# 10.4.2 快速排序

快速排序是一种所需**比较次数较少、速度较快**的排序方法，由英国计算机科学家，图灵奖获得者东尼·霍尔于1961年发表。在基于比较的排序算法中，快速排序的**平均性能突出**。

**基本思想**：通过递归分治方法，基于轴点（基准值pivot）将待排序序列拆分成两个子序列并分别排序，直到序列有序。

**排序步骤**：

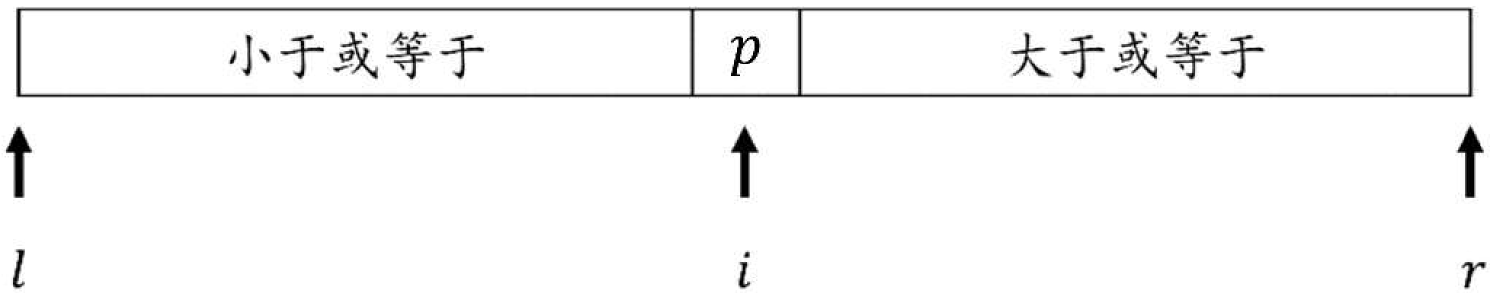
- 从待排序序列中**选取轴点**。
- 通过交换序列元素，将待排序序列**拆分**为左右两个子序列，左子序列元素**小于等于**轴点，右子序列元素**大于等于**轴点。
- 对两个子序列**递归**进行上述操作，直到子序列**元素个数小于等于1**。

## 快速排序的序列拆分（划分）

- 序列拆分（Partition）会根据所选**轴点**将序列拆分成两个子序列，进而确定轴点在序列的**最终位置**
- 对于序列$a_l, \ldots, a_r$和轴点元素$p$，我们需要交换序列中的元素，将轴点$p$置于正确的位置$a_i$，使得

$$\boldsymbol{a_l, \ldots, a_{i-1} \le a_i \le a_{i+1}, \ldots, a_r}$$

- 返回轴点所在下标$i$



## 快速排序的序列拆分

- 序列拆分（Partition）会根据所选**轴点**将序列拆分成两个子序列，进而确定轴点在序列的**最终位置**
- 对于序列$a_l, \dots, a_r$和轴点元素$p$，我们需要交换序列中的元素，将轴点$p$置于正确的位置$a_i$，使得

$$\boldsymbol{a_l, \dots, a_{i-1} \le a_i \le a_{i+1}, \dots, a_r}$$

- 返回轴点所在下标$i$

| | | |
|---|---|---|
| $a_l$ | … | $a_r$ |

Partition

| | | | |
|---|---|---|---|
| $a_l$　　$a_{i-1}$ | $a_i$ | $a_{i+1}$　　$a_r$ | |

Partition　　　　Partition

…　　　　…

思考：为何划分后，轴点左右两个子序列可以分开独立排序？

# 序列拆分示例

对序列$a = \{75, 34, 45, 32, 78, 12, 34', 40\}$进行一次拆分，其中$l = 0, r = 7$。

1. $i$指向待划分区域首元素，$j$指向待划分区域尾元素的前一个元素
2. 待划分区域最后的元素作为轴点

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34’** | **40** |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34’** | **40** |
| i | | | | | | j | 轴点 |

# 序列拆分示例

对序列$a = \{75, 34, 45, 32, 78, 12, 34', 40\}$进行一次拆分，其中$l = 0, r = 7$。

1. $i$指向待划分区域首元素，$j$指向待划分区域尾元素的前一个元素
2. 待划分区域最后的元素作为轴点
3. $i$从前往后移动直到找到一个比轴点大或相等的项
4. $j$从后往前移动直到找到一个比轴点小或相等的项

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34’** | **40** |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34’** | **40** |
| i | | | | | | j | 轴点 |

