Leetcode真题刷题讲解代码

> 京程一灯

Leetcode简单题部分

题号 #1 两数之和

解法1：

```java
public class Solution {
    public int[] twoSum(int[] nums, int target) {
        HashMap<Integer, Integer> m = new HashMap<Integer, Integer>();
        int[] res = new int[2];
        for (int i = 0; i < nums.length; ++i) {
            m.put(nums[i], i);
        }
        for (int i = 0; i < nums.length; ++i) {
            int t = target - nums[i];
            if (m.containsKey(t) && m.get(t) != i) {
                res[0] = i;
                res[1] = m.get(t);
                break;
            }
        }
        return res;
    }
}
```

解法2:

```java
public class Solution {
    public int[] twoSum(int[] nums, int target) {
        HashMap<Integer, Integer> m = new HashMap<Integer, Integer>();
        int[] res = new int[2];
        for (int i = 0; i < nums.length; ++i) {
            if (m.containsKey(target - nums[i])) {
                res[0] = i;
                res[1] = m.get(target - nums[i]);
                break;
            }
            m.put(nums[i], i);
        }
        return res;
    }
}
```

题号 #7 整数反转

解法1：

```java
public class Solution {
    public int reverse(int x) {
        int res = 0;
        while (x != 0) {
            int t = res * 10 + x % 10;
            if (t / 10 != res) return 0;
            res = t;
            x /= 10;
        }
        return res;
    }
};
```

解法2:

```java
public class Solution {
    public int reverse(int x) {
        int res = 0;
        while (x != 0) {
            if (Math.abs(res) > INT_MAX / 10) return 0;
            res = res * 10 + x % 10;
            x /= 10;
        }
        return res;
    }
};
```

题号 #14 最长公共前缀

解法1：

```java
public class Solution {
    public String longestCommonPrefix(String[] strs) {
        if (strs == null || strs.length == 0) return "";
        String res = new String();
        for (int j = 0; j < strs[0].length(); ++j) {
            char c = strs[0].charAt(j);
            for (int i = 1; i < strs.length; ++i) {
                if (j >= strs[i].length() || strs[i].charAt(j) != c) {
                    return res;
                }
            }
```

```
                res += Character.toString(c);
            }
            return res;
        }
    }
```

解法2:

```

class Solution {
    public String longestCommonPrefix(String[] strs) {
        if (strs == null || strs.length == 0) return "";
        Arrays.sort(strs);
        int i = 0, len = Math.min(strs[0].length(), strs[strs.length - 1].length());
        while (i < len && strs[0].charAt(i) == strs[strs.length - 1].charAt(i)) i++;
        return strs[0].substring(0, i);
    }
}
```

题号 #13 罗马数字转整数

解法1:

```
class Solution {
    public int romanToInt(String s) {
        Map<Character,Integer> map=new HashMap<>();
        map.put('I',1);
        map.put('V',5);
        map.put('X',10);
        map.put('L',50);
        map.put('C',100);
        map.put('D',500);
        map.put('M',1000);
        StringBuffer rev=new StringBuffer();
        rev.append(s);
        char[] num=rev.reverse().toString().toCharArray();
        int res= map.get(num[0]);
        for(int i=1;i<num.length;i++){
            if((int)map.get(num[i-1])>(int)map.get(num[i])){
                res=res-map.get(num[i]);
            }else
                res+=map.get(num[i]);
        }
        return res;
    }
}
```

