排序

快速排序：快速划分：DUP版

14-A6

邓俊辉

deng@tsinghua.edu.cn

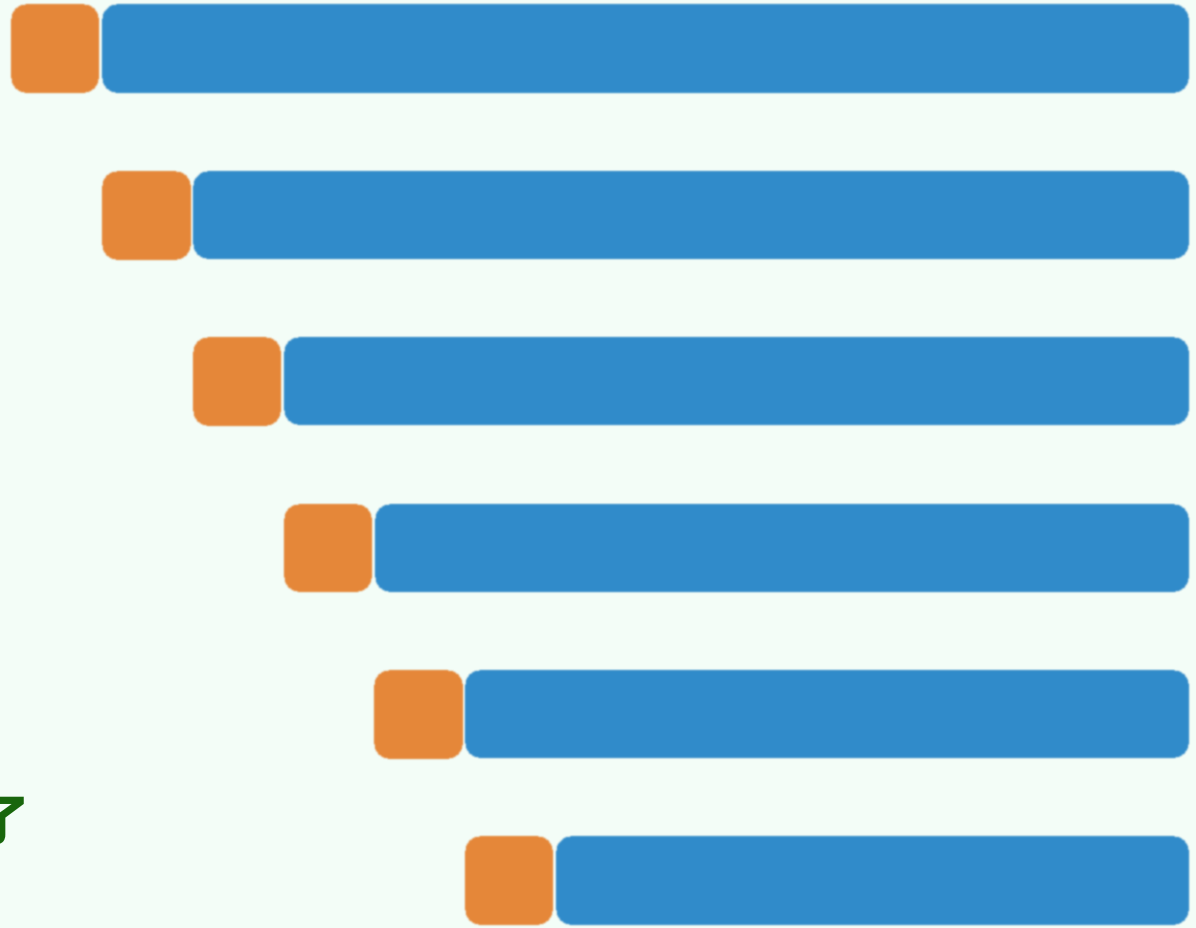
那么，一个生命的出生也就是一个世界的出生了，任何个人，都是独一无二的世界

相等元素

- ❖ 有大量元素与轴点相等时
  - 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$O(n^2)$
  - 而尴尬的是...
- ❖ 当所有元素都相等时，其实已经无需排序了
- ❖ 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)] ); T pivot = _elem[lo]; //随机轴点

   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)] ); T pivot = _elem[lo]; //随机轴点

   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

- 遇到连续的相等元素时
  - 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，且更不稳定