单边递归（Single-Side Recursion） 是快速排序（或类似分治算法）的一种优化技术，通过 **减少递归调用次数** 来降低栈空间开销。其核心思想是：**每次递归只处理一个子区间，另一个子区间通过循环迭代处理。**

---

1. 单边递归 vs 传统双边递归

(1) 传统双边递归（普通快速排序）

cpp 复制 下载

```cpp
void quick_sort(int arr[], int l, int r) {
    if (l >= r) return;
    int p = partition(arr, l, r); // 分区
    quick_sort(arr, l, p - 1);    // 递归处理左子数组
    quick_sort(arr, p + 1, r);    // 递归处理右子数组
}
```

问题：递归深度可能达到 **O(n)**（如数组已排序时），导致栈溢出。

(2) 单边递归优化

cpp 复制 下载

```cpp
void quick_sort_optimized(int arr[], int l, int r) {
    while (l < r) {               // 改用循环
        int p = partition(arr, l, r);
        quick_sort_optimized(arr, l, p - 1); // 仅递归处理左子数组
        l = p + 1;                // 右子数组通过循环迭代处理
    }
}
```

优势：递归深度降至 **O(log n)**，栈空间复杂度从 O(n) 优化为 O(log n)。

---

2. 单边递归的工作原理

步骤：

对当前区间进行分区，得到分界点 `p`。

递归处理较短的一个子数组（左或右）。

通过循环迭代处理较长的子数组（更新边界 `l` 或 `r`）。

关键点：优先递归较小的子区间，确保递归深度最小化。

示例：

cpp 复制 下载

```cpp
void quick_sort_optimized(int arr[], int l, int r) {
    while (l < r) {
        int p = partition(arr, l, r);
        if (p - l < r - p) {      // 左子数组更短
            quick_sort_optimized(arr, l, p - 1); // 递归处理短的左子数组
            l = p + 1;            // 迭代处理长的右子数组
        } else {                  // 右子数组更短
            quick_sort_optimized(arr, p + 1, r); // 递归处理短的右子数组
            r = p - 1;            // 迭代处理长的左子数组
        }
    }
}
```

---

3. 为什么单边递归能减少栈空间?

数学原理：每次递归调用处理 **较短的一半**，递归深度最多为 $\log_2 n$。

类比二叉树：每次优先遍历较短的子树，树高最小化。

极端案例：

传统递归：已排序数组的递归深度为 `n`。

单边递归：递归深度仅为 `log n`。

---

4. 单边递归的适用场景

深度递归风险高：如排序大规模数据时避免栈溢出。

尾调用优化（TCO）不可用：某些语言（如C++）默认不优化尾递归，需手动改为单边递归。

性能敏感场景：减少函数调用开销。

5. 与其他优化的结合

结合插入排序：对小区间（如 `r - l < 16`）改用插入排序。

随机化基准值：避免最坏时间复杂度。

cpp 复制 下载

```cpp
void quick_sort_final(int arr[], int l, int r) {
    while (r - l > 16) {          // 小区间优化
        int p = random_partition(arr, l, r);
        if (p - l < r - p) {
            quick_sort_final(arr, l, p - 1);
            l = p + 1;
        } else {
            quick_sort_final(arr, p + 1, r);
            r = p - 1;
        }
    }
    insertion_sort(arr + l, r - l + 1); // 插入排序处理小区间
}
```

总结

单边递归通过 **减少递归调用次数** 和 **优先处理较短子区间**，将快速排序的栈空间复杂度从 **O(n)** 优化到 **O(log n)**，是分治算法中经典的优化手段。其核心思想是：**用循环替代一半的递归**，适合处理大规模数据或递归深度受限的场景。