02-D2-4 原理 & 02-D2-5 实现 & 02-D2-6 实例

#数据结构邓神

版本A：原理

减而治之：以任一元素x = S[mi] 为界限，都可以将带查找的区间分为三个部分

S[lo,mi)<=S[mi]<=S(mi,hi)

e < x left

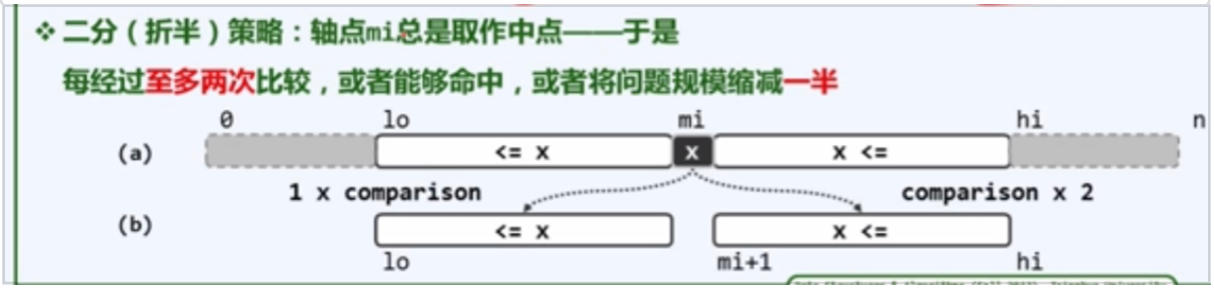
e > x right

e == x 直接返回 mi

我想到了一种简单的递归方法：就是以 e==x 为递归集，然后不断递归

其实是一种二分（折半策略）：折点总是取为重点

所以每经过至多两次的比较，或者能够命中，或者可以将问题的规模减少一半



实现

```
template <typename T> static Rank binSearch(T* A,const& e,Rank lo,Rank hi){
    while (lo < hi){ //
        Rank mi = (lo + hi) >> 1; //算出中点 >> 1 == /2 但是速度很快
        if(e < A[mi]){ // 前半段
            hi = mi;
        }
        else if(A[mi] < e){ // 后半段
            lo = mi + 1;
        }
        else{
            return mi; // 命中!
        }
    }
    return -1; // 查找失败
```

```
}
```

建议大家多使用小于号

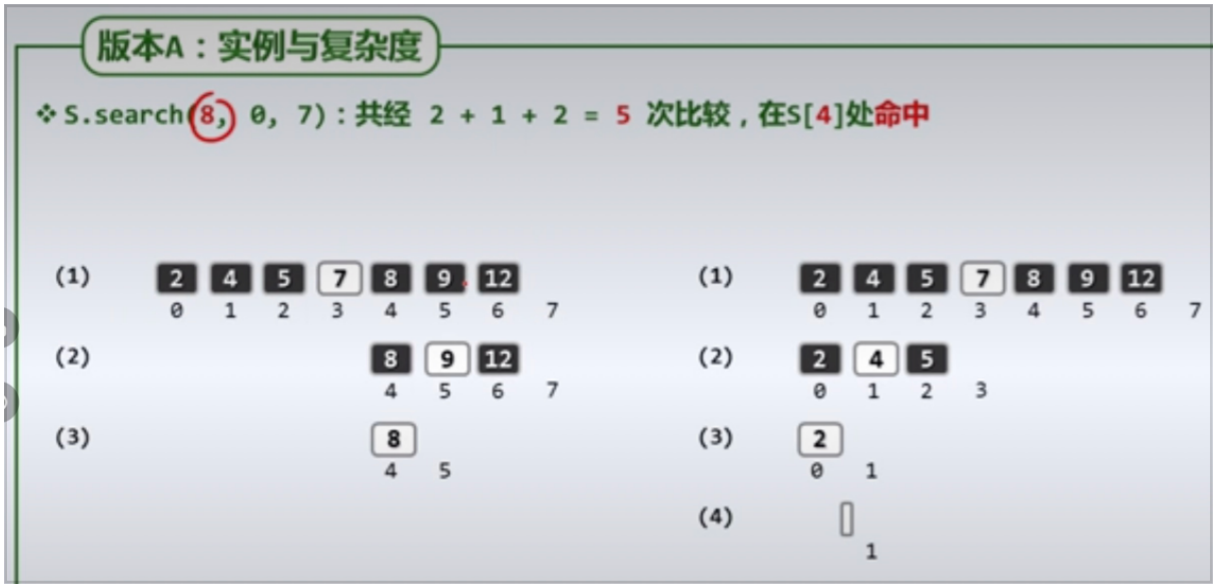
这样会更一般的认知相吻合

`e < A[mi]` 应该向左查找

而 `A[mi] < e` 就很明显可以看出应该去右侧查找

实例

```
S.search(8,0,7)
```



线性递归： `T(n) = T(n/2) + O(1) = O(logn)` 大大优于顺序查找

递归跟踪 递归总取出中点，递归深度`O(logn)`，各个递归实例均耗时为`O（1）`