解法2:

```
class Solution {
    public int romanToInt(String s) {
        int end=0;

        char[] cha = s.toCharArray();
        for(int i=cha.length-1;i>=0;i--){
            if(cha[i]=='I')
                end+=1;
            else if(cha[i]=='V')end+=(i-1)>=0&&cha[i-1]=='I'?4+i-i--:5;
            else if(cha[i]=='X')end+=(i-1)>=0&&cha[i-1]=='I'?9+i-i--:10;
            else if(cha[i]=='L')end+=(i-1)>=0&&cha[i-1]=='X'?40+i-i--:50;
            else if(cha[i]=='C')end+=(i-1)>=0&&cha[i-1]=='X'?90+i-i--:100;
            else if(cha[i]=='D')end+=(i-1)>=0&&cha[i-1]=='C'?400+i-i--:500;
            else if(cha[i]=='M')end+=(i-1)>=0&&cha[i-1]=='C'?900+i-i--:1000;
        }

        return end<4000?end:0;
    }
}
```

leetcode 中等题部分

题号 #2 两数相加

解法:

```
public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
    ListNode dummyHead = new ListNode(0);
    ListNode p = l1, q = l2, curr = dummyHead;
```

```java
    int carry = 0;
    while (p != null || q != null) {
        int x = (p != null) ? p.val : 0;
        int y = (q != null) ? q.val : 0;
        int sum = carry + x + y;
        carry = sum / 10;
        curr.next = new ListNode(sum % 10);
        curr = curr.next;
        if (p != null) p = p.next;
        if (q != null) q = q.next;
    }
    if (carry > 0) {
        curr.next = new ListNode(carry);
    }
    return dummyHead.next;
}
```

题号 #5 最长回文子串

解法:

```java
public String longestPalindrome(String s) {
    if (s == null || s.length() < 1) return "";
    int start = 0, end = 0;
    for (int i = 0; i < s.length(); i++) {
        int len1 = expandAroundCenter(s, i, i);
        int len2 = expandAroundCenter(s, i, i + 1);
        int len = Math.max(len1, len2);
        if (len > end - start) {
            start = i - (len - 1) / 2;
            end = i + len / 2;
        }
    }
    return s.substring(start, end + 1);
}

private int expandAroundCenter(String s, int left, int right) {
    int L = left, R = right;
    while (L >= 0 && R < s.length() && s.charAt(L) == s.charAt(R)) {
        L--;
        R++;
    }
    return R - L - 1;
}
```

题号 #19 删除链表的倒数第N个点

解法1:

```java
public ListNode removeNthFromEnd(ListNode head, int n) {
    ListNode dummy = new ListNode(0);
    dummy.next = head;
    int length  = 0;
    ListNode first = head;
    while (first != null) {
        length++;
        first = first.next;
    }
    length -= n;
    first = dummy;
    while (length > 0) {
        length--;
        first = first.next;
    }
    first.next = first.next.next;
    return dummy.next;
}
```

解法2:

```java
public ListNode removeNthFromEnd(ListNode head, int n) {
    ListNode dummy = new ListNode(0);
    dummy.next = head;
    ListNode first = dummy;
    ListNode second = dummy;
    // Advances first pointer so that the gap between first and second
is n nodes apart
    for (int i = 1; i <= n + 1; i++) {
        first = first.next;
    }
    // Move first to the end, maintaining the gap
    while (first != null) {
        first = first.next;
        second = second.next;
    }
    second.next = second.next.next;
    return dummy.next;
}
```

Leetcode 难题部分

题号 #4 寻找两个有序数组的中位数

解法：

```java
class Solution {
    public double findMedianSortedArrays(int[] A, int[] B) {
        int m = A.length;
        int n = B.length;
        if (m > n) { // to ensure m<=n
            int[] temp = A; A = B; B = temp;
            int tmp = m; m = n; n = tmp;
        }
        int iMin = 0, iMax = m, halfLen = (m + n + 1) / 2;
        while (iMin <= iMax) {
            int i = (iMin + iMax) / 2;
            int j = halfLen - i;
            if (i < iMax && B[j-1] > A[i]){
                iMin = i + 1; // i is too small
            }
            else if (i > iMin && A[i-1] > B[j]) {
                iMax = i - 1; // i is too big
            }
            else { // i is perfect
                int maxLeft = 0;
                if (i == 0) { maxLeft = B[j-1]; }
                else if (j == 0) { maxLeft = A[i-1]; }
                else { maxLeft = Math.max(A[i-1], B[j-1]); }
                if ( (m + n) % 2 == 1 ) { return maxLeft; }

                int minRight = 0;
                if (i == m) { minRight = B[j]; }
                else if (j == n) { minRight = A[i]; }
                else { minRight = Math.min(B[j], A[i]); }

                return (maxLeft + minRight) / 2.0;
            }
        }
        return 0.0;
    }
}
```