# 序列拆分示例

对序列$a = \{75, 34, 45, 32, 78, 12, 34', 40\}$进行一次拆分，其中$l = 0, r = 7$。

1. $i$指向待划分区域首元素，$j$指向待划分区域尾元素的前一个元素
2. 待划分区域最后的元素作为轴点
3. $i$从前往后移动直到找到一个比轴点大或相等的项
4. $j$从后往前移动直到找到一个比轴点小或相等的项
5. 交换 $a_0$ 与 $a_6$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34'** | **40** |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34'** | **40** |
| i | | | | | | j | 轴点 |

# 序列拆分示例

对序列$a = \{75, 34, 45, 32, 78, 12, 34', 40\}$进行一次拆分，其中$l = 0, r = 7$。

1. $i$指向待划分区域首元素，$j$指向待划分区域尾元素的前一个元素
2. 待划分区域最后的元素作为轴点
3. $i$从前往后移动直到找到一个比轴点大或相等的项
4. $j$从后往前移动直到找到一个比轴点小或相等的项
5. 交换 $a_0$ 与 $a_6$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34'** | **40** |
| **34'** | **34** | **45** | **32** | **78** | **12** | **75** | **40** |
| i | | | | | | j | 轴点 |

# 序列拆分示例

对序列$a = \{75, 34, 45, 32, 78, 12, 34', 40\}$进行一次拆分，其中$l = 0, r = 7$。

1. $i$指向待划分区域首元素，$j$指向待划分区域尾元素的前一个元素
2. 待划分区域最后的元素作为轴点
3. $i$从前往后移动直到找到一个比轴点大或相等的项
4. $j$从后往前移动直到找到一个比轴点小或相等的项
5. 交换 $a_0$ 与 $a_6$
6. goto 3
7. 再一次循环后：交换 $a_2$ 与 $a_5$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34’** | **40** |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **34’** | **34** | **45** | **32** | **78** | **12** | **75** | **40** |
| | | i | | | j | | 轴点 |

高等教育出版社

# 序列拆分示例

对序列$a = \{75, 34, 45, 32, 78, 12, 34', 40\}$进行一次拆分，其中$l = 0, r = 7$。

1. $i$指向待划分区域首元素，$j$指向待划分区域尾元素的前一个元素
2. 待划分区域最后的元素作为轴点
3. $i$从前往后移动直到找到一个比轴点大或相等的项
4. $j$从后往前移动直到找到一个比轴点小或相等的项
5. 交换 $a_0$ 与 $a_6$
6. goto 3
7. 再一次循环后：交换 $a_2$ 与 $a_5$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 75 | 34 | 45 | 32 | 78 | 12 | 34’ | 40 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 34’ | 34 | 12 | 32 | 78 | 45 | 75 | 40 |
| | | i | | | j | | 轴点 |

# 序列拆分示例

对序列$a = \{75, 34, 45, 32, 78, 12, 34', 40\}$进行一次拆分，其中$l = 0, r = 7$。

1. $i$指向待划分区域首元素，$j$指向待划分区域尾元素的前一个元素
2. 待划分区域最后的元素作为轴点
3. $i$从前往后移动直到找到一个比轴点大或相等的项
4. $j$从后往前移动直到找到一个比轴点小或相等的项
5. 交换 $a_0$ 与 $a_6$
6. goto 3
7. 再一次循环后：交换 $a_2$ 与 $a_5$
8. 再一次循环后：$i \geq j$循环结束

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34’** | **40** |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **34’** | **34** | **12** | **32** | **78** | **45** | **75** | **40** |
| | | | j | i | | | 轴点 |

# 序列拆分示例

对序列$a = \{75, 34, 45, 32, 78, 12, 34', 40\}$进行一次拆分，其中$l = 0, r = 7$。

1. $i$指向待划分区域首元素，$j$指向待划分区域尾元素的前一个元素
2. 待划分区域最后的元素作为轴点
3. $i$从前往后移动直到找到一个比轴点大或相等的项
4. $j$从后往前移动直到找到一个比轴点小或相等的项
5. 交换 $a_0$ 与 $a_6$
6. goto 3
7. 再一次循环后：交换 $a_2$ 与 $a_5$
8. 再一次循环后：$i \geq j$循环结束
9. 交换$a_i$与$a_r$，划分结束

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 75 | 34 | 45 | 32 | 78 | 12 | 34’ | 40 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 34’ | 34 | 12 | 32 | 78 | 45 | 75 | 40 |
| | | | j | i | | | 轴点 |

