第2章　递归与分治策略(3)

快速排序

内容

2.1 求解递归式

2.2 递归

1. 阶乘函数
2. Fibonacci数列
3. 排列问题
4. 整数划分问题
5. Hanoi塔问题

2.3 分治

1. 二分搜索技术
2. 合并排序
3. **快速排序**
4. 线性时间选择

递归与分治策略总体思想

- 将一个难以直接解决的大问题，分割成一些规模较小的相同子问题。如果这些子问题都可解，并可利用这些子问题的解求出原问题的解，则这种分治法可行。

分治法的适用条件

- 分治法所能解决的问题一般具有以下特征：
  1. 该问题的规模缩小到一定的程度就可以容易地解决；
  2. 该问题可以分解为若干个规模较小的相同问题
  3. 利用该问题分解出的子问题的解可以合并为该问题的解；
  4. 该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子问题。

分治法的基本步骤

DIVIDE-AND-CONQUER(P)

//当问题规模小于n_0时，直接求解

if $|P| \leq n_0$, **ADHOC**(P), return

//分解问题

divide P into smaller subinstances $P_1, P_2, \ldots, P_a$

//递归的解各子问题

for i=1 to a

y_i=**DIVIDE-AND-CONQUER** (P_i)

//将各子问题的解合并为原问题的解

return **MERGE**($y_1, \ldots, y_a$)

2.3 分治

1. 二分搜索技术
2. 合并排序
3. 快速排序
4. 线性时间选择

3 快速排序

- 对于输入的一个数组A[p..r]，按以下三个步骤进行排序：

1. **分解**：以其中一个元素为基准元素将数组A划分成3段，并满足**A[p..q-1] ≤ A[q] ≤ A[q+1..r]**；
2. **递归求解**：通过递归调用快速排序算法分别对A[p..q-1]和A[q+1..r]进行排序
3. **合并**：由于对A[p..q-1]和A[q+1..r]的排序是就地进行的，所以在它们这两段都已排好序后，不需要执行任何计算，A[p..r]就已排好序

快速排序两种算法

- 根据选取基准元素的方法：
  1. **快速排序**
     - 以数组第一个或最后一个元素为划分基准
  2. **随机选择策略的快速排序**
     - 随机选出一个元素为划分基准

快速排序算法描述

```
1  QUICKSORT(A, p, r)
2    if p<r
3      q=PARTITION(A, p, r)   //A[p:q-1]≤A[q]≤A[q+1:r]
4      QUICKSORT(A, p, q-1)
5      QUICKSORT(A, q+1, r)
```

PARTITION函数

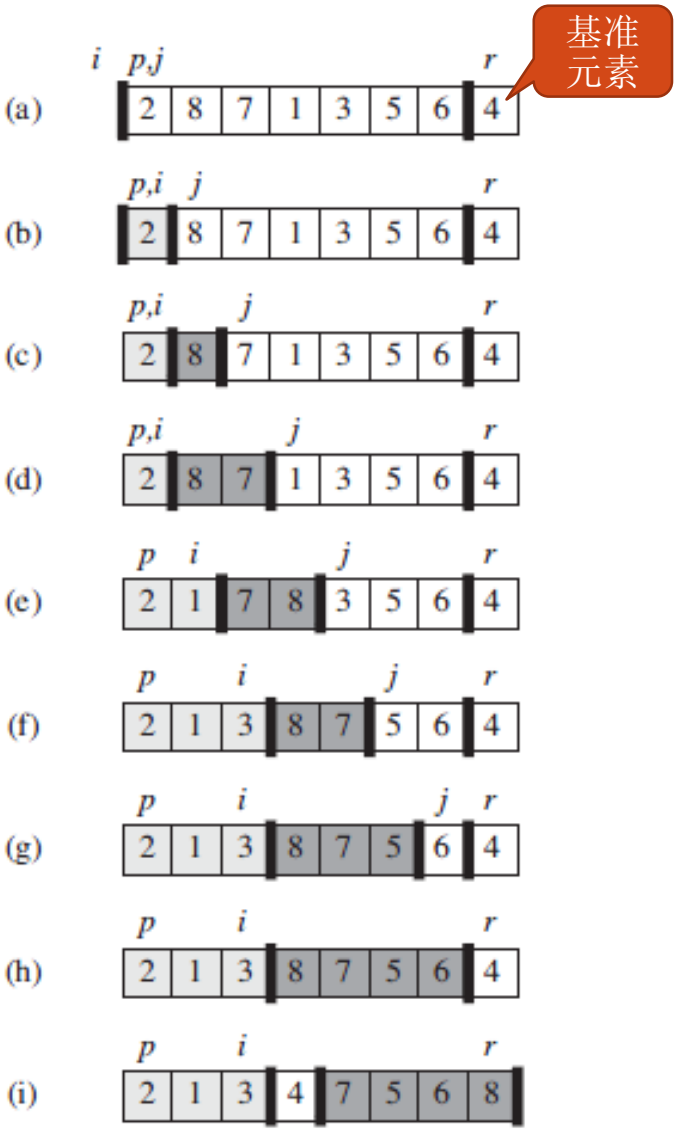
- 以A[r]为基准，将数组A[p..r] 划分成3段，并满足 $A[p..q-1] \le A[q] \le A[q+1..r]$
- 划分结束后，原A[r]就放在划分后q的位置

