12.排序

（a1）快速排序：算法A

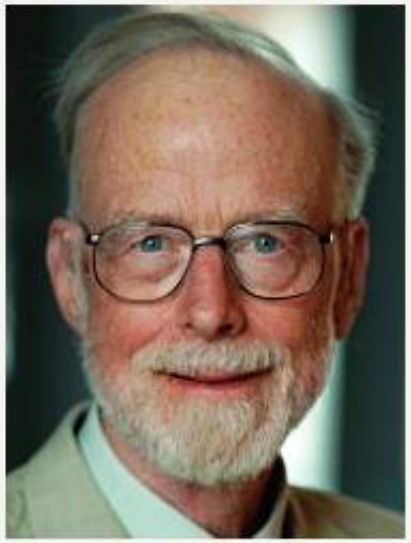
左朱雀之茷茷兮

右苍龙之躍躍

邓俊辉

deng@tsinghua.edu.cn

分而治之



C. A. R. Hoare
(1934 ~)
Turing Award, 1980

❖ 将序列分为两个子序列：S = S_L + S_R //O(n)

规模缩小：max { $|S_L|$, $|S_R|$ } < n

彼此独立：max(S_L) ≤ min(S_R)

❖ 在子序列分别递归地排序之后，原序列自然有序

sorted(S) = sorted(S_L) + sorted(S_R)