# 序列拆分示例

对序列$a = \{75, 34, 45, 32, 78, 12, 34', 40\}$进行一次拆分，其中$l = 0, r = 7$

1. $i$指向待划分区域首元素，$j$指向待划分区域尾元素的前一个元素
2. 待划分区域最后的元素作为轴点
3. $i$从前往后移动直到找到一个比轴点大或相等的项
4. $j$从后往前移动直到找到一个比轴点小或相等的项
5. 交换 $a_0$ 与 $a_6$
6. goto 3
7. 再一次循环后：交换 $a_2$ 与 $a_5$
8. 再一次循环后：$i \geq j$循环结束
9. 交换$a_i$与$a_r$，划分结束

**10. 返回4，即轴点最终位置！**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **75** | **34** | **45** | **32** | **78** | **12** | **34'** | **40** |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **34'** | **34** | **12** | **32** | **40** | **45** | **75** | **78** |
| | | | j | i | | | 轴点 |

# 序列拆分伪代码

**算法10-5 序列拆分 Partition$(a, l, r)$**

**输入：** 序列$a$，左端点下标$l$，右端点下标$r$

**输出：** 将序列a根据轴点拆分，并输出轴点在序列中的位置

$i \leftarrow l$
$j \leftarrow r-1$
$p \leftarrow a_r$ //选择序列最后一个元素作为轴点
while true do
| **while** $a_i < p$ **do** //找到$i$以右第一个大于等于轴点的元素
| | $i \leftarrow i+1$
| **end**
| **while** $a_j > p$ **and** $j > l$ **do** //找到$j$以前第一个小于等于轴点的元素
| | $j \leftarrow j-1$
| **end**
| **if** $i \geq j$ **then** //如果i大于等于j，完成拆分，退出循环
| | **break**
| **end**
| Swap$(a_i, a_j)$ //交换$a_i$和$a_j$并右移$i$左移$j$
| $i \leftarrow i+1$
| $j \leftarrow j-1$
**end**
Swap$(a_i, a_r)$
//此时$\{a_l, \ldots, a_{i-1}\} \leq a_i \leq \{a_{i+1}, \ldots, a_r\}$
**return** $i$

## 单选题 1分

对长度为n的序列进行划分，时间复杂度是

A O(n)

B O(nlog(n))

C $O(n\sqrt{n})$

D O(1)

多选题 1分

由1-10这十个数值按任意顺序构成的序列，5排在最后且作为轴点进行了划分，下面哪些描述是正确的?

A 1一定在序列的最前端，10一定在序列的最后

B 总的交换次数不会超过五次

C 划分结束时一定是 i > j

D 如果要让10排到序列最后，开始时10就必须在序列第五个位置

# 快速排序伪代码

**算法10-5 快速排序** QuickSort$(a, l, r)$

**输入：** 序列$a$，左端点下标$l$，右端点下标$r$

**输出：** 调整$a_l, a_{l+1}, \dots, a_r$元素顺序，使元素按照非递减顺序排列

```
if i < r then //超过1个元素才进行排序
|  i ← Partition(a, l, r)  //划分，返回轴点最终位置
|  QuickSort(a, l, i − 1)  //对轴点前段数据排序
|  QuickSort(a, i + 1, r)  //对轴点后段数据排序
end
```

高等教育出版社

# 快速排序示例

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 68 | 65 | 84 | 65 | 83 | 84 | 82 | 85 | 67 | 84 | 85 | 82 | 69 |
| 68 | 65 | 67 | 65 | 69 | 84 | 82 | 85 | 84 | 84 | 85 | 82 | 83 |
| 65 | 65 | 67 | 68 | | | | | | | | | |
| | | 67 | 68 | | | | | | | | | |
| | | | | | 82 | 82 | 83 | 84 | 84 | 85 | 84 | 85 |
| | | | | | 82 | 82 | | | | | | |
| | | | | | | | | 84 | 84 | 84 | 85 | 85 |
| | | | | | | | | 84 | 84 | 84 | | |
| 65 | 65 | 67 | 68 | 69 | 82 | 82 | 83 | 84 | 84 | 84 | 85 | 85 |

被选为轴点

序列拆分后轴点所在位置

# 快速排序性能分析

## 最好情况分析：

假设每次对序列的划分结果，两个子序列的长度比为1:1。这种情况下，设快速排序算法的时间为T(n), 则T(n)可表示成以下的递推方程：

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n),\ \mathrm{T}(1) = \mathrm{O}(1)$$

