快速排序

大连理工大学

于 红

快速排序思想

- 选择轴值（pivot）
- 将序列划分为两个子序列L和R，使得L中所有记录都小于或等于轴值，R中记录都大于轴值
- 对子序列L和R递归进行快速排序

快速排序——轴值选择

- 尽可能使L，R长度相等

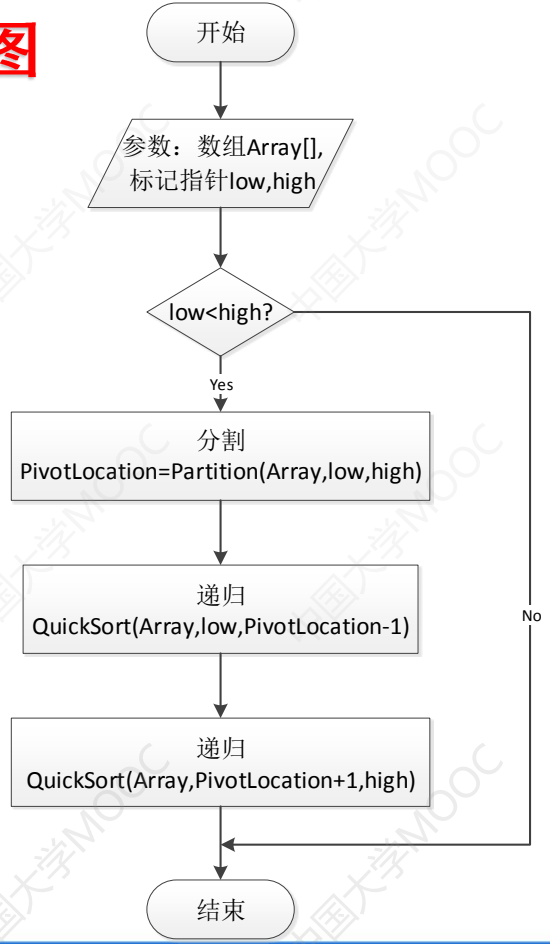
- 常用策略：
  - 选择最左边记录(第一个记录)
  - 随机选择
  - 选择中间值

快速排序——分割过程

- 分割后使得
  - L中所有记录小于轴值
  - R中记录大于轴值
  - 轴值位于正确位置

快速排序

快速排序算法流程图



快速排序——分割过程

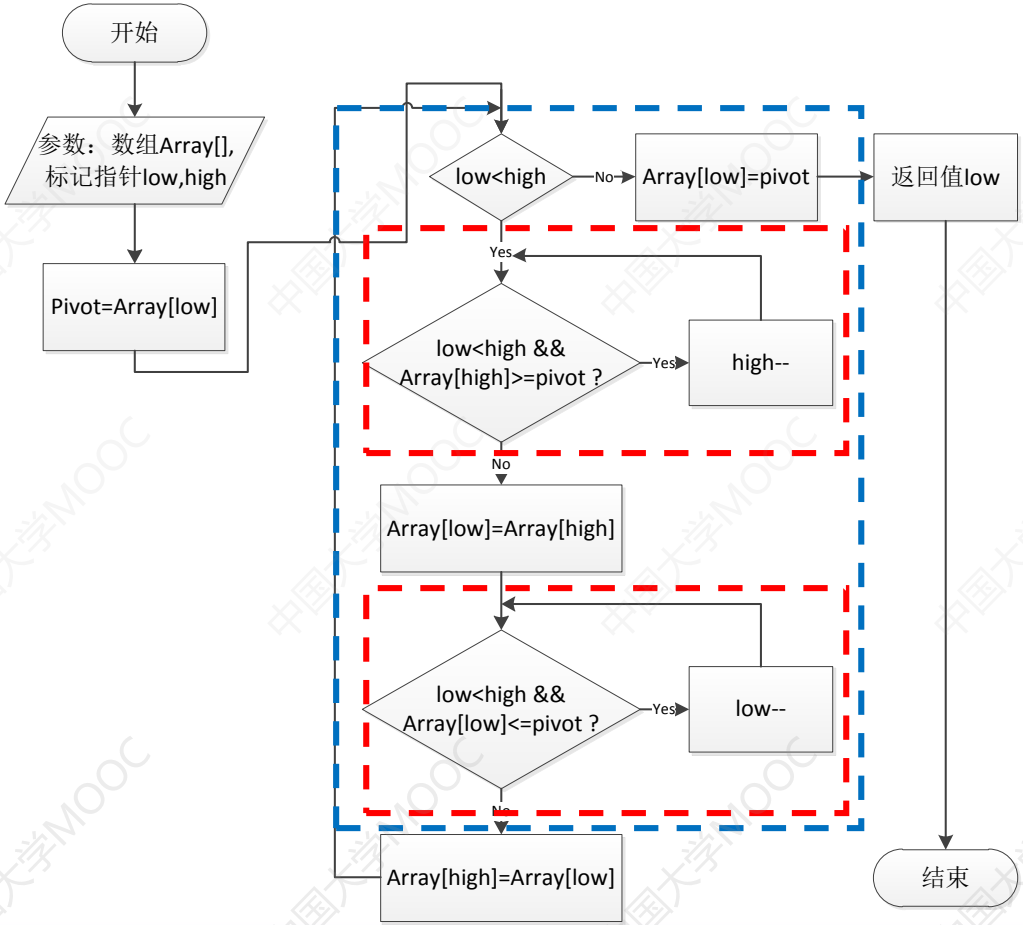
- 备份轴记录
- 取两个指针low和high，初始值就是序列的两端下标，保证low<=high
- 移动两个指针
  - 从high向左找到第一个小于轴的元素, 放在low的位置
  - 从low向右找到第一个大于轴的元素，放在high的位置
- 重复，直到low=high
- 把轴放在low所指的位置