PARTITION函数算法(1)描述

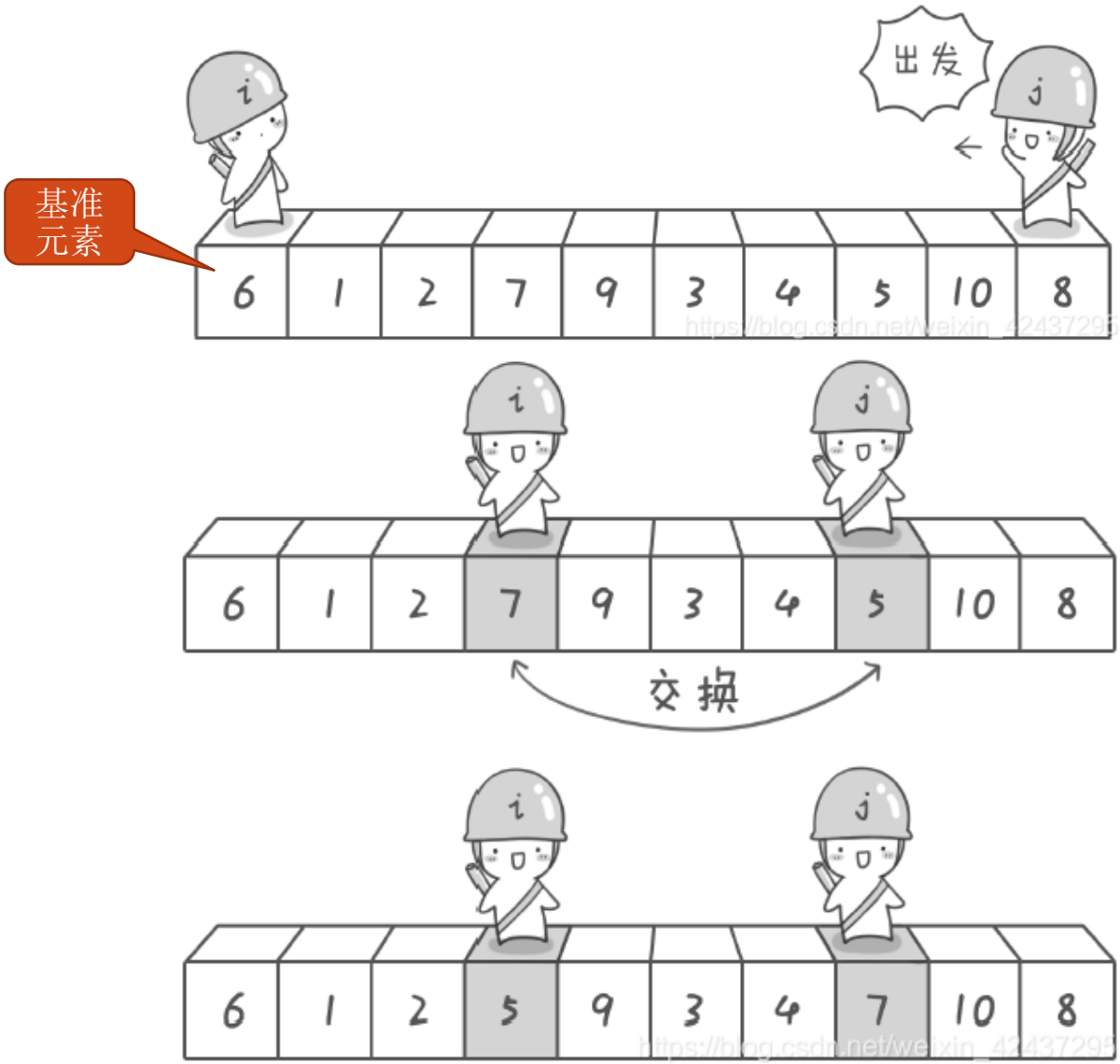
```
1  PARTITION(A, p, r)
2    x = A[r]
3    i = p-1
4    for j = p to r-1
5      if A[j] ≤ x
6        i = i+1
7        exchange A[i] with A[j]
8    exchange A[i+1] with A[r]
9    return i+1
```

