快速排序及优化

快速排序及优化

- 随堂实现
- 习题部分
  - 快速排序基础
  - 快速排序扩展
  - 温故知新
  - 智力发散
  - 自学推荐

随堂实现

```cpp
#include <bits/stdc++.h>

using namespace std;

/*
 * 最简单方法实现快速排序
 */
void quick_sort_v1(int *arr, int l, int r) {
    if (l >= r) { return; }

    int x = l, y = r, base = arr[l];
    while (x < y) {
        while (x < y && arr[y] >= base) {
            y--;
        }
        if (x < y) { arr[x++] = arr[y]; }
        while (x < y && arr[x] <= base) {
            x++;
        }
        if (x < y) { arr[y--] = arr[x]; }
    }
    arr[x] = base;
    quick_sort_v1(arr, l, x - 1);
    quick_sort_v1(arr, x + 1, r);

    return;
}


/*
 * 使用了单递归法实现的快速排序, 同时保证了无监督
 */
void quick_sort_v2(int *arr, int l, int r) {
    while (l < r) {
        int x = l, y = r, base = arr[l];
        while (x < y) {
            while (x < y && arr[y] >= base) {
                y--;
            }
            if (x < y) { arr[x++] = arr[y]; }
            while (x < y && arr[x] <= base) {
                x++;
            }
            if (x < y) { arr[y--] = arr[x]; }
        }
        arr[x] = base;
        quick_sort_v2(arr, x + 1, r);
        r = x - 1;
    }

    return;
}


/*
 * 模拟STL中实现的快速排序，实现了用插入排序优化快速排序
 */
const int threshold = 16;

inline int median(int a, int b, int c) {
    if (a > b) { swap(a, b); }
    if (a > c) { swap(a, c); }
    if (b > c) { swap(b, c); }
    return b;
}

void __quick_sort_v3(int *arr, int l, int r) {
    while (r - l > threshold) {
        int x = l, y = r, base = median(arr[l], arr[(l + r) / 2], arr[r]);
        do {
            while (arr[x] < base) { x++; }
            while (arr[y] > base) { y--; }
            if (x <= y) {
                swap(arr[x], arr[y]);
                x++, y--;
            }
        } while (x <= y);
        __quick_sort_v3(arr, x, r);
        r = y;
    }

    return;
}

void final_insert_sort(int *arr, int l, int r) {
    int ind = l;
    for (int i = l + 1; i <= r; i++) {
        if (arr[i] < arr[ind]) { ind = i; }
    }
    while (ind > l) {
        swap(arr[ind], arr[ind - 1]);
        --ind;
    }
    for (int i = l + 2; i <= r; i++) {
        int j = i;
        while (arr[j] < arr[j - 1]) {
            swap(arr[j], arr[j - 1]);
```

```
                j--;
            }
        }

        return;
}

void quick_sort_v3(int *arr, int l, int r) {
    __quick_sort_v3(arr, l, r);
    final_insert_sort(arr, l, r);

    return;
}


int main(int argc, char *argv[]) {
    int arr[10] = {3, 1, 4, 5, 9, 10};
    quick_sort_v1(arr, 0, 5);

    for (int i = 0; i < 6; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

习题部分

快速排序基础

- 148. 排序链表

```
class Solution {
public:
    ListNode *sortList(ListNode *head) {
        if (head == NULL) { return head; }
        int l = head->val, r = head->val;
        double mid;
        ListNode *p = head, *q, *h1 = NULL, *h2 = NULL;
        while (p) { l = min(p->val, l), r = max(p->val, r), p = p->next; }

        if (l == r) { return head; }
        mid = (l + r) / 2.0;
        p = head;
        while (p) {
            q = p->next;
            if (p->val <= mid) {
                p->next = h1;
                h1 = p;
            } else {
                p->next = h2;
                h2 = p;
            }
            p = q;
        }
        h1 = sortList(h1);
        h2 = sortList(h2);
        p = h1;
        while (p->next) { p = p->next; }
        p->next = h2;
        return h1;
    }
};
```

- [912. 排序数组]

```
class Solution {
public:

    const int threshold = 16;

    inline int median(int a, int b, int c) {
        if (a > b) { swap(a, b); }
        if (a > c) { swap(a, c); }
        if (b > c) { swap(b, c); }
        return b;
    }

    void __quick_sort_v3(vector<int> &arr, int l, int r) {
        while (r - l > threshold) {
            int x = l, y = r, base = median(arr[l], arr[(l + r) / 2], arr[r]);
            do {
                while (arr[x] < base) { x++; }
                while (arr[y] > base) { y--; }
                if (x <= y) {
                    swap(arr[x], arr[y]);
                    x++, y--;
                }
            } while (x <= y);
            __quick_sort_v3(arr, x, r);
            r = y;
        }
        return;
    }

    void final_insert_sort(vector<int> &arr, int l, int r) {
        int ind = l;
        for (int i = l + 1; i <= r; i++) {
            if (arr[i] < arr[ind]) { ind = i; }
        }
        while (ind > l) {
            swap(arr[ind], arr[ind - 1]);
            --ind;
        }
        for (int i = l + 2; i <= r; i++) {
            int j = i;
            while (arr[j] < arr[j - 1]) {
                swap(arr[j], arr[j - 1]);
                j--;
            }
        }
        return;
    }

    void quick_sort_v3(vector<int> &arr, int l, int r) {
        __quick_sort_v3(arr, l, r);
        final_insert_sort(arr, l, r);
        return;
    }

    vector<int> sortArray(vector<int> &nums) {

        quick_sort_v3(nums, 0, nums.size() - 1);

        return nums;
    }
};
```

- 剑指 Offer 21. 调整数组顺序使奇数位于偶数前面

```
class Solution {
public:
    vector<int> exchange(vector<int> &nums) {
        if (nums.size() == 0) { return nums; }
        int x = 0, y = nums.size() - 1;
        do {
            while (x < nums.size() && nums[x] % 2) {
                x++;
            }
            while (y >= 0 && nums[y] % 2 == 0) {
                y--;
            }
            if (x <= y) {
                swap(nums[x], nums[y]);
                x++, y--;
            }
        } while (x <= y);
        return nums;
    }
};
```

快速排序扩展

- 面试题 17.14. 最小K个数