快速排序——分割过程演示

0	1	2	3	4	5	6	7	
low							high	pivot = 49
49	38	65	97	76	13	27	49	
						high		
27	38	65	97	76	13		49	
		low						
27	38		97	76	13	65	49	
					high			
27	38	13	97	76		65	49	
			low					
27	38	13	49	76	97	65	49	
			high					

快速排序

分割算法流程图



快速排序——分割过程代码

```
int Partition (T Array[], int low, int high){
  T pivot = Array[low];
        while(low < high){
            while(low < high && Array[high] >= pivot)   high --;
              Array[low] = Array[high];
            while(low < high && Array[low] <= pivot)    low++;
             Array[high] = Array[low];
        }
   Array[low] = pivot;
   return low;
}
```

快速排序过程演示

第一趟排序后　13　38　27　49　76　97　65　52

49 将序列分成两部分，分别进行新的快速排序；

第二趟排序　13　38　27　76　97　65　52

第二趟排序后　13　38　27　65　52　76　97

第三趟排序　38　27　65　52

第三趟排序后　27　38　52　65

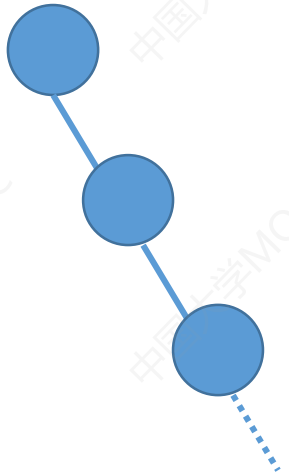
最终有序序列为：13　27　38　52　65　76　97

快速排序的效率分析——最好及平均

- 若快速排序出现最好的情形（左、右子区间的长度大致相等），则结点数n与二叉树深度h应满足$\log_2 n<h<\log_2(n+1)$，所以总的比较次数不会超过$(n+1)\log_2 n$。
- 快速排序的最好时间复杂度应为$O(n\log_2 n)$。
- 在理论上已经证明，快速排序的平均时间复杂度也为$O(n\log_2 n)$。

快速排序的效率分析——最坏情形

- 每次能划分成两个子区间，但其中一个是空，得到一棵单分枝树，得到的非空子区间包含有n-i个(i 代表二叉树的层数（1≤i≤n）元素。
- 每层划分需要比较n-i+2次，所以总的比较次数为（n^2+3n-4）/2。
- 快速排序的最坏时间复杂度为$O(n^2)$。



快速排序的效率分析——空间复杂度与稳定性

- 快速排序所占用的辅助空间为栈的深度，故最好的空间复杂度为$O(\log_2 n)$，最坏的空间复杂度为$O(n)$。
- 快速排序是一种不稳定的排序方法。

(举例：6, 7, 5, 2, 5, 8)

快速排序

大连理工大学

于 红