❖ 平凡解：只剩单个元素时，本身就是解

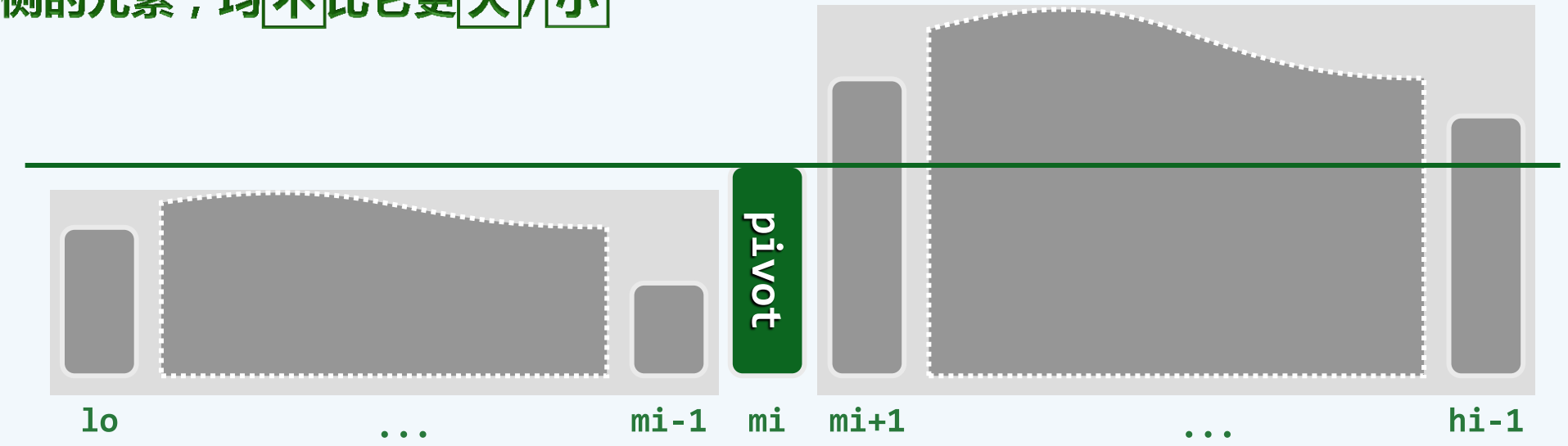
❖ mergesort的计算量和难点在于合，而quicksort在于分

//如何实现上述划分呢？

轴点

❖ pivot：

左/右侧的元素，均不比它更大/小

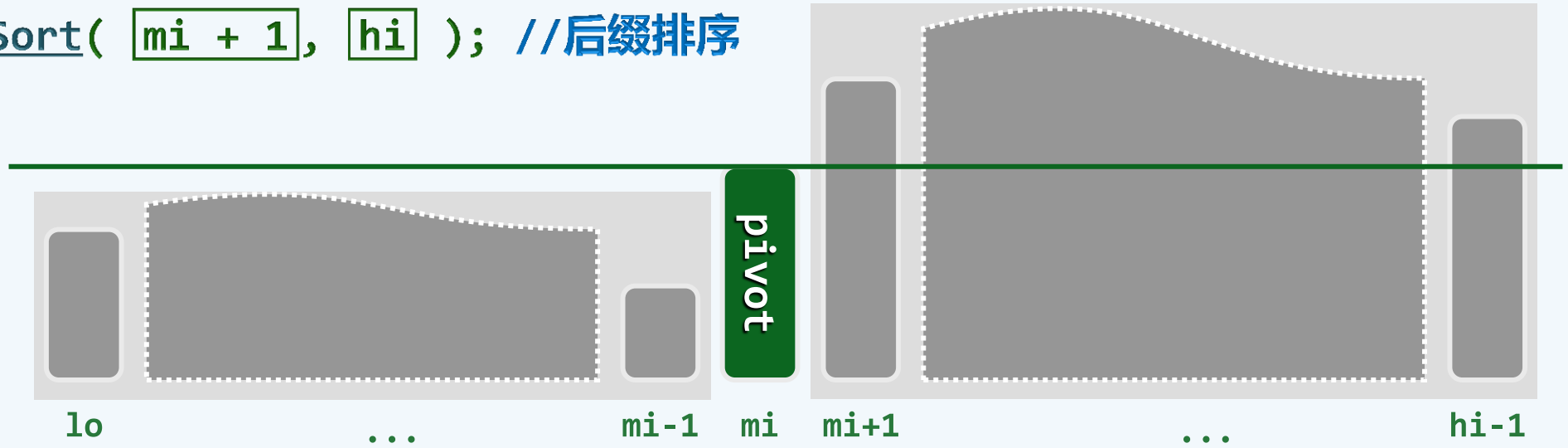


❖ 以轴点为界，原序列的划分自然实现：

```
[lo, hi)  =  [lo, mi) + [mi] + (mi, hi)
```

快速排序

```
❖ template <typename T> void Vector<T>::quickSort( Rank lo, Rank hi ) {
     if ( hi - lo < 2 ) return; //单元素区间自然有序，否则
     Rank mi = partition( lo, hi - 1 ); //先构造轴点，再
     quickSort( lo, mi ); //前缀排序
     quickSort( mi + 1, hi ); //后缀排序
  }
```

