排序

快速排序：快速划分：DUP版

14-A6

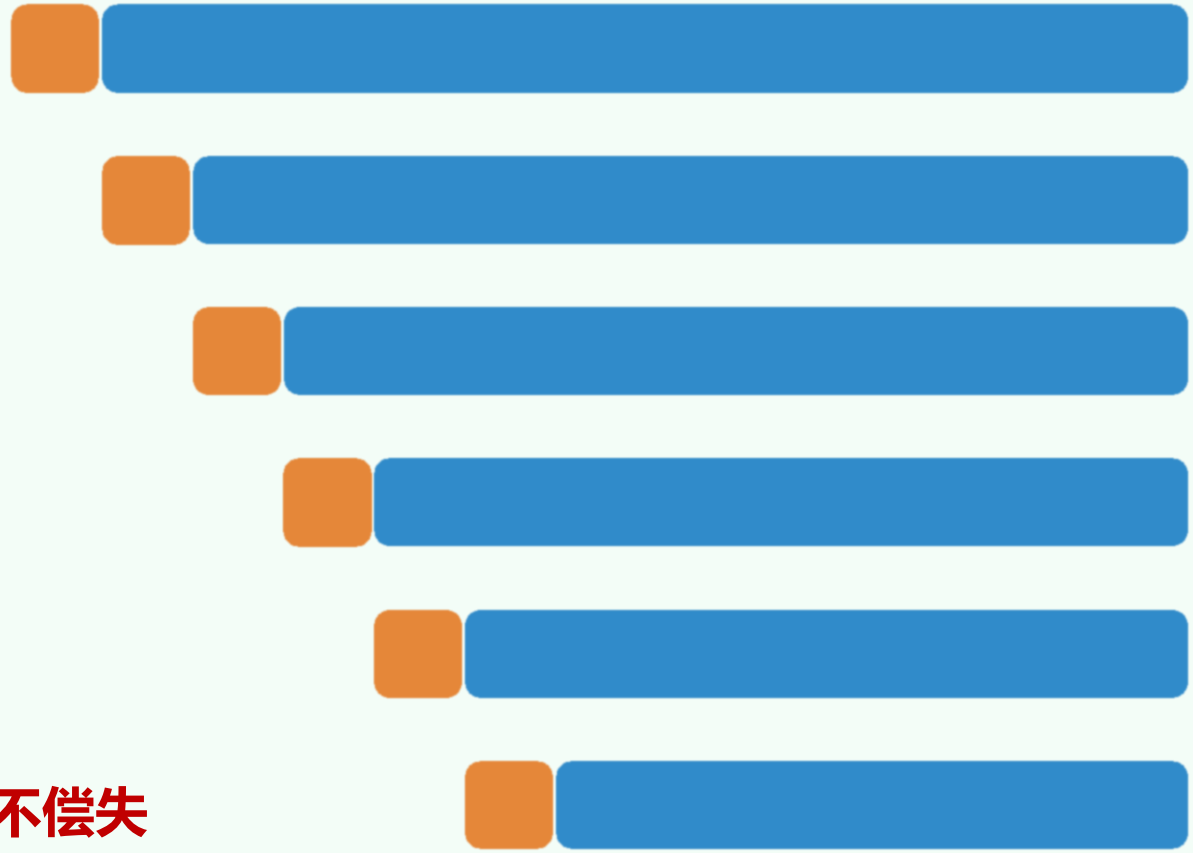
邓俊辉

deng@tsinghua.edu.cn

左见兮鸣鵙，右睹兮呼枭

重复元素

- 大量甚至全部元素重复时
  - 轴点位置总是接近于lo
  - 子序列的划分极度失衡
  - 二分递归退化为线性递归
  - 递归深度接近于$O(n)$
  - 运行时间接近于$O(n^2)$
- 移动lo和hi的过程中，同时比较相邻元素
  若属于相邻的重复元素，则不再深入递归
- 但一般情况下，如此计算量反而增加，得不偿失
- 对LUG版略做调整，即可解决问题
  ——为便于对比，以下依次给出LUG'版、DUP'版、DUP版



LUG'版

```
template <typename T> Rank Vector<T>::partition( Rank lo, Rank hi ) { //[lo, hi)
   swap( _elem[ lo ], _elem[ lo + rand() % ( hi - lo ) ] ); //随机交换
   hi--; 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
      if ( lo < hi ) _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
      if ( lo < hi ) _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
}
```