O(n)是划分的时间

$$\begin{aligned}
\mathrm{T}(n) &= 2\,\mathrm{T}(\frac{n}{2}) + c_1 n \\
&= 4\,\mathrm{T}(\frac{n}{4}) + c_1 \frac{n}{2} \times 2 + c_1 n \\
&= 8\,\mathrm{T}(\frac{n}{8}) + c_1 \frac{n}{4} \times 4 + c_1 \frac{n}{2} \times 2 + c_1 n \\
&= \ldots \\
&= 2^k\,\mathrm{T}(\frac{n}{2^k}) + c_1\left(\frac{n}{2^{k-1}} \times 2^{k-1} + \ldots + \frac{n}{2} \times 2 + n\right)
\end{aligned}$$

$$\text{if } \frac{n}{2^k} = 1 \text{ then}$$

$$\mathrm{T}(n) = c_2 2^k + c_1 \underbrace{\left(\frac{n}{2^{k-1}} \times 2^{k-1} + \ldots + \frac{n}{2} \times 2 + n\right)}_{\log(n)\text{个}n}$$

$$T(n) = O(\mathrm{n}\log(n))$$

# 快速排序性能分析

**最好情况分析：**

假设每次对序列的划分结果，两个子序列的长度比为1:1。这种情况下，设快速排序算法的时间为T(n), 则T(n)可表示成以下的递推方程：

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n),\ \mathrm{T}(1) = \mathrm{O}(1)$$

O(n)是划分的时间

**最好情况出现的条件：**

- 每次都能选出序列的中位数作为轴点
- 序列中所有元素相同（？）

高等教育出版社

# 快速排序性能分析

## 最坏情况分析：

每次划分，都选择最大值或最小值作为轴点，划分的结果是其中一个子序列为空序列，而另一个子序列包含除轴点外的所有元素。因此，时间复杂度为

$$T(n) = T(n-1) + O(n),\ T(1) = O(1)$$

$$\begin{aligned}
\mathrm{T}(n) &= \mathrm{T}(n-1) + c_1 n \\
&= \left(\mathrm{T}(n-2) + c_1(n-1)\right) + c_1 n \\
&= \mathrm{T}(n-2) + c_1 n + c_1(n-1) \\
&= \mathrm{T}(n-3) + c_1 n + c_1(n-1) + c_1(n-2) \\
&\ \vdots \\
&= \mathrm{T}(1) + c_1 \sum_{i=2}^{n} i = c_2 + c_1 \sum_{i=2}^{n} i \\
&= c_1 \sum_{i=1}^{n} i + (c_2 - c_1) = c_1 \frac{n(n+1)}{2} + (c_2 - c_1)
\end{aligned}$$

$$\Rightarrow \quad T(n) = O(n^2)$$

# 快速排序性能分析

**平均情况分析：**

- 假设 $a_1 < \cdots < a_k < \cdots < a_n$，且$\text{perm}(a_1, \dots, a_n)$表示$n$个元素的某个序列。因此，共有$n!$种不同的序列，而其中 $a_k$ $(1 \leq k \leq n)$在末尾的序列有$(n-1)!$，即$a_k$出现在序列末尾的概率为 $\frac{1}{n}$。

- 考虑$a_k$出现在序列末尾，作为轴点划分的结果如下：

$$\text{perm}(a_1, \dots a_{k-1}),\ a_k, \text{perm}(a_{k+1}, \dots, a_n)$$

单选题 1分

从n个互不相同的数值中，随机选择一个值作为轴点，将所有数据分成小于该轴点和大于该轴点的两组，两组长度比例的平均值（期待值）是:

A 1:3

B 1:2

C 1:1

D 2:1

E 3:1

# 快速排序性能分析

**平均情况分析：**

- 假设 $a_1 < \cdots < a_k < \cdots < a_n$，且$\text{perm}(a_1, \ldots, a_n)$表示$n$个元素的某个序列。因此，共有$n!$种不同的序列，而其中 $a_k$ $(1 \le k \le n)$在末尾的序列有$(n-1)!$，即$a_k$出现在序列末尾的概率为 $\frac{1}{n}$。
- 考虑$a_k$出现在序列末尾，作为基准值划分的结果如下：

$$\text{perm}(a_1, \ldots a_{k-1}),\ a_k, \text{perm}(a_{k+1}, \ldots, a_n)$$

- 划分的平衡性：元素数量较少的子序列 ⟺ 元素数量较多的子序列

- $a_k$ 划分出的元素数量少的子序列长度 $= \begin{cases} k-1, & \text{if } k \le \frac{n}{2} \\ n-k, & \text{if } k > \frac{n}{2} \end{cases}$

