

## 【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

```
1 class Solution {
2 public:
3     int dfs(TreeNode *root, int x, TreeNode *&father) {
4         if (root == nullptr) return -1;
5         if (root->val == x) return 0;
6         int l;
7         father = root;
8         l = dfs(root->left, x, father);
9         if (l != -1) return l + 1;
10        father = root;
11        l = dfs(root->right, x, father);
12        if (l != -1) return l + 1;
13        return -1;
14    }
15    bool isCousins(TreeNode* root, int x, int y) {
16        int d1, d2;
17        TreeNode *father_x = nullptr, *father_y = nullptr;
18        d1 = dfs(root, x, father_x);
19        d2 = dfs(root, y, father_y);
20        return d1 == d2 && father_x != father_y;
21    }
22};
```

广搜

```
1 class Solution {
2 public:
3     struct Data {
4         Data(TreeNode *node = nullptr, TreeNode *father = nullptr, int depth =
5         0)
6             : node(node), father(father), depth(depth) {}
7         TreeNode *node, *father;
8         int depth;
9     };
10    bool isCousins(TreeNode* root, int x, int y) {
11        int d1 = -1, d2 = -1;
12        TreeNode *father_x = nullptr, *father_y = nullptr;
13        queue<Data> q;
14        q.push(Data(root, nullptr, 0));
15        while (!q.empty()) {
16            Data cur = q.front();
17            if (cur.node->val == x) d1 = cur.depth, father_x = cur.father;
```

```

17         if (cur.node->val == y) d2 = cur.depth, father_y = cur.father;
18         if (cur.node->left) {
19             q.push(Data(cur.node->left, cur.node, cur.depth + 1));
20         }
21         if (cur.node->right) {
22             q.push(Data(cur.node->right, cur.node, cur.depth + 1));
23         }
24         q.pop();
25     }
26     return d1 == d2 && father_x != father_y;
27 }
28 };

```

## 2. LeetCode 542. 01 矩阵

```

1 class Solution {
2 public:
3     struct Data {
4         Data(int i = 0, int j = 0, int k = 0)
5             : i(i), j(j), k(k) {}
6         int i, j, k;
7     };
8     void init_queue(
9         queue<Data> &q, vector<vector<int>> &vis,
10        int n, int m, vector<vector<int>> &mat
11    ) {
12        for (int i = 0; i < n; i++) {
13            vis.push_back(vector<int>());
14            for (int j = 0; j < m; j++) {
15                vis[i].push_back(-1);
16            }
17        }
18        for (int i = 0; i < n; i++) {
19            for (int j = 0; j < m; j++) {
20                if (mat[i][j]) continue;
21                vis[i][j] = 0;
22                q.push(Data(i, j, 0));
23            }
24        }
25        return ;
26    }
27    int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
28    vector<vector<int>> updateMatrix(vector<vector<int>> & mat) {
29        int n = mat.size(), m = mat[0].size();
30        queue<Data> q;
31        vector<vector<int>> vis;
32        init_queue(q, vis, n, m, mat);
33        while (!q.empty()) {
34            Data cur = q.front();
35            for (int k = 0; k < 4; k++) {
36                int x = cur.i + dir[k][0];

```

```

37         int y = cur.j + dir[k][1];
38         if (x < 0 || x >= n) continue;
39         if (y < 0 || y >= m) continue;
40         if (vis[x][y] != -1) continue;
41         vis[x][y] = cur.k + 1;
42         q.push(Data(x, y, cur.k + 1));
43     }
44     q.pop();
45 }
46 return vis;
47 }
48 };