DUP'版

```
template <typename T> Rank Vector<T>::partition( Rank lo, Rank hi ) { //[lo, hi)
   swap( _elem[ lo ], _elem[ lo + rand() % ( hi - lo ) ] ); //随机交换
   hi--; 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
      if ( lo < hi ) _elem[ lo++ ] = _elem[ hi ]; //凡不大于轴点者，皆归入L
      while ( lo < hi  &&  _elem[ lo ] <  pivot ) lo++; //向右拓展L
      if ( lo < hi ) _elem[ hi-- ] = _elem[ lo ]; //凡不小于轴点者，皆归入G
   } //assert: lo == hi
   _elem[ lo ] = pivot; return lo; //候选轴点归位；返回其秩
}
```

DUP版

```
template <typename T> Rank Vector<T>::partition( Rank lo, Rank hi ) { //[lo, hi)
   swap( _elem[ lo ], _elem[ lo + rand() % ( hi - lo ) ] ); //随机交换
   hi--; 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 )
         if ( pivot < _elem[ hi ] ) hi--; //向左拓展G，直至遇到不大于轴点者
         else { _elem[ lo++ ] = _elem[ hi ]; break; } //将其归入L
      while ( lo < hi )
         if ( _elem[ lo ] < pivot ) lo++; //向右拓展L，直至遇到不小于轴点者
         else { _elem[ hi-- ] = _elem[ lo ]; break; } //将其归入G
   } //assert: lo == hi
   _elem[ lo ] = pivot; return lo; //候选轴点归位；返回其秩
}
```

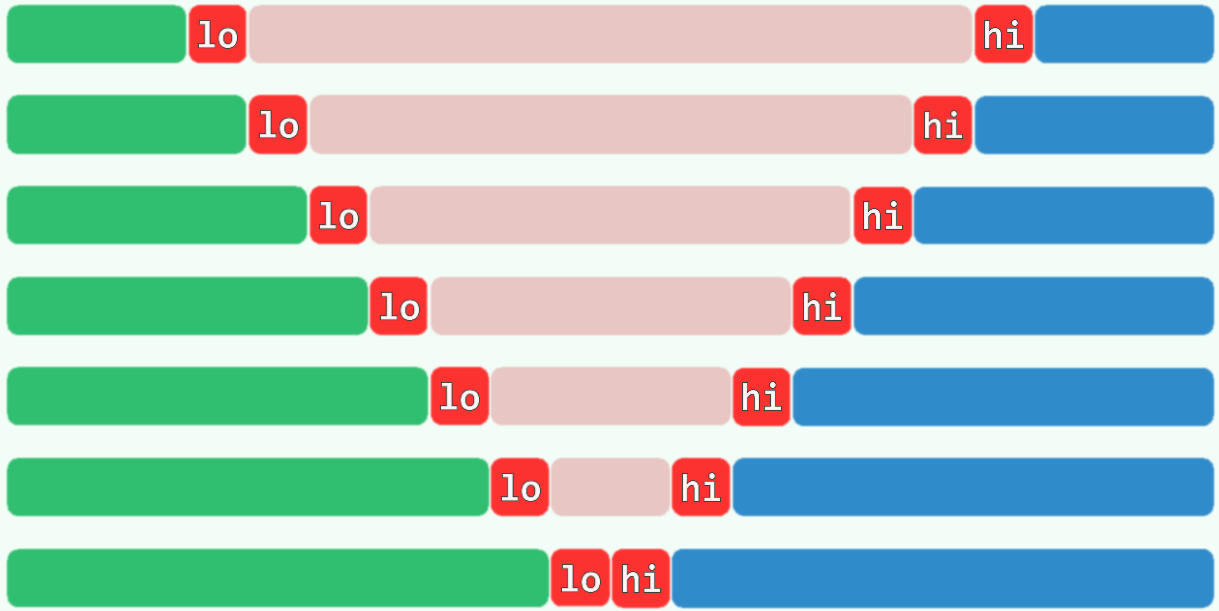
性能

❖ 可以正确地处理一般情况，同时复杂度并未实质增高

❖ 处理重复元素时

- lo和hi会交替移动
- 二者移动的距离大致相当

轴点最终被安置于(lo+hi)/2处



❖ 由LUG版的勤于拓展、懒于交换

转为懒于拓展、勤于交换

❖ 因此，交换操作有所增加——在LUG版中雷同元素相遇时并不移动——“更”不稳定