```
class Solution {
public:
    int getmid(int a, int b, int c) {
        if (a > b) { swap(a, b); }
        if (a > c) { swap(a, c); }
        if (b > c) { swap(b, c); }
        return b;
    }

    void quick_select(vector<int> &arr, int l, int r, int k) {
        if (l >= r) { return; }
        int x = l, y = r, mid = getmid(arr[l], arr[(l + r) / 2], arr[r]);
        do {
            while (arr[x] < mid) { x++; }
            while (arr[y] > mid) { y--; }
            if (x <= y) {
                swap(arr[x], arr[y]);
                x++, y--;
            }
        } while (x <= y);
        if (y - l == k - 1) { return; } // 左区间数量等于k, 直接返回
        if (y - l >= k) { // 左区间数量大于k, 继续扩大
            quick_select(arr, l, y, k);
        } else {
            quick_select(arr, x, r, k - x + l);
        }
        return;
    }

    vector<int> smallestK(vector<int> &arr, int k) {
        vector<int> ans;
        if (k == 0) { return ans; }
        quick_select(arr, 0, arr.size() - 1, k);
        while (k) { ans.push_back(arr[--k]); }
        return ans;
    }
};
```

- 75. 颜色分类

```
class Solution {
public:
    void three_partition(vector<int> &arr, int l, int r, int mid) {
        if (l >= r) { return; }
        int x = -1, y = r + 1, i = l;
        while (i < y) {
            if (arr[i] == mid) {
                i++;
            } else if (arr[i] < mid) {
                x++;
                swap(arr[x], arr[i]);
                i++;
            } else if (arr[i] > mid) {
                y--;
                swap(arr[y], arr[i]);
            }
        }
    }

    void sortColors(vector<int> &arr) {
        three_partition(arr, 0, arr.size() - 1, 1);
        return;
    }
};
```

温故知新

- 95. 不同的二叉搜索树 II

```cpp
class Solution {
public:
    vector<TreeNode *> dfs(int l, int r) {
        vector<TreeNode *> ans;
        if (l > r) {
            ans.push_back(nullptr);
            return ans;
        }

        for (int i = l; i <= r; i++) {
            vector<TreeNode *> left_tree = dfs(l, i - 1);
            vector<TreeNode *> right_tree = dfs(i + 1, r);
            //eg : i = 3
            for (TreeNode *left : left_tree) { // 遍历left_tree {1, null, 2}, {2, 1, null}
                for (TreeNode *right : right_tree) { // {4}
                    TreeNode *t = new TreeNode(i, left, right);
                    ans.push_back(t);
                }
            }
        }
        return ans;
    }

    vector<TreeNode *> generateTrees(int n) {
        vector<TreeNode *> ans;
        if (n == 0) { return ans; }
        return dfs(1, n);
    }
};
```

- 394. 字符串解码

```cpp
class Solution {
public:
    string decodeString(string s) {
        string ret;
        int i = 0;
        while (s[i]) {
            if (s[i] < '0' || s[i] > '9') {
                ret += s[i];
                i++;
            } else {
                int num = 0;
                while (s[i] >= '0' && s[i] <= '9') {
                    num = num * 10 + (s[i++] - '0');
                }
                i++;
                int l = i, r = i, cnt = 1;
                while (cnt) {
                    r += 1;
                    if (s[r] == '[') { cnt++; }
                    else if (s[r] == ']') { cnt--; }
                }
                string tmp = decodeString(s.substr(l, r - l));
                while (num--) { ret += tmp; }
                i = r + 1;
            }
        }
        return ret;
    }
};
```

智力发散

- [11. 盛最多水的容器]

```
class Solution {
public:
    int maxArea(vector<int> &height) {
        int ans = 0, i = 0, j = height.size() - 1;
        while (i < j) {
            ans = max(ans, (j - i) * min(height[i], height[j]));

            if (height[i] < height[j]) { i++; }
            else { j--; }
        }
        return ans;
    }
};
```

- [470. 用 Rand7() 实现 Rand10()]

```
class Solution {
public:
    int rand10() {
        int x;
        while (1) {
            x = rand7();
            x = (x - 1) * 7 + rand7(); // 1 - 49
            if (x <= 40) { return x % 10 + 1; }
            x = (x - 1) * 7 + rand7(); // 1 - 63
            if (x <= 60) { return x % 10 + 1; }
            x = (x - 1) * 7 + rand7(); // 1 - 21
            if (x <= 20) { return x % 10 + 1; }
        }
        return 0;
    }
};
```

自学推荐

- 239. 滑动窗口最大值

```
class Solution {
public:
    vector<int> maxSlidingWindow(vector<int> &nums, int k) {
        vector<int> ans;
        if (k == 0) { return ans; }
        deque<int> q;

        int idx = 0;
        while (idx < nums.size()) {
            if (!q.empty() && q.front() + k <= idx) {
                q.pop_front();
            }
            while (!q.empty() && nums[q.back()] < nums[idx]) {
                q.pop_back();
            }
            q.push_back(idx);
            idx++;
            if (idx >= k) { ans.push_back(nums[q.front()]); }
        }
        return ans;
    }
};
```