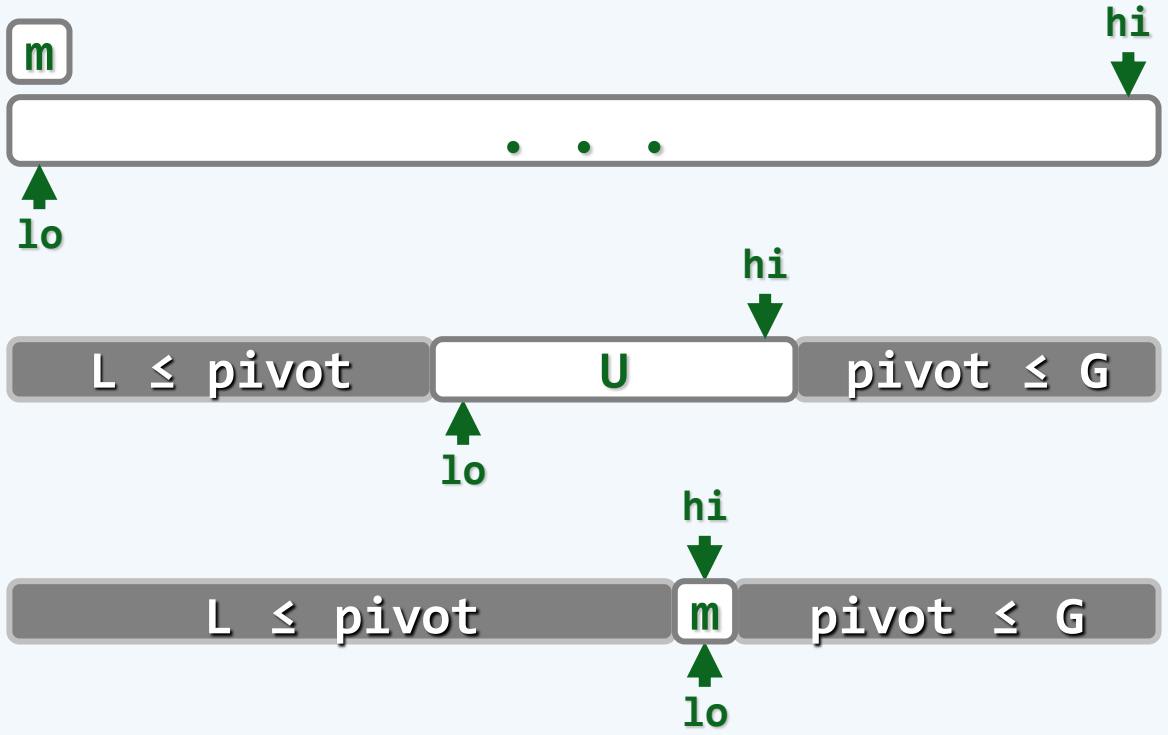


轴点

- 坏消息：在原始序列中，轴点未必存在...
- 必要条件：轴点必定已然就位 //尽管反之不然
- derangement：2 3 4 ... n 1
- 特别地：在有序序列中，所有元素皆为轴点；反之亦然
- 快速排序：就是将所有元素逐个转换为轴点的过程
- 好消息：通过适当交换，可使任一元素转换为轴点
- 问题：如何交换？成本多高？

构造轴点

- 任取一候选者（如[0]）
- 2个指针，3个子序列
  - 前缀L：≤ 候选者，初始为空
  - 后缀G：≥ 候选者，初始为空
  - 中段U：? 待确定，初始为全集