- 平均长度 $\frac{1}{n}\left(\sum_{1 \le k \le \frac{n}{2}} (k-1) + \sum_{\frac{n}{2} < k \le n} (n-k)\right) = \frac{1}{n}\sum_{1 \le k \le \frac{n}{2}} \left[k-1+n-\left(\frac{n}{2}+k\right)\right] = \frac{1}{n}\sum_{1 \le k \le \frac{n}{2}} \left(\frac{n}{2}-1\right) \approx \frac{n}{4}\ (n \to \infty)$

# 快速排序性能分析

## 平均情况分析：

- 假设 $a_1 < \cdots < a_k < \cdots < a_n$，且$\mathrm{perm}(a_1, \ldots, a_n)$表示$n$个元素的某个序列。因此，共有$n!$种不同的序列，而其中 $a_k$ $(1 \le k \le n)$在末尾的序列有$(n-1)!$，即$a_k$出现在序列末尾的概率为 $\frac{1}{n}$。
- 考虑$a_k$出现在序列末尾，作为基准值划分的结果如下：

$$\mathrm{perm}(a_1, \ldots a_{k-1}),\ a_k,\ \mathrm{perm}(a_{k+1}, \ldots, a_n)$$

## 轴点选择的重要性质：

- 随机选择轴点，则划分出的 数量较少组 与 数量较多组 的平均长度比为 1 : 3

Q: 长度比不低于 1:9 的概率是多少?

A: 如果选 $a_k$ $(k \in [\frac{n}{10}, \frac{9n}{10}])$为基准值，则长度比大于等于 1 : 9

随机选择（或选择末尾）元素作为轴点，80%以上的几率划分出的两组长度比好于 1 : 9

# 快速排序性能分析

**平均情况分析：**

- 假设 $a_1 < \cdots < a_k < \cdots < a_n$，且$\text{perm}(a_1, \ldots, a_n)$表示$n$个元素的某个序列。因此，共有$n!$种不同的序列，而其中 $a_k$ $(1 \le k \le n)$在末尾的序列有$(n-1)!$，即$a_k$出现在序列末尾的概率为 $\frac{1}{n}$。
- 考虑$a_k$出现在序列末尾，作为基准值划分的结果如下：

$$\text{perm}(a_1, \ldots a_{k-1}),\ a_k, \text{perm}(a_{k+1}, \ldots, a_n)$$

**轴点选择的重要性质：**

- 随机选择轴点，则划分出的 数量少组 与 数量多组 的平均长度比为 1 : 3
- 随机选择轴点，80%以上的几率划分出的两组长度比好于 1 : 9
- 同理可证，90%以上的几率划分出的两组长度比不低于1 : 19

# 快速排序性能分析

**平均情况分析：**

- 随机选择轴点，则划分出的 数量少组 与 数量多组 的平均长度比为 1 : 3
- 随机选择轴点，80%以上的几率划分出的两组长度比好于 1 : 9
- 同理可证，90%以上的几率划分出的两组长度比不低于1 : 19

假设在排序过程中，每次划分出的两组比例不低于 $1:\gamma\ (\gamma \geq 1)$

时间复杂度：$T(n) = T\left(\frac{1}{1+\gamma}n\right) + T\left(\frac{\gamma}{1+\gamma}n\right) + O(n),\ T(1) = O(1)$

$$T(n) = O(\mathrm{n}\log(n))$$

--证明可见《算法导论》等

注：每次划分的比例越趋近于1 : 1 $(\gamma \to 1)$， $\mathrm{n}\log(n)$)前面的常数越小，因此实际的排序效率越高！

# 快速排序性能分析

**平均情况分析：** 基于组合数学的直接证明

- 元素互异且长度为n的序列共有$n!$种排列顺序，设$C(n)$表示用快速排序对所有可能排列进行排序所需的总比较次数。则所有排列的平均比较次数为$\frac{C(n)}{n!}$。
- 对于$n!$种可能排列进行划分，需要进行$n\times n!$次比较。
- 在快速排序中，如果轴点是第$k$大的值，则将序列分成长度为$n-k$和$k-1$两个子序列，将这两个子序列排序分别需要$C(n-k)$和$C(k-1)$次比较，划分后的总比较次数分别为：

$$C(n-k)\frac{(n-1)!}{(n-k)!} \quad \text{和} \quad C(k-1)\frac{(n-1)!}{(k-1)!}$$

- 于是有

$$\frac{C(n)}{n!}=\frac{n\times n!+\sum_{k=1}^{n}\left[C(n-k)\frac{(n-1)!}{(n-k)!}+C(k-1)\frac{(n-1)!}{(k-1)!}\right]}{n!}<2n\log n$$