for $p \le k \le i, A[k] \le A[r]$

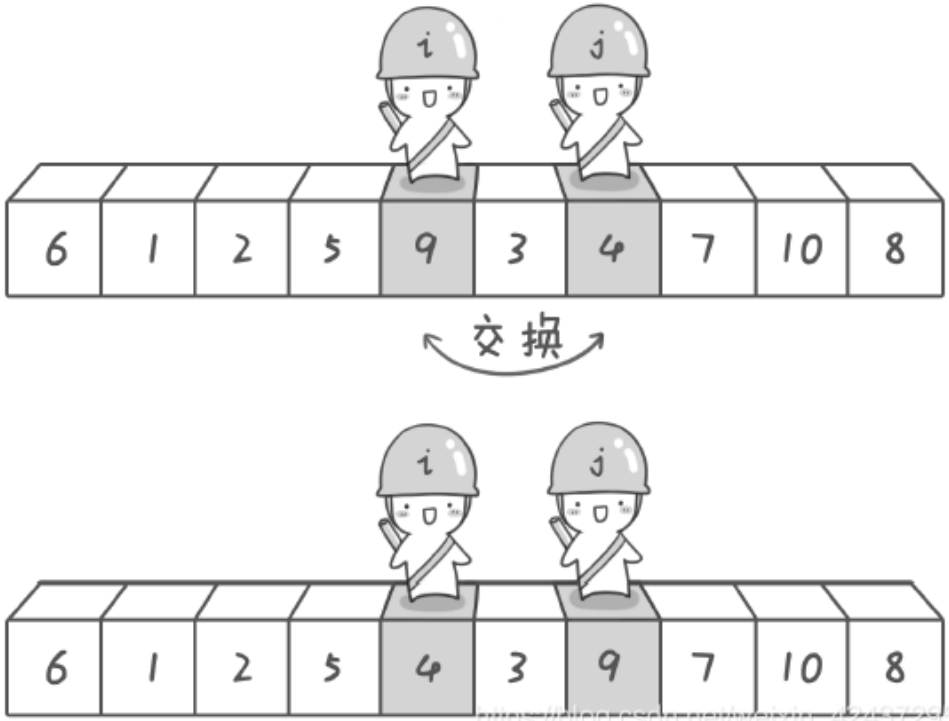
for $i+1 \le k < j, A[k] > A[r]$



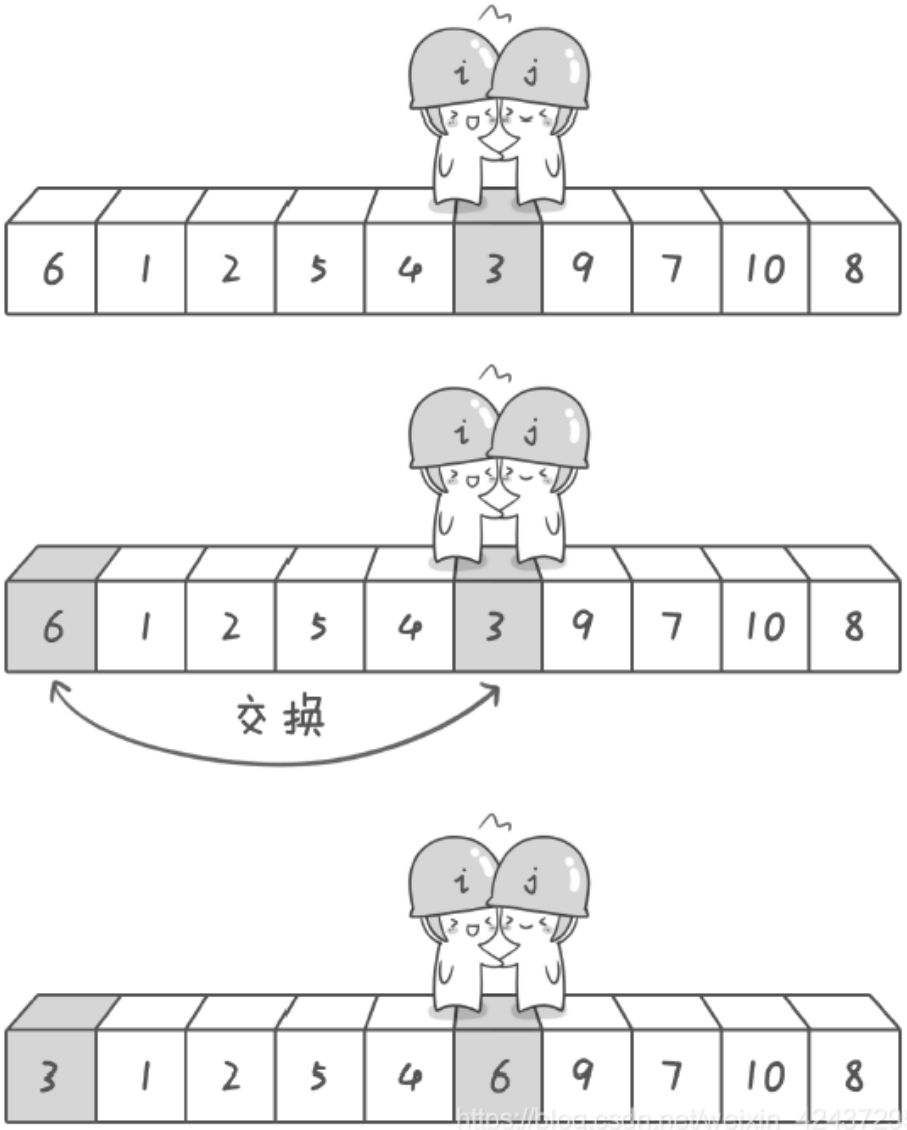
PARTITION函数算法(2)描述



PARTITION函数算法(2)描述



PARTITION函数算法(2)描述



快速排序算法分析–每个元素互不相同

- 最坏情况下：
  - 输入是正序或逆序
  - 其中一个划分没有元素

$$T(n) = T(n-1) + T(0) + \Theta(n)$$
$$= T(n-1) + \Theta(n)$$
$$T(n) = \Theta(n^2)$$

- 最好情况下：
  - 每次划分正好将n个元素一分为二

$$T(n) = 2T(n/2) + \Theta(n)$$
$$T(n) = \Theta(n \log n)$$

快速排序算法分析–每个元素互不相同

- 平衡划分情况下：
  - 每次划分正好是 $\frac{n}{10} : \frac{9n}{10}$

$$T(n) = T(9n/10) + T(n/10) + \Theta(n)$$

$$T(n) = \Theta(n \log n)$$

随机选择策略的快速排序

- 快速排序算法的性能取决于划分的对称性。