题号 #23 合并K个排序链表

解法:

```
class Solution {
private:
struct cmp
{
    bool operator ()(const ListNode *a, const ListNode *b)
    {
            return a->val > b->val;
    }
};
public:
    ListNode *mergeKLists(vector<ListNode *> &lists) {
        int n = lists.size();
        if(n == 0)return NULL;
        ListNode node(0), *res = &node;
        priority_queue<ListNode*, vector<ListNode*>, cmp> que;
        for(int i = 0; i < n; i++)
            if(lists[i])
                que.push(lists[i]);
        while(!que.empty())
        {
            ListNode * p = que.top();
            que.pop();
            res->next = p;
            res = p;

            if(p->next)
                que.push(p->next);
        }
        return node.next;
    }
};
```

题号#30 与所有单词相关联的字串

解法1:

```
class Solution {
public:
    vector<int> findSubstring(string s, vector<string>& words) {
        vector<int> res;
        if (s.empty() || words.empty()) return res;
```

```cpp
        int n = words.size(), m = words[0].size();
        unordered_map<string, int> m1;
        for (auto &a : words) ++m1[a];
        for (int i = 0; i <= (int)s.size() - n * m; ++i) {
            unordered_map<string, int> m2;
            int j = 0;
            for (j = 0; j < n; ++j) {
                string t = s.substr(i + j * m, m);
                if (m1.find(t) == m1.end()) break;
                ++m2[t];
                if (m2[t] > m1[t]) break;
            }
            if (j == n) res.push_back(i);
        }
        return res;
    }
};
```

解法2:

```cpp
class Solution {
public:
    vector<int> findSubstring(string s, vector<string>& words) {
        if (s.empty() || words.empty()) return {};
        vector<int> res;
        int n = s.size(), cnt = words.size(), len = words[0].size();
        unordered_map<string, int> m1;
        for (string w : words) ++m1[w];
        for (int i = 0; i < len; ++i) {
            int left = i, count = 0;
            unordered_map<string, int> m2;
            for (int j = i; j <= n - len; j += len) {
                string t = s.substr(j, len);
                if (m1.count(t)) {
                    ++m2[t];
                    if (m2[t] <= m1[t]) {
                        ++count;
                    } else {
                        while (m2[t] > m1[t]) {
                            string t1 = s.substr(left, len);
                            --m2[t1];
                            if (m2[t1] < m1[t1]) --count;
                            left += len;
                        }
                    }
                    if (count == cnt) {
                        res.push_back(left);
                        --m2[s.substr(left, len)];
                        --count;
                        left += len;
                    }
                } else {
                    m2.clear();
                    count = 0;
                    left = j + len;
                }
            }
        }
        return res;
    }
};
```