故快速排序的平均复杂度为$\boldsymbol{O(n\log n)}$。

# 快速排序性能分析

**时间复杂度**:

- 最好时间复杂度：$O(n\log n)$
- 最坏时间复杂度：$O(n^2)$
- 平均时间复杂度：$O(n\log n)$

**空间复杂度**（递归栈深度）：

- 最好空间复杂度：$O(\log n)$
- 最坏空间复杂度：$O(n)$
- 平均空间复杂度：$O(\log n)$

快速排序是**不稳定排序**。

## 快速排序的应用

(1) 百度面试题：假设一整型数组存在若干正数和负数，现在通过某种算法使得该数组的所有负数在正数的左边，时间复杂度O(n)

算法：选0作基准值进行划分！

(2) 长度为n的正整数数组，分成两个不相交的子数组并分别计算其中的元素和。求两个子数组长度差最小，且和相差最大的分法。

分法1：从小到大快速排序，前$\left\lfloor\frac{n}{2}\right\rfloor$个元素为一组，其它一组

*时间复杂度$O(n\log(n))$

分法2：找到排位$\left\lfloor\frac{n}{2}\right\rfloor$的数值，作为轴点进行一次划分即可！？

快速查找算法：QuickSearch

# 快速排序的改进：三数取中

- 快速排序具有优秀的平均性能，但是最坏情况下性能会退化成和冒泡排序相当。
- 快速排序的时间效率受序列拆分（划分）的平衡性影响大，划分越平衡时间效率越高
- 实现最好的时间效率需要每次划分时，先快速（线性时间内？）查找序列中的中位数，难度大（不比排序问题简单！！！）

- 一种直观的改进：不从整个序列，而是从一部分元素（子序列）中找中位数，然后作为轴点进行划分

例如：随机选择三个数，选其中的中位数作为轴点，即三数取中法

## 快速排序的改进：三数取中

三数取中方法对轴点选择进行改进：

- 选取序列中$a_l, a_{\frac{l+r}{2}}, a_r$三个元素的**中位数**作为轴点
- 或者从序列内**随机**选择三个元素，并选择三个元素的**中位数**作为轴点

需要指出的是，上述轴点选择策略都只能降低快速排序退化的概率，而**不能完全避免**快速排序退化。

# 快速排序的改进：三数取中

## 三数取中法的简单实现

```
median(A, a, b, c)      //序列A, 三数位置a,b,c
if A[a] < A[b]  then
|  if A[b] < A[c]  then
|  |  return b    //中位数在位置b
|  else if A[a] < A[c]  then
|  |     |  return c   //中位数在位置c
|  |     else
|  |     |  return a  //中位数在位置a
|  |     end
|  end
else if A[c] < A[b]  then
|      |  return b
|      else if A[a] < A[c] then
|      |     |  return c
|      |     else
|      |     |  return a
|      |     end
|      end
end
```

# 快速排序的改进：三数取中

```
median(A, a, b, c)      //序列A, 三数位置a,b,c
if A[a] < A[b]   then
|  if A[b] < A[c]   then
|  |  return b     //中位数在位置b
|  else if A[a] < A[c]  then
|  |     |  return c   //中位数在位置c
|  |     else
|  |     |  return a  //中位数在位置a
|  |     end
|  end
else if A[c] < A[b]  then
|     |  return b
|     else if A[a] < A[c] then
|     |     |  return c
|     |     else
|     |     |  return a
|     |     end
|     end
end
```

设 $a_1 < \cdots < a_k < \cdots < a_n$

- 随机挑选元素，每个数值被选的概率相同
- 按三数选中法，先随机选出三个元素，再从中选出中位数
- 问$a_k$被选作轴点(pivot)的概率是多少？

思考： $a_k$成为轴点的条件是什么？

# 快速排序的改进：三数取中

```
median(A, a, b, c)      //序列A, 三数位置a,b,c
if A[a] < A[b]  then
|  if A[b] < A[c]  then
|  |  return b    //中位数在位置b
|  else if A[a] < A[c]  then
|  |    |  return c   //中位数在位置c
|  |    else
|  |    |  return a  //中位数在位置a
|  |    end
|  end
else if A[c] < A[b]  then
|     |  return b
|     else if A[a] < A[c] then
|     |     |  return c
|     |     else
|     |     |  return a
|     |     end
|     end
end
```

设 $a_1 < \cdots < a_k < \cdots < a_n$

- 随机挑选元素，每个数值被选的概率相同
- 按三数选中法，先随机选出三个元素，再从中选出中位数
- 问$a_k$被选作轴点(pivot)的概率是多少?