```

### 3. LeetCode 1091. 二进制矩阵中的最短路径

```

1 class Solution {
2 public:
3     struct Data {
4         Data(int i = 0, int j = 0, int l = 0)
5             : i(i), j(j), l(l) {}
6         int i, j, l;
7     };
8     int dir[8][2] = {
9         0, 1, 1, 0, 0, -1, -1, 0,
10        1, -1, -1, 1, 1, 1, -1, -1
11    };
12    int shortestPathBinaryMatrix(vector<vector<int>>& grid) {
13        int n = grid.size();
14        vector<vector<int>> vis;
15        for (int i = 0; i < n; i++) {
16            vis.push_back(vector<int>(n));
17        }
18        queue<Data> q;
19        if (grid[0][0]) return -1;
20        vis[0][0] = 1;
21        q.push(Data(0, 0, 1));
22        while (!q.empty()) {
23            Data cur = q.front();
24            if (cur.i == n - 1 && cur.j == n - 1) return cur.l;
25            for (int k = 0; k < 8; k++) {
26                int x = cur.i + dir[k][0];
27                int y = cur.j + dir[k][1];
28                if (x < 0 || x >= n) continue;
29                if (y < 0 || y >= n) continue;
30                if (grid[x][y]) continue;
31                if (vis[x][y]) continue;
32                vis[x][y] = 1;
33                q.push(Data(x, y, cur.l + 1));
34            }
35            q.pop();
36        }

```

```
37     return -1;
38 }
39 };
```

#### 4. LeetCode 752. 打开转盘锁

```
1 class Solution {
2 public:
3     struct Data {
4         Data(string s = "", int l = 0)
5             : s(s), l(l) {}
6         string s;
7         int l;
8     };
9     string getS(string &s, int i, int k) {
10         string ret = s;
11         switch (k) {
12             case 0: ret[i] += 1; break;
13             case 1: ret[i] -= 1; break;
14         }
15         if (ret[i] < '0') ret[i] = '9';
16         if (ret[i] > '9') ret[i] = '0';
17         return ret;
18     }
19     int openLock(vector<string>& deadends, string target) {
20         queue<Data> q;
21         unordered_set<string> h;
22         for (auto x : deadends) h.insert(x);
23         if (h.find("0000") != h.end()) return -1;
24         h.insert("0000");
25         q.push(Data("0000", 0));
26         while (!q.empty()) {
27             Data cur = q.front();
28             if (cur.s == target) return cur.l;
29             for (int i = 0; i < 4; i++) {
30                 for (int k = 0; k < 2; k++) {
31                     string t = getS(cur.s, i, k);
32                     if (h.find(t) != h.end()) continue;
33                     h.insert(t);
34                     q.push(Data(t, cur.l + 1));
35                 }
36             }
37             q.pop();
38         }
39         return -1;
40     }
41 };
```

## 5.剑指 Offer 13. 机器人的运动范围

```
1 class Solution {
2 public:
3     struct Data {
4         Data(int i = 0, int j = 0)
5             : i(i), j(j) {}
6         int i, j;
7     };
8     int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
9     int movingCount(int m, int n, int k) {
10         vector<int> dsum(100);
11         for (int i = 0; i < 10; i++) {
12             for (int j = 0; j < 10; j++) {
13                 dsum[i * 10 + j] = i + j;
14             }
15         }
16         queue<Data> q;
17         unordered_set<int> h;
18         h.insert(0);
19         q.push(Data(0, 0));
20         int ans = 0;
21         while (!q.empty()) {
22             Data cur = q.front();
23             ans += 1;
24             for (int i = 0; i < 4; i++) {
25                 int x = cur.i + dir[i][0];
26                 int y = cur.j + dir[i][1];
27                 if (x < 0 || x >= m) continue;
28                 if (y < 0 || y >= n) continue;
29                 if (h.find(x * n + y) != h.end()) continue;
30                 if (dsum[x] + dsum[y] > k) continue;
31                 h.insert(x * n + y);
32                 q.push(Data(x, y));
33             }
34             q.pop();
35         }
36         return ans;
37     }
38 };
```

## 6.LeetCode 130. 被围绕的区域

```
1 class Solution {
2 public:
3     int n, m;
4     int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
5     void dfs(int i, int j, vector<vector<char>> &board) {
6         board[i][j] = 'o';
7         for (int k = 0; k < 4; k++) {
8             int x = i + dir[k][0];
```

```

9         int y = j + dir[k][1];
10        if (x < 0 || x >= n) continue;
11        if (y < 0 || y >= m) continue;
12        if (board[x][y] != 'O') continue;
13        dