

# 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

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  /*
6   * 最简单方法实现快速排序
7   */
8  void quick_sort_v1(int *arr, int l, int r) {
9      if (l >= r) { return; }
10
11     int x = l, y = r, base = arr[l];
12     while (x < y) {
13         while (x < y && arr[y] >= base) {
14             y--;
15         }
16         if (x < y) { arr[x++] = arr[y]; }
17         while (x < y && arr[x] <= base) {
18             x++;
19         }
20         if (x < y) { arr[y--] = arr[x]; }
21     }
22     arr[x] = base;
23     quick_sort_v1(arr, l, x - 1);
24     quick_sort_v1(arr, x + 1, r);
25
26     return;
27 }
28
29
30 /*
31 * 使用了单递归法实现的快速排序，同时保证了无监督
32 */
33 void quick_sort_v2(int *arr, int l, int r) {
34     while (l < r) {
35         int x = l, y = r, base = arr[l];
36         while (x < y) {
37             while (x < y && arr[y] >= base) {
38                 y--;
39             }
```

```

40         if (x < y) { arr[x++] = arr[y]; }
41         while (x < y && arr[x] <= base) {
42             x++;
43         }
44         if (x < y) { arr[y--] = arr[x]; }
45     }
46     arr[x] = base;
47     quick_sort_v2(arr, x + 1, r);
48     r = x - 1;
49 }
50
51 return;
52 }
53
54
55 /*
56  * 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
57  */
58 const int threshold = 16;
59
60 inline int median(int a, int b, int c) {
61     if (a > b) { swap(a, b); }
62     if (a > c) { swap(a, c); }
63     if (b > c) { swap(b, c); }
64     return b;
65 }
66
67 void __quick_sort_v3(int *arr, int l, int r) {
68     while (r - l > threshold) {
69         int x = l, y = r, base = median(arr[l], arr[(l + r) / 2], arr[r]);
70         do {
71             while (arr[x] < base) { x++; }
72             while (arr[y] > base) { y--; }
73             if (x <= y) {
74                 swap(arr[x], arr[y]);
75                 x++, y--;
76             }
77         } while (x <= y);
78         __quick_sort_v3(arr, x, r);
79         r = y;
80     }
81
82     return;
83 }
84
85 void final_insert_sort(int *arr, int l, int r) {
86     int ind = l;
87     for (int i = l + 1; i <= r; i++) {
88         if (arr[i] < arr[ind]) { ind = i; }
89     }
90     while (ind > l) {
91         swap(arr[ind], arr[ind - 1]);
92         --ind;
93     }
94     for (int i = l + 2; i <= r; i++) {
95         int j = i;
96         while (arr[j] < arr[j - 1]) {
97             swap(arr[j], arr[j - 1]);

```

```

98         j--;
99     }
100 }
101
102     return;
103 }
104
105 void quick_sort_v3(int *arr, int l, int r) {
106     __quick_sort_v3(arr, l, r);
107     final_insert_sort(arr, l, r);
108
109     return;
110 }
111
112
113 int main(int argc, char *argv[]) {
114     int arr[10] = {3, 1, 4, 5, 9, 10};
115     quick_sort_v1(arr, 0, 5);
116
117     for (int i = 0; i < 6; i++) {
118         cout << arr[i] << " ";
119     }
120     cout << endl;
121
122     return 0;
123 }

```

## 习题部分

### 快速排序基础

- [148. 排序链表](#)

```

1  class Solution {
2  public:
3      ListNode *sortList(ListNode *head) {
4          if (head == NULL) { return head; }
5          int l = head->val, r = head->val;
6          double mid;
7          ListNode *p = head, *q, *h1 = NULL, *h2 = NULL;
8          while (p) { l = min(p->val, l), r = max(p->val, r), p = p->next; }
9
10         if (l == r) { return head; }
11         mid = (l + r) / 2.0;
12         p = head;
13         while (p) {
14             q = p->next;
15             if (p->val <= mid) {
16                 p->next = h1;
17                 h1 = p;
18             } else {
19                 p->next = h2;
20                 h2 = p;
21             }
22             p = q;

```

```

23     }
24     h1 = sortList(h1);
25     h2 = sortList(h2);
26     p = h1;
27     while (p->next) { p = p->next; }
28     p->next = h2;
29     return h1;
30 }
31 };

```

- [912. 排序数组](#)

```

1  class Solution {
2  public:
3
4      const int threshold = 16;
5
6      inline int median(int a, int b, int c) {
7          if (a > b) { swap(a, b); }
8          if (a > c) { swap(a, c); }
9          if (b > c) { swap(b, c); }
10         return b;
11     }
12
13     void __quick_sort_v3(vector<int> &arr, int l, int r) {
14         while (r - l > threshold) {
15             int x = l, y = r, base = median(arr[l], arr[(l + r) / 2],
arr[r]);
16             do {
17                 while (arr[x] < base) { x++; }
18                 while (arr[y] > base) { y--; }
19                 if (x <= y) {
20                     swap(arr[x], arr[y]);
21                     x++, y--;
22                 }
23             } while (x <= y);
24             __quick_sort_v3(arr, x, r);
25             r = y;
26         }
27         return;
28     }
29
30     void final_insert_sort(vector<int> &arr, int l, int r) {
31         int ind = l;
32         for (int i = l + 1; i <= r; i++) {
33             if (arr[i] < arr[ind]) { ind = i; }
34         }
35         while (ind > l) {
36             swap(arr[ind], arr[ind - 1]);
37             --ind;
38         }
39         for (int i = l + 2; i <= r; i++) {
40             int j = i;
41             while (arr[j] < arr[j - 1]) {
42                 swap(arr[j], arr[j - 1]);

```

```

43         j--;
44     }
45 }
46 return;
47 }
48
49 void quick_sort_v3(vector<int> &arr, int l, int r) {
50     __quick_sort_v3(arr, l, r);
51     final_insert_sort(arr, l, r);
52     return;
53 }
54
55 vector<int> sortArray(vector<int> &nums) {
56
57     quick_sort_v3(nums, 0, nums.size() - 1);
58
59     return nums;
60 }
61 };

```

- [剑指 Offer 21. 调整数组顺序使奇数位于偶数前面](#)

```

1  class Solution {
2  public:
3      vector<int> exchange(vector<int> &nums) {
4          if (nums.size() == 0) { return nums; }
5          int x = 0, y = nums.size() - 1;
6          do {
7              while (x < nums.size() && nums[x] % 2) {
8                  x++;
9              }
10             while (y >= 0 && nums[y] % 2 == 0) {
11                 y--;
12             }
13             if (x <= y) {
14                 swap(nums[x], nums[y]);
15                 x++, y--;
16             }
17         } while (x <= y);
18         return nums;
19     }
20 };

```

## 快速排序扩展

- [面试题 17.14. 最小K个数](#)

```

1  class Solution {
2  public:
3      int getmid(int a, int b, int c) {
4          if (a > b) { swap(a, b); }
5          if (a > c) { swap(a, c); }
6          if (b > c) { swap(b, c); }

```

```

7         return b;
8     }
9
10    void quick_select(vector<int> &arr, int l, int r, int k) {
11        if (l >= r) { return; }
12        int x = l, y = r, mid = getmid(arr[l], arr[(l + r) / 2], arr[r]);
13        do {
14            while (arr[x] < mid) { x++; }
15            while (arr[y] > mid) { y--; }
16            if (x <= y) {
17                swap(arr[x], arr[y]);
18                x++, y--;
19            }
20        } while (x <= y);
21        if (y - l == k - 1) { return; } // 左区间数量等于k, 直接返回
22        if (y - l >= k) { // 左区间数量大于k, 继续扩大
23            quick_select(arr, l, y, k);
24        } else {
25            quick_select(arr, x, r, k - x + 1);
26        }
27        return;
28    }
29
30    vector<int> smallestK(vector<int> &arr, int k) {
31        vector<int> ans;
32        if (k == 0) { return ans; }
33        quick_select(arr, 0, arr.size() - 1, k);
34        while (k) { ans.push_back(arr[--k]); }
35        return ans;
36    }
37 };

```

- [75. 颜色分类](#)

```

1  class Solution {
2  public:
3      void three_partition(vector<int> &arr, int l, int r, int mid) {
4          if (l >= r) { return; }
5          int x = -1, y = r + 1, i = l;
6          while (i < y) {
7              if (arr[i] == mid) {
8                  i++;
9              } else if (arr[i] < mid) {
10                 x++;
11                 swap(arr[x], arr[i]);
12                 i++;
13             } else if (arr[i] > mid) {
14                 y--;
15                 swap(arr[y], arr[i]);
16             }
17         }
18     }
19
20     void sortColors(vector<int> &arr) {
21         three_partition(arr, 0, arr.size() - 1, 1);

```

```

22         return;
23     }
24 };

```

## 温故知新

- [95. 不同的二叉搜索树 II](#)

```

1  class Solution {
2  public:
3      vector<TreeNode*> dfs(int l, int r) {
4          vector<TreeNode*> ans;
5          if (l > r) {
6              ans.push_back(nullptr);
7              return ans;
8          }
9
10         for (int i = l; i <= r; i++) {
11             vector<TreeNode*> left_tree = dfs(l, i - 1);
12             vector<TreeNode*> right_tree = dfs(i + 1, r);
13             //eg : i = 3
14             for (TreeNode* left : left_tree) { // 遍历left_tree {1, null, 2},
{2, 1, null}
15                 for (TreeNode* right : right_tree) { // {4}
16                     TreeNode* t = new TreeNode(i, left, right);
17                     ans.push_back(t);
18                 }
19             }
20         }
21         return ans;
22     }
23
24     vector<TreeNode*> generateTrees(int n) {
25         vector<TreeNode*> ans;
26         if (n == 0) { return ans; }
27         return dfs(1, n);
28     }
29 };

```

- [394. 字符串解码](#)

```

1  class Solution {
2  public:
3      string decodeString(string s) {
4          string ret;
5          int i = 0;
6          while (s[i]) {
7              if (s[i] < '0' || s[i] > '9') {
8                  ret += s[i];
9                  i++;
10             } else {
11                 int num = 0;
12                 while (s[i] >= '0' && s[i] <= '9') {

```

```

13         num = num * 10 + (s[i++] - '0');
14     }
15     i++;
16     int l = i, r = i, cnt = 1;
17     while (cnt) {
18         r += 1;
19         if (s[r] == '[') { cnt++; }
20         else if (s[r] == ']') { cnt--; }
21     }
22     string tmp = decodeString(s.substr(l, r - l));
23     while (num--) { ret += tmp; }
24     i = r + 1;
25 }
26 }
27 return ret;
28 }
29 };

```

## 智力发散

- [11. 盛最多水的容器](#)

```

1  class Solution {
2  public:
3      int maxArea(vector<int> &height) {
4          int ans = 0, i = 0, j = height.size() - 1;
5          while (i < j) {
6              ans = max(ans, (j - i) * min(height[i], height[j]));
7
8              if (height[i] < height[j]) { i++; }
9              else { j--; }
10         }
11         return ans;
12     }
13 };

```

- [470. 用 Rand7\(\) 实现 Rand10\(\)](#)

```

1  class Solution {
2  public:
3      int rand10() {
4          int x;
5          while (1) {
6              x = rand7();
7              x = (x - 1) * 7 + rand7(); // 1 - 49
8              if (x <= 40) { return x % 10 + 1; }
9              x = (x - 1) * 7 + rand7(); // 1 - 63
10             if (x <= 60) { return x % 10 + 1; }
11             x = (x - 1) * 7 + rand7(); // 1 - 21
12             if (x <= 20) { return x % 10 + 1; }
13         }
14         return 0;
15     }

```



## 自学推荐

- [239. 滑动窗口最大值](#)

```
1  class Solution {
2  public:
3      vector<int> maxSlidingWindow(vector<int> &nums, int k) {
4          vector<int> ans;
5          if (k == 0) { return ans; }
6          deque<int> q;
7
8          int idx = 0;
9          while (idx < nums.size()) {
10             if (!q.empty() && q.front() + k <= idx) {
11                 q.pop_front();
12             }
13             while (!q.empty() && nums[q.back()] < nums[idx]) {
14                 q.pop_back();
15             }
16             q.push_back(idx);
17             idx++;
18             if (idx >= k) { ans.push_back(nums[q.front()]); }
19         }
20         return ans;
21     }
22 };
```