查找算法

朱睿

June 13, 2019

大纲

1. 无序表的查找
   - 顺序查找
2. 有序表的查找
   - 顺序查找
   - 二分查找
   - 插值查找
   - 分块查找

顺序查找

- 遍历所有元素，直到匹配为止
- 时间复杂度：$O(N)$。平均查找时间：

$$\sum_{i=0}^{N} \frac{1}{N+1}(N-i+1) = \frac{N}{2}$$

- 一个优化：把待查元素放在 `arr[0]` 处，在从后向前遍历，每次检查可以不用检查下标合法性

直接插入排序

```
int seqSearch(int arr[], int size, const int &x) {
  arr[0] = x;
  for (int i = size; x != arr[i]; --i);
  return i;      // i 不为 0 则存在，返回下标
}
```

顺序查找

- 同无序表的顺序查找

二分查找

- **时间复杂度**：$O(\log N)$

E.g. 找 3

	1	2	3	4	5	6	7	8
	l			m				r
	l	m	r					
			lmr					

二分查找

- **时间复杂度**：$O(\log N)$

E.g. 找不到 3

	1	2	4	5	6	7	8	9
	l			m				r
	l	m	r					
			lmr					
		mr	l					

二分查找

```
int binarySearch(int arr[], int size, const int &x) {
  int l = 1, r = size, m;
  while (l <= r) {
    m = (l + r) / 2;
    if (x == arr[m])
      return m;
    if (x < arr[m])
      r = m - 1;
    else
      l = m + 1;
  }
  return 0;
}
```

插值查找

- 一个公式：

$$\mathtt{next} = \mathtt{low} + \left(\frac{\mathtt{x} - \mathtt{a[low]}}{\mathtt{a[high]} - \mathtt{a[low]}} \times (\mathtt{high} - \mathtt{low} - 1) \right)$$

- 适用条件：
  - 数据有序且分布均匀
  - 访问数据比计算费时的多

分块查找

- 分块查找表：

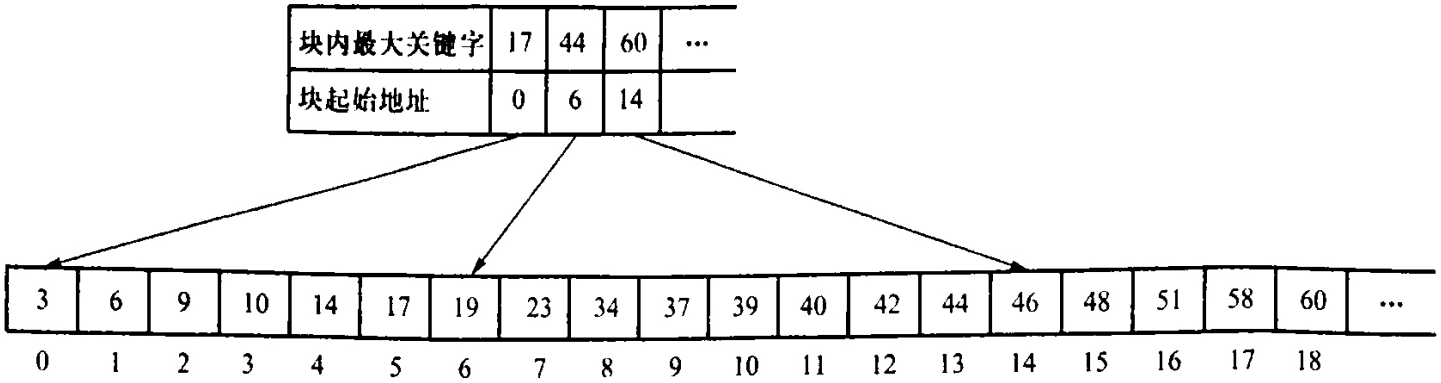

图 8-2 分块查找表示例

Source: 翁惠玉，俞勇《数据结构：思想与实现》

- 平均查找时间（当索引表与块内都是顺序查找），表长为 n，块数为 m：

$$\frac{m+1}{2}+\frac{n/m+1}{2}=\frac{1}{2}\left(m+\frac{n}{m}\right)+1$$

当 $m=\sqrt{n}$ 时，平均查找时间取得最小值 $\sqrt{n}+1$