- 通过修改算法PARTITION，可以设计出采用随机选择策略的快速排序算法。
- 在快速排序算法的每一步中，当数组还没有被划分时，可以在A[p..r]中随机选出一个元素作为划分基准，这样可以使划分基准的选择是随机的，从而可以使期望划分是较对称的。

随机选择策略的快速排序 - 算法描述

```
1  RANDOMIZED_QUICKSORT(A, p, r)
2    if p < r
3      q=RANDOMIZED_PARTITION(A, p, r)
4      RANDOMIZED_QUICKSORT(A, p, q-1)
5      RANDOMIZED_QUICKSORT(A, q+1, r)
6
7  RANDOMIZED_PARTITION(A, p, r)
8    i = RANDOM(p, r)
9    exchange A[r] with A[i]
10   return PARTITION(A, p, r)
```

随机选择策略的快速排序 - 时间复杂性分析

- 最坏时间复杂度：$\mathrm{O}\left(n^2\right)$
- 平均时间复杂度：$\mathrm{O}\left(n \log n\right)$

快速排序总结

- Sort “in place”，无需额外的空间
- 非常实用的排序算法
- 随机选择策略的快速排序
  - 无需对输入序列的分布做任何假设
  - 没有一种特定输入会引起最差的运行效率
  - 最差的情况是由随机数产生器决定

End