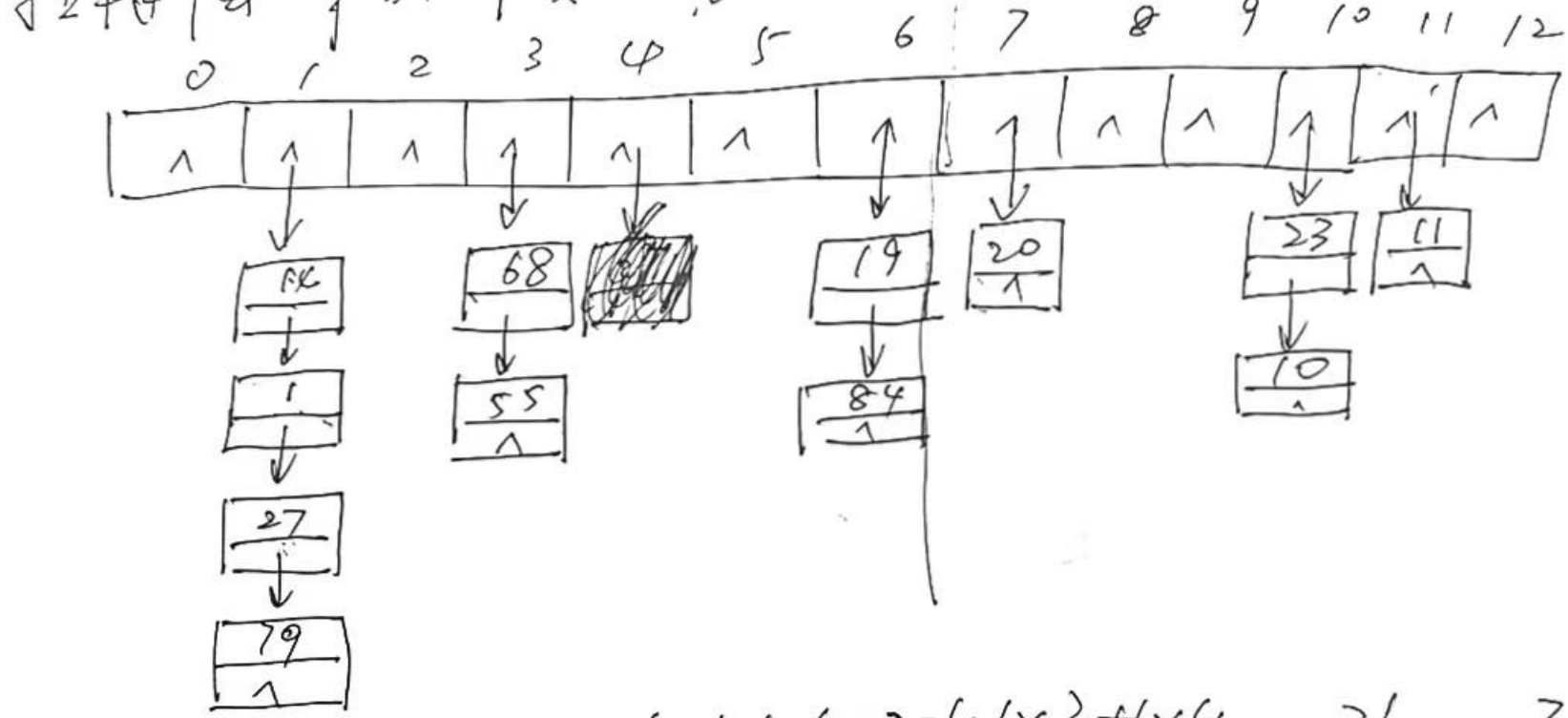
1%13=1 (冲突) 2.　68%13=3 (1)　20%13=7 (1)

84%13=6.7.8.　27%13=1.2.3.4　55%13=3.4.5

11%13=11 (1).　10%13=10.11.12　79%13=1.2.3.4.5.6.7.8.9

查找成功的平均查找长度: ASL = (6×1+1×2+3×3+4×1+9)/12 = 30/12 = 5/2 = 2.5

拉链法解决冲突：{19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}



查找成功 $ASL = \frac{6\times1+4\times2+1\times3+1\times4}{12} = \frac{21}{12} = \frac{7}{4}$

2010-41

数据集合 {7, 8, 30, 11, 18, 9, 14}. 21

Hash函数: H(key)=(key * 3) MOD 7

装填因子 $= \frac{\text{表中填入的记录数}}{\text{表的长度}} = \frac{7}{10} = 0.7$

0	1	2	3	4	5	6	7	8	9
7	14		8		11	30	18	9	

H(7)=7×3%7=0. (1)

H(30)=30×3%7=6. (1).

H(18)=18×3%7=5.6.7

H(14)=14×3%7=0.1.

H(8)=8×3%7=3. (1).

H(11)=11×3%7=5 (1).

H(9)=9×3%7=6.7.8

查找成功 $ASL = \frac{\text{查找次数}}{\text{元素}} = \frac{4\times1+1\times2+2\times3}{7} = \frac{12}{7}$

查找表：在MOD的除数上构造一个查找失败表。

H(key) = (key*3) MOD 7 ∈ 0~6

H(key)：0~6

MOD 7 ← 表长

0	1	2	3	4	5	6
3	2	1	2	1	5	4

$$ASL = \frac{3+2+1+2+1+5+4}{7} = \frac{18}{7}$$

2019-1-8

0	1	2	3	4	5	6	7	8	9	10
98	22	30	87	11	40	6	20	1	.	.
						↑				

H(key) = key % 7

87. 40. 30. 6. 11. 22. 98. 20

87%7=3 40%7=5 30%7=2

6%7=6 11%7=4 22%7=1

98%7=0 20%7=6

0	1	2	3	4	5	6
9	8	7	6	5	4	3

$$ASL = \frac{9+8+7+6+5+4+3}{7} = \frac{42}{7} = 6$$