# 排序

## 快速排序：快速划分：DUP版

# 14-A6

邓俊辉

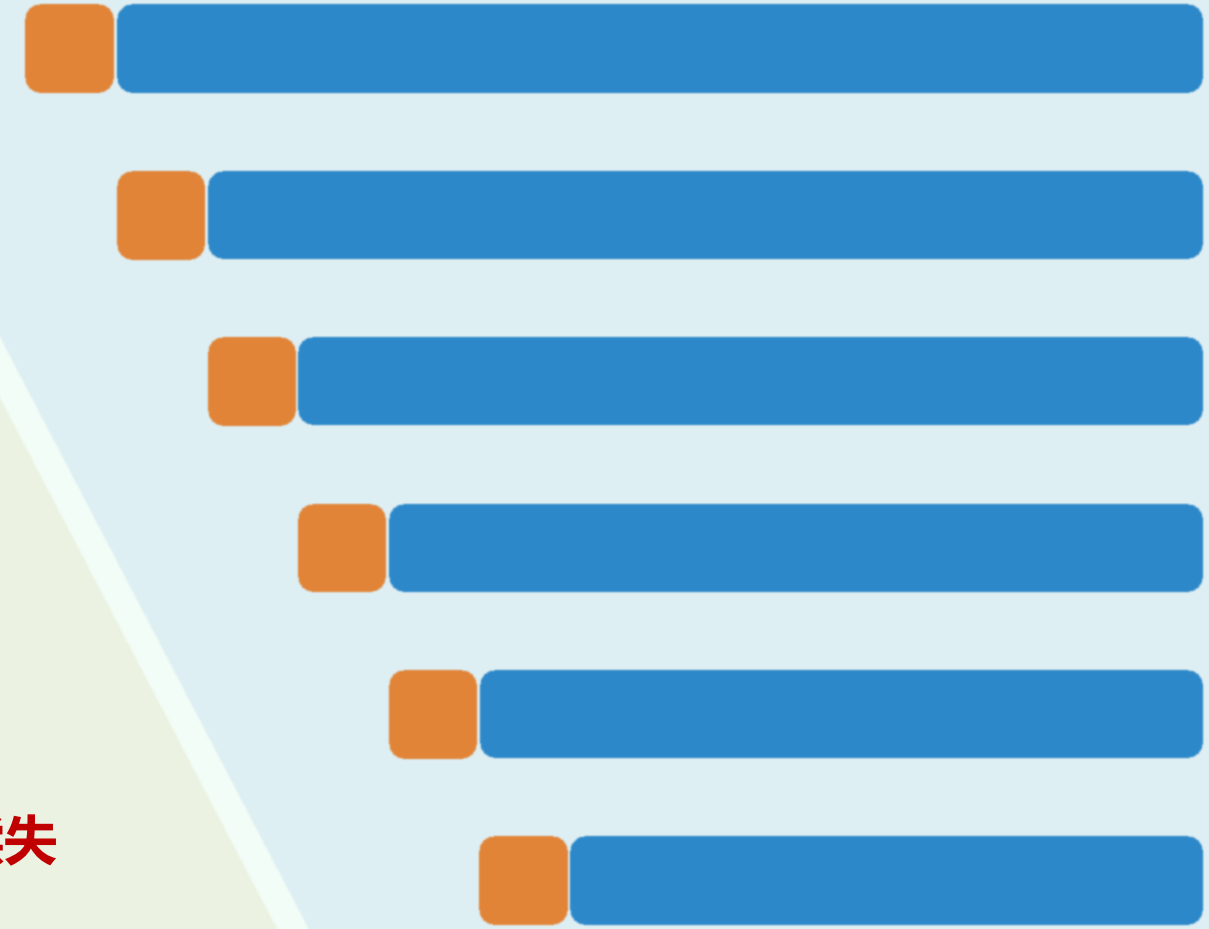
deng@tsinghua.edu.cn

左见兮鸣鵙，右睹兮呼枭

# 重复元素

- 有大量元素与轴点雷同时
  - 切分点将接近于lo
  - 划分极度失衡
  - 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
- 实际上，LUG版略做调整，即可解决问题...



# 快速划分：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
   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
      do hi--; while ( (lo < hi) && (pivot <= _elem[hi]) ); //向左拓展G
      if (lo < hi) _elem[lo] = _elem[hi]; //凡 小于 轴点者，皆归入L
      do lo++; while ( (lo < hi) && (_elem[lo] <= pivot) ); //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 or hi+1
   _elem[hi] = pivot; return hi; //候选轴点归位；返回其秩
}
```

# 快速划分：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
   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
      do hi--; while ( (lo < hi) && (pivot <  _elem[hi]) ); //向左拓展G
      if (lo < hi) _elem[lo] = _elem[hi]; //凡不大于轴点者，皆归入L
      do lo++; while ( (lo < hi) && (_elem[lo] <  pivot) ); //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 or hi+1
   _elem[hi] = pivot; return hi; //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

- 遇到连续的重复元素时
  - lo和hi会交替移动
  - 切分点接近于(lo+hi)/2

- 由LUG版的勤于拓展、懒于交换

  转为懒于拓展、勤于交换

- 交换操作有所增加，“更”不稳定