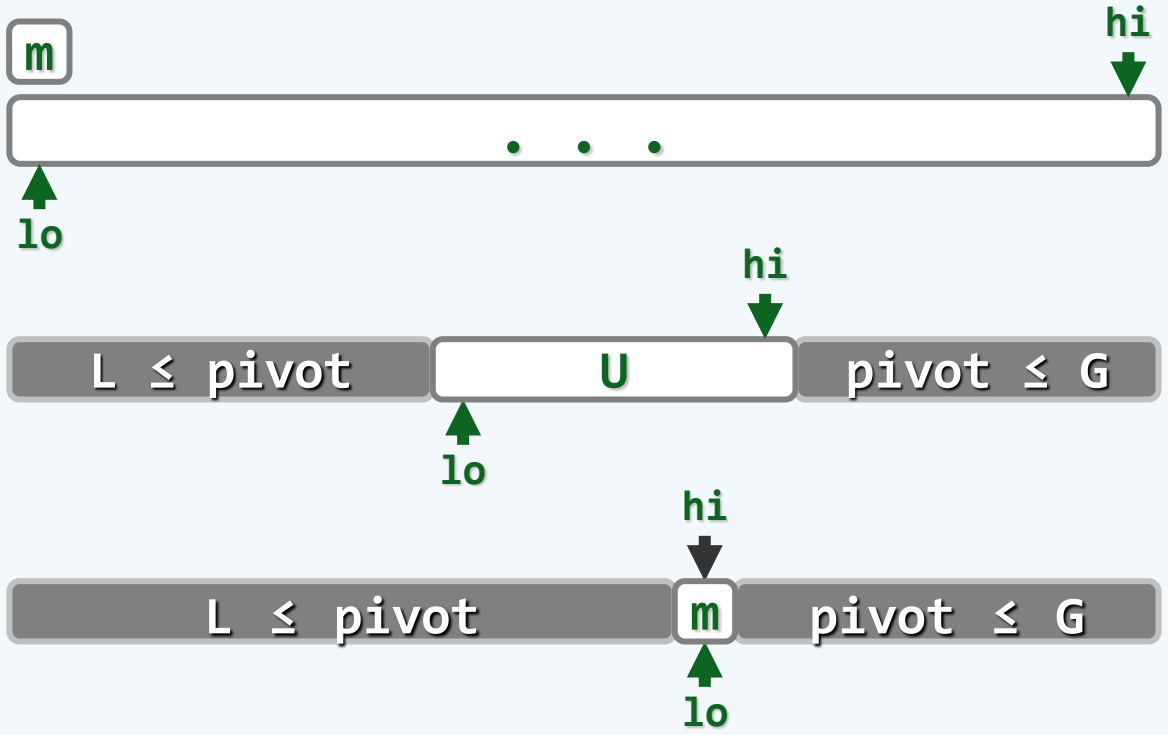


构造轴点

- 交替地向内移动lo和hi
- 逐个检查当前元素：

  若更小/大，则转移归入L/G
- 当lo = hi时

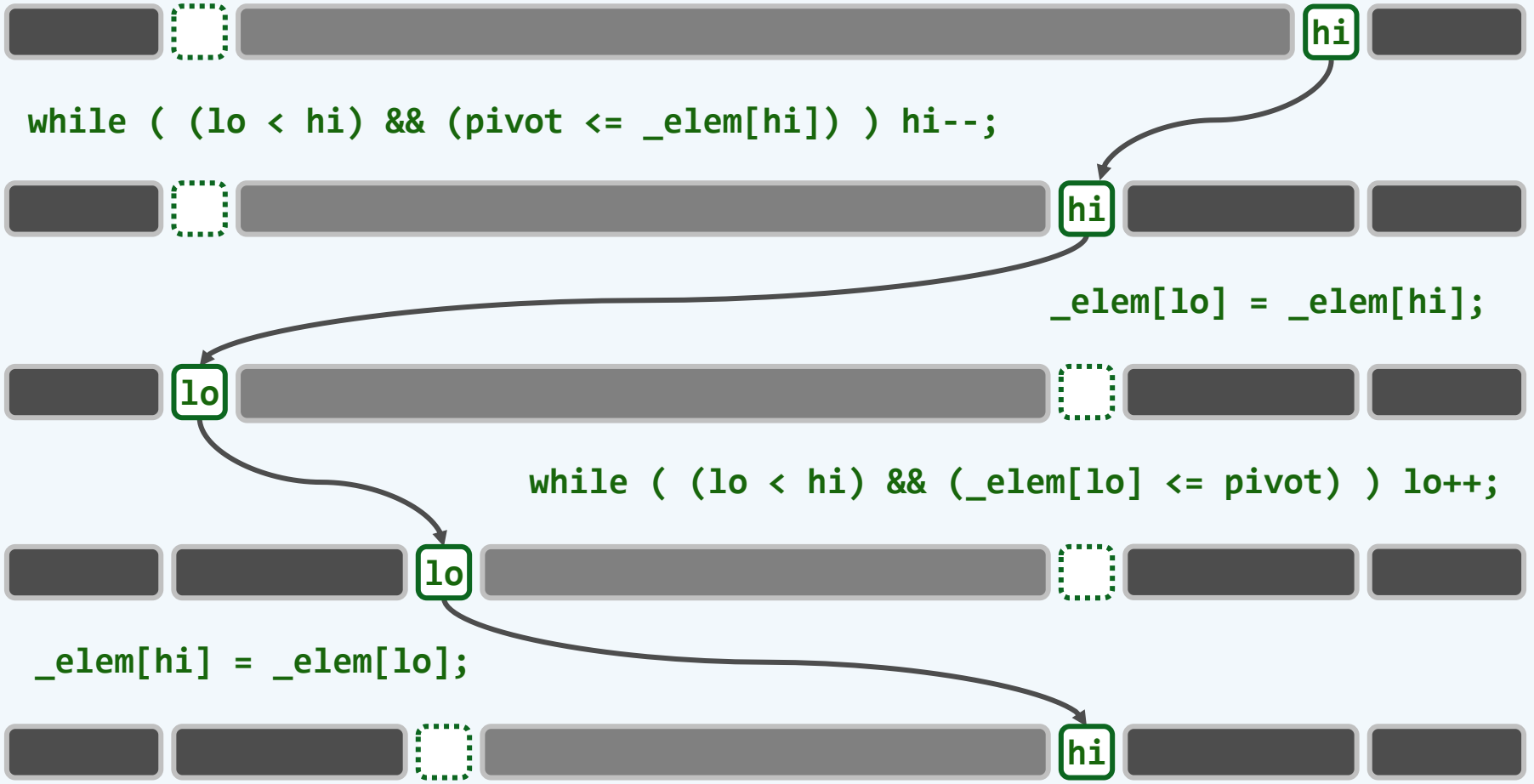
  只需将候选者嵌入于L、G之间，它即是轴点！



- 整个过程中，各元素最多移动一次（候选者两次）——累计O(n)时间、O(1)辅助空间

不变性 + 单调性

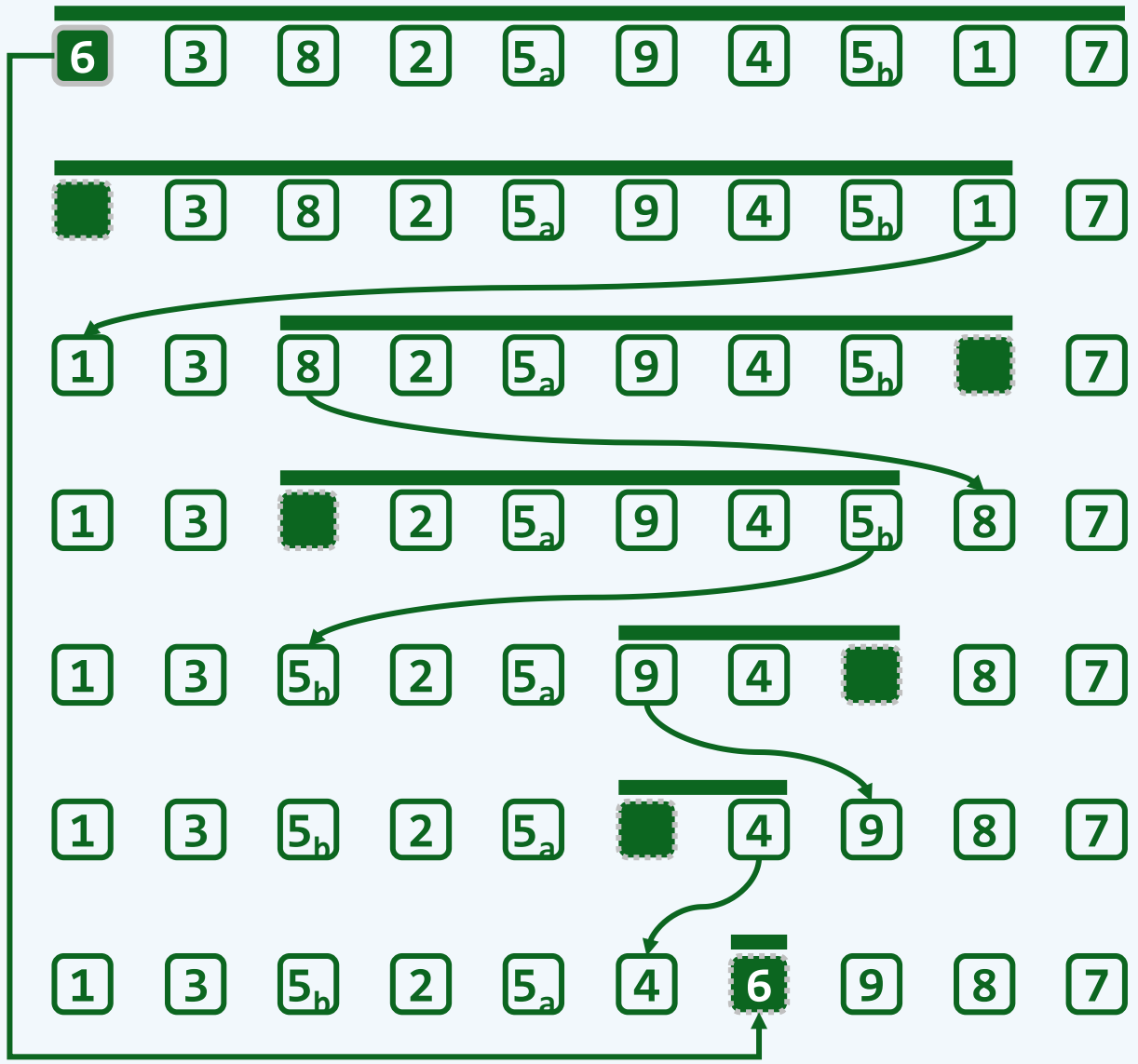
❖ L ≤ pivot ≤ G ; U = [lo, hi]中，[lo]和[hi]交替空闲



实现

```
template <typename T> Rank Vector<T>::partition( Rank lo, Rank hi ) { //[lo, hi]
   swap( _elem[ lo ], _elem[ lo + rand() % ( hi - lo + 1 ) ] ); //随机交换
   T pivot = _elem[ lo ]; //经以上交换，等效于随机选取候选轴点
   while ( lo < hi ) { //从两端交替地向中间扫描，彼此靠拢
      while ( lo < hi && pivot <= _elem[ hi ] ) hi--; //向左拓展G
      _elem[ lo ] = _elem[ hi ]; //凡小于轴点者，皆归入L
      while ( lo < hi && _elem[ lo ] <= pivot ) lo++; //向右拓展L
      _elem[ hi ] = _elem[ lo ]; //凡大于轴点者，皆归入G
   } //assert: lo == hi
   _elem[ lo ] = pivot; return lo; //候选轴点归位；返回其秩
}
```

实例


6 3 8 2 5a 9 4 5b 1 7
1 3 8 2 5a 9 4 5b 7
1 3 2 5a 9 4 5b 8 7
1 3 5b 2 5a 9 4 8 7
1 3 5b 2 5a 4 9 8 7
1 3 5b 2 5a 4 6 9 8 7