题号 #37 解数独

解法：

```cpp
class Solution {
public:
    void solveSudoku(vector<vector<char> > &board) {
        if (board.empty() || board.size() != 9 || board[0].size() != 9) return;
        solveSudokuDFS(board, 0, 0);
    }
    bool solveSudokuDFS(vector<vector<char> > &board, int i, int j) {
        if (i == 9) return true;
        if (j >= 9) return solveSudokuDFS(board, i + 1, 0);
        if (board[i][j] == '.') {
            for (int k = 1; k <= 9; ++k) {
                board[i][j] = (char)(k + '0');
                if (isValid(board, i , j)) {
                    if (solveSudokuDFS(board, i, j + 1)) return true;
                }
                board[i][j] = '.';
            }
        } else {
            return solveSudokuDFS(board, i, j + 1);
        }
        return false;
    }
    bool isValid(vector<vector<char> > &board, int i, int j) {
        for (int col = 0; col < 9; ++col) {
            if (col != j && board[i][j] == board[i][col]) return false;
        }
        for (int row = 0; row < 9; ++row) {
            if (row != i && board[i][j] == board[row][j]) return false;
        }
        for (int row = i / 3 * 3; row < i / 3 * 3 + 3; ++row) {
            for (int col = j / 3 * 3; col < j / 3 * 3 + 3; ++col) {
                if ((row != i || col != j) && board[i][j] == board[row][col]) return false;
            }
        }
        return true;
    }
};
```

题号 #51 N皇后

解法:

```
class Solution {
public:
    vector<vector<string> > solveNQueens(int n) {
        vector<vector<string> > res;
        vector<int> pos(n, -1);
        solveNQueensDFS(pos, 0, res);
        return res;
    }
    void solveNQueensDFS(vector<int> &pos, int row, vector<vector<string> > &res) {
        int n = pos.size();
        if (row == n) {
            vector<string> out(n, string(n, '.'));
            for (int i = 0; i < n; ++i) {
                out[i][pos[i]] = 'Q';
            }
            res.push_back(out);
        } else {
            for (int col = 0; col < n; ++col) {
                if (isValid(pos, row ,col)) {
                    pos[row] = col;
                    solveNQueensDFS(pos, row + 1, res);
                    pos[row] = -1;
                }
            }
        }
    }
    bool isValid(vector<int> &pos, int row, int col) {
        for (int i = 0; i < row; ++i) {
            if (col == pos[i] || abs(row - i) == abs(col - pos[i])) {
                return false;
            }
        }
        return true;
    }
};
```

题号 # 749 隔离病毒

```
class Solution {
public:
    int containVirus(vector<vector<int>>& grid) {
        int res = 0, m = grid.size(), n = grid[0].size();
        vector<vector<int>> dirs{{-1,0},{0,1},{1,0},{0,-1}};
        while (true) {
            unordered_set<int> visited;
            vector<vector<vector<int>>> all;
            for (int i = 0; i < m; ++i) {
                for (int j = 0; j < n; ++j) {
                    if (grid[i][j] == 1 && !visited.count(i * n + j)) {
                        queue<int> q{{i * n + j}};
                        vector<int> virus{i * n + j};
                        vector<int> walls;
                        visited.insert(i * n + j);
                        while (!q.empty()) {
                            auto t = q.front(); q.pop();
                            for (auto dir : dirs) {
                                int x = (t / n) + dir[0], y = (t % n) + dir[1];
                                if (x < 0 || x >= m || y < 0 || y >= n || visited.count(x * n + y)) continue;
                                if (grid[x][y] == -1) continue;
                                else if (grid[x][y] == 0) walls.push_back(x * n + y);
                                else if (grid[x][y] == 1) {
                                    visited.insert(x * n + y);
                                    virus.push_back(x * n + y);
                                    q.push(x * n + y);
                                }
                            }
                        }
                        unordered_set<int> s(walls.begin(), walls.end());
                        vector<int> cells{(int)s.size()};
                        all.push_back({cells ,walls, virus});
                    }
                }
            }
            if (all.empty()) break;
```

```cpp
            sort(all.begin(), all.end(), [](vector<vector<int>> &a, vector<vector<int>> &b) {return a[0][0] > b[0][0];});
            for (int i = 0; i < all.size(); ++i) {
                if (i == 0) {
                    vector<int> virus = all[0][2];
                    for (int idx : virus) grid[idx / n][idx % n] = -1;
                    res += all[0][1].size();
                } else {
                    vector<int> wall = all[i][1];
                    for (int idx : wall) grid[idx / n][idx % n] = 1;
                }
            }
        }
        return res;
    }
};
```