【第十五课】深搜与广搜-课堂笔记

1.LeetCode 993. 二叉树的堂兄弟节点

深搜

```cpp
class Solution {
public:
    int dfs(TreeNode *root, int x, TreeNode *&father) {
        if (root == nullptr) return -1;
        if (root->val == x) return 0;
        int l;
        father = root;
        l = dfs(root->left, x, father);
        if (l != -1) return l + 1;
        father = root;
        l = dfs(root->right, x, father);
        if (l != -1) return l + 1;
        return -1;
    }
    bool isCousins(TreeNode* root, int x, int y) {
        int d1, d2;
        TreeNode *father_x = nullptr, *father_y = nullptr;
        d1 = dfs(root, x, father_x);
        d2 = dfs(root, y, father_y);
        return d1 == d2 && father_x != father_y;
    }
};
```

广搜

```cpp
class Solution {
public:
    struct Data {
        Data(TreeNode *node = nullptr, TreeNode *father = nullptr, int deepth = 0)
        : node(node), father(father), deepth(deepth) {}
        TreeNode *node, *father;
        int deepth;
    };
    bool isCousins(TreeNode* root, int x, int y) {
        int d1 = -1, d2 = -1;
        TreeNode *father_x = nullptr, *father_y = nullptr;
        queue<Data> q;
        q.push(Data(root, nullptr, 0));
        while (!q.empty()) {
            Data cur = q.front();
            if (cur.node->val == x) d1 = cur.deepth, father_x = cur.father;
            if (cur.node->val == y) d2 = cur.deepth, father_y = cur.father;
            if (cur.node->left) {
                q.push(Data(cur.node->left, cur.node, cur.deepth + 1));
            }
            if (cur.node->right) {
                q.push(Data(cur.node->right, cur.node, cur.deepth + 1));
            }
            q.pop();
        }
        return d1 == d2 && father_x != father_y;
    }
};
```

2.LeetCode 542. 01 矩阵

```
class Solution {
public:
    struct Data {
        Data(int i = 0, int j = 0, int k = 0)
        : i(i), j(j), k(k) {}
        int i, j, k;
    };
    void init_queue(
        queue<Data> &q, vector<vector<int>> &vis,
        int n, int m, vector<vector<int>> &mat
    ) {
        for (int i = 0; i < n; i++) {
            vis.push_back(vector<int>());
            for (int j = 0; j < m; j++) {
                vis[i].push_back(-1);
            }
        }
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                if (mat[i][j]) continue;
                vis[i][j] = 0;
                q.push(Data(i, j, 0));
            }
        }
        return ;
    }
    int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
    vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {
        int n = mat.size(), m = mat[0].size();
        queue<Data> q;
        vector<vector<int>> vis;
        init_queue(q, vis, n, m, mat);
        while (!q.empty()) {
            Data cur = q.front();
            for (int k = 0; k < 4; k++) {
                int x = cur.i + dir[k][0];
                int y = cur.j + dir[k][1];
                if (x < 0 || x >= n) continue;
                if (y < 0 || y >= m) continue;
                if (vis[x][y] != -1) continue;
                vis[x][y] = cur.k + 1;
                q.push(Data(x, y, cur.k + 1));
            }
            q.pop();
        }
        return vis;
    }
};
```

3.LeetCode 1091. 二进制矩阵中的最短路径

```
class Solution {
public:
    struct Data {
        Data(int i = 0, int j = 0, int l = 0)
        : i(i), j(j), l(l) {}
        int i, j, l;
    };
    int dir[8][2] = {
        0, 1, 1, 0, 0, -1, -1, 0,
        1, -1, -1, 1, 1, 1, -1, -1
    };
    int shortestPathBinaryMatrix(vector<vector<int>>& grid) {
        int n = grid.size();
        vector<vector<int>> vis;
        for (int i = 0; i < n; i++) {
            vis.push_back(vector<int>(n));
        }
        queue<Data> q;
        if (grid[0][0]) return -1;
        vis[0][0] = 1;
        q.push(Data(0, 0, 1));
        while (!q.empty()) {
            Data cur = q.front();
            if (cur.i == n - 1 && cur.j == n - 1) return cur.l;
            for (int k = 0; k < 8; k++) {
                int x = cur.i + dir[k][0];
                int y = cur.j + dir[k][1];
                if (x < 0 || x >= n) continue;
                if (y < 0 || y >= n) continue;
                if (grid[x][y]) continue;
                if (vis[x][y]) continue;
                vis[x][y] = 1;
                q.push(Data(x, y, cur.l + 1));
            }
            q.pop();
        }
        return -1;
    }
};
```

4.LeetCode 752. 打开转盘锁

```
class Solution {
public:
    struct Data {
        Data(string s = "", int l = 0)
        : s(s), l(l) {}
        string s;
        int l;
    };
    string getS(string &s, int i, int k) {
        string ret = s;
        switch (k) {
            case 0: ret[i] += 1; break;
            case 1: ret[i] -= 1; break;
        }
        if (ret[i] < '0') ret[i] = '9';
        if (ret[i] > '9') ret[i] = '0';
        return ret;
    }
    int openLock(vector<string>& deadends, string target) {
        queue<Data> q;
        unordered_set<string> h;
        for (auto x : deadends) h.insert(x);
        if (h.find("0000") != h.end()) return -1;
        h.insert("0000");
        q.push(Data("0000", 0));
        while (!q.empty()) {
            Data cur = q.front();
            if (cur.s == target) return cur.l;
            for (int i = 0; i < 4; i++) {
                for (int k = 0; k < 2; k++) {
                    string t = getS(cur.s, i, k);
                    if (h.find(t) != h.end()) continue;
                    h.insert(t);
                    q.push(Data(t, cur.l + 1));
                }
            }
            q.pop();
        }
        return -1;
    }
};
```