$a_k$成为轴点的条件：

（1）选了$a_k$

（2）选了比$a_k$小的元素

（3）选了比$a_k$大的元素

## 单选题 1分

设 $a_1 < \cdots < a_k < \cdots < a_n$，从中随机选一个元素，该元素 $< a_k$ 的概率是：

A $\frac{1}{n}$

B $\frac{k-1}{n}$

C $\frac{k}{n}$

D $\frac{n-k}{n}$

# 快速排序的改进：三数取中

```
median(A, a, b, c)      //序列A, 三数位置a,b,c
if A[a] < A[b]   then
|  if A[b] < A[c]   then
|  |  return b     //中位数在位置b
|  else if A[a] < A[c]  then
|  |     |  return c   //中位数在位置c
|  |     else
|  |     |  return a  //中位数在位置a
|  |     end
|  end
else if A[c] < A[b]  then
|     |  return b
|     else if A[a] < A[c] then
|     |     |  return c
|     |     else
|     |     |  return a
|     |     end
|     end
end
```

设 $a_1 < \cdots < a_k < \cdots < a_n$

- 随机挑选元素，每个数值被选的概率相同
- 按三数选中法，先随机选出三个元素，再从中选出中位数
- 问$a_k$被选作轴点(pivot)的概率是多少?

$a_k$成为轴点的条件：

(1) 选了$a_k$ -------- $p_1 = \dfrac{1}{n}$

(2) 选了比$a_k$小的元素 ------------ $p_2 = \dfrac{k-1}{n-1}$

(3) 选了比$a_k$大的元素 ------ $p_3 = \dfrac{n-k}{n-2}$

(1), (2) 和 (3) 的顺序可以任意交换!

$$P(a_k \rightarrow pivot) = \frac{3!\,(k-1)(n-k)}{n(n-1)(n-2)}$$

高等教育出版社

# 快速排序的改进：三数取中

```
median(A, a, b, c)      //序列A, 三数位置a,b,c
if A[a] < A[b]   then
|  if A[b] < A[c]   then
|  |  return b     //中位数在位置b
|  else if A[a] < A[c]  then
|  |     |  return c   //中位数在位置c
|  |     else
|  |     |  return a  //中位数在位置a
|  |     end
|  end
else if A[c] < A[b]  then
|     |  return b
|     else if A[a] < A[c] then
|     |     |  return c
|     |     else
|     |     |  return a
|     |     end
|     end
end
```

设 $a_1 < \cdots < a_k < \cdots < a_n$

- 随机挑选元素，每个数值被选的概率相同，即 $\frac{1}{n}$
- 按三数选中法，先随机选出三个元素，再从中选出中位数
- 问$a_k$被选作轴点(pivot)的概率是多少?

$$P(a_k \rightarrow pivot) = \frac{3!\,(k-1)(n-k)}{n(n-1)(n-2)}$$

思考：$\{a_1, \dots, a_n\}$ 中，哪个元素被选为轴点的概率最大? 即

$$\max_{1 \le k \le n} \left\{ \frac{3!\,(k-1)(n-k)}{n(n-1)(n-2)} \right\}$$

# 快速排序的改进：三数取中

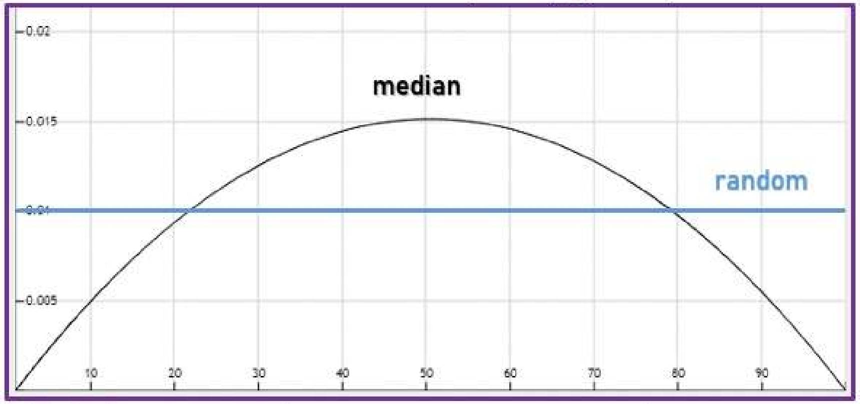
```
median(A, a, b, c)      //序列A, 三数位置a,b,c
if A[a] < A[b]   then
|  if A[b] < A[c]   then
|  |  return b     //中位数在位置b
|  else if A[a] < A[c]  then
|  |     |  return c   //中位数在位置c
|  |     else
|  |     |  return a  //中位数在位置a
|  |     end
|  end
else if A[c] < A[b]  then
|     |  return b
|     else if A[a] < A[c] then
|     |     |  return c
|     |     else
|     |     |  return a
|     |     end
|     end
end
```

