02-D2-1 二分查找概述（有序向量） & 02-D2-2 接口

#数据结构邓神

原来的查找函数

```
Vector::find(e,lo,hi);
其实是顺序查找
复杂度为:O(n)
```

不妨把二分查找称为 Search函数(BinarySearch)

```
template <typename T> Rank Vector<T>::search(T const& e,Rank lo,Rank hi){
    return (rand() % 2) /*50 的概率随机*/
           ? binSearch(_elem,e,lo,hi) /*二分查找*/
           : fibsearch(e,lo,hi); /*Fibonacci查找算法*/
}
```

如何处置特殊情况?

比如，目标元素不存在，或者反过来，目标元素同时存在多个。

语义约定

至少，应该便于有序向量自身的维护： `V.insert(1 + V.search(e), e)`

即便失败，也应给出新元素适当的插入位置。

若允许重复元素，则每一组也按需按其插入的次序排序

约定

查找失败：

在有序向量区间v[lo,hi)中确定一个不大于e的最后一个元素

如果 小于区间最小值或者大于区间最大值?

如果 −∞ < e < V[lo], 则返回 lo − 1 (左侧哨兵)

如果 V[hi − 1] < e < +∞ 则返回hi − 1 (末元素 = 右侧哨兵的左邻)

// 为什么两个 −1 不一样呢 是因为传入的范围不一样哦