5.剑指 Offer 13. 机器人的运动范围

```cpp
class Solution {
public:
    struct Data {
        Data(int i = 0, int j = 0)
        : i(i), j(j) {}
        int i, j;
    };
    int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
    int movingCount(int m, int n, int k) {
        vector<int> dsum(100);
        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 10; j++) {
                dsum[i * 10 + j] = i + j;
            }
        }
        queue<Data> q;
        unordered_set<int> h;
        h.insert(0);
        q.push(Data(0, 0));
        int ans = 0;
        while (!q.empty()) {
            Data cur = q.front();
            ans += 1;
            for (int i = 0; i < 4; i++) {
                int x = cur.i + dir[i][0];
                int y = cur.j + dir[i][1];
                if (x < 0 || x >= m) continue;
                if (y < 0 || y >= n) continue;
                if (h.find(x * n + y) != h.end()) continue;
                if (dsum[x] + dsum[y] > k) continue;
                h.insert(x * n + y);
                q.push(Data(x, y));
            }
            q.pop();
        }
        return ans;
    }
};
```

6.LeetCode 130. 被围绕的区域

```cpp
class Solution {
public:
    int n, m;
    int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
    void dfs(int i, int j, vector<vector<char>> &board) {
        board[i][j] = 'o';
        for (int k = 0; k < 4; k++) {
            int x = i + dir[k][0];
            int y = j + dir[k][1];
            if (x < 0 || x >= n) continue;
            if (y < 0 || y >= m) continue;
            if (board[x][y] != 'O') continue;
            dfs(x, y, board);
        }
        return ;
    }
    void solve(vector<vector<char>>& board) {
        n = board.size(), m = board[0].size();
        for (int i = 0; i < n; i++) {
            if (board[i][0] == 'O') dfs(i, 0, board);
            if (board[i][m - 1] == 'O') dfs(i, m - 1, board);
        }
        for (int j = 0; j < m; j++) {
            if (board[0][j] == 'O') dfs(0, j, board);
            if (board[n - 1][j] == 'O') dfs(n - 1, j, board);
        }
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                if (board[i][j] == 'O') board[i][j] = 'X';
                else if (board[i][j] == 'o') board[i][j] = 'O';
            }
        }
        return ;
    }
};
```

7.LeetCode 494. 目标和

```cpp
class Solution {
public:
    typedef pair<int, int> PII;
    struct CMP {
        int operator()(const PII &a) const {
            return a.first ^ a.second;
        }
    };
    unordered_map<PII, int, CMP> h;
    int dfs(int i, int target, vector<int> &nums) {
        if (i == nums.size()) {
            return target == 0;
        }
        if (h.find(PII(i, target)) != h.end()) {
            return h[PII(i, target)];
        }
        int ans = 0;
        ans += dfs(i + 1, target - nums[i], nums); // +
        ans += dfs(i + 1, target + nums[i], nums); // -
        h[PII(i, target)] = ans;
        return ans;
    }
    int findTargetSumWays(vector<int>& nums, int target) {
        h.clear();
        return dfs(0, target, nums);
    }
};
```

8.LeetCode 473. 火柴拼正方形

```
class Solution {
public:
    bool dfs(int ind, vector<int> &arr, vector<int> &ms) {
        if (ind == -1) return true;
        for (int i = 0; i < 4; i++) {
            if (arr[i] < ms[ind]) continue;
            if (arr[i] == ms[ind] || arr[i] >= ms[ind] + ms[0]) {
                arr[i] -= ms[ind];
                if (dfs(ind - 1, arr, ms)) return true;
                arr[i] += ms[ind];
            }
        }
        return false;
    }
    bool makesquare(vector<int>& matchsticks) {
        sort(matchsticks.begin(), matchsticks.end());
        vector<int> arr(4);
        int sum = 0;
        for (auto x : matchsticks) sum += x;
        if (sum % 4) return false;
        for (int i = 0; i < 4; i++) arr[i] = sum / 4;
        return dfs(matchsticks.size() - 1, arr, matchsticks);
    }
};
```

9.LeetCode 39. 组合总和

```
class Solution {
public:
    void dfs(int ind, int target, vector<int> &nums,
        vector<int> &buff, vector<vector<int>> &ret
    ) {
        if (target < 0) return ;
        if (target == 0) {
            ret.push_back(buff);
            return ;
        }
        if (ind == nums.size()) return ;
        dfs(ind + 1, target, nums, buff, ret);
        buff.push_back(nums[ind]);
        dfs(ind, target - nums[ind], nums, buff, ret);
```

```cpp
        buff.pop_back();
        return ;
    }
    vector<vector<int>> combinationSum(vector<int>& candidates, int target) {
        vector<int> buff;
        vector<vector<int>> ret;
        dfs(0, target, candidates, buff, ret);
        return ret;
    }
};
```

10.LeetCode 51. N 皇后

答案在彩蛋里，彩蛋作业敬请期待~