设 $a_1 < \cdots < a_k < \cdots < a_n$

- 随机挑选元素，每个数值被选的概率相同，即 $\frac{1}{n}$
- 按三数选中法，先随机选出三个元素，再从中选出中位数
- 问$a_k$被选作轴点(pivot)的概率是多少?

$$P(a_k \rightarrow pivot) = \frac{3!\,(k-1)(n-k)}{n(n-1)(n-2)}$$

概率 P




$k \in [1, 100] \quad (n = 100)$

# 三数取中法对快速排序的作用

**随机选择轴点的性能：**

- 随机选择轴点，划分出的 数量较少组 与 数量较多组 的平均长度比为 1 : 3
- 80%以上的几率划分出的两组长度比好于 1 : 9
- 90%以上的几率划分出的两组长度比不低于1 : 19

**三数取中法选择轴点的性能：**

- 三数选中法选择轴点，划分出的 数量较少组 与 数量较多组 的平均长度比为 5 : 11 (≈ 1 : 2)
- 90%以上的几率划分出的两组长度比不低于 1 : 6.388

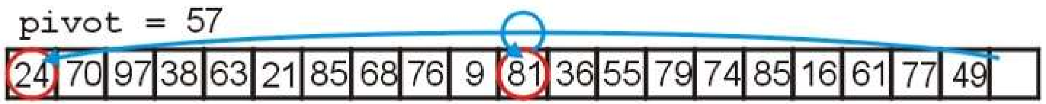
# 快速排序的示例

pivot =

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 57 | 70 | 97 | 38 | 63 | 21 | 85 | 68 | 76 | 9 | 81 | 36 | 55 | 79 | 74 | 85 | 16 | 61 | 77 | 49 | 24 |

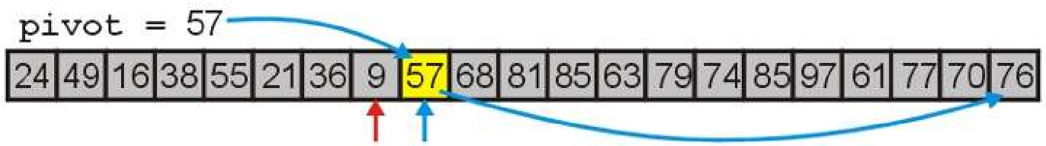
- 三数取中法，选择57作为轴点

# 快速排序的示例




- 将57与24交换，轴点移到序列末尾

# 快速排序的示例




- 划分结果

# 快速排序的示例

pivot =

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 24 | 49 | 16 | 38 | 55 | 21 | 36 | 9 | 57 | 68 | 81 | 85 | 63 | 79 | 74 | 85 | 97 | 61 | 77 | 70 | 76 |

- 对57前面的子序列快速排序
- 三数取中选择轴点

# 快速排序的示例

pivot = 24

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 49 | 16 | 38 | 55 | 21 | 36 | | 57 | 68 | 81 | 85 | 63 | 79 | 74 | 85 | 97 | 61 | 77 | 70 | 76 |

- 对57前面的子序列快速排序
- 三数取中选择轴点
- 将轴点移到子序列末尾，开始排序

# 快速排序的示例

pivot =

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 16 | 21 | 24 | 36 | 38 | 49 | 55 | 57 | 68 | 81 | 85 | 63 | 79 | 74 | 85 | 97 | 61 | 77 | 70 | 76 |

- 对57前面的子序列快速排序
- 三数取中选择轴点
- 将轴点移到子序列末尾，开始排序
- 前段子序列排序结束
- 对57后面的子序列排序，三数取中选轴点

# 快速排序的示例

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 16 | 21 | 24 | 36 | 38 | 49 | 55 | 57 | 61 | 63 | 68 | 70 | 74 | 76 | 77 | 79 | 81 | 85 | 85 | 97 |

- 对57前面的子序列快速排序
- 三数取中选择轴点
- 将轴点移到子序列末尾，开始排序
- 前段子序列排序结束
- 对57后面的子序列排序，三数取中选轴点
- 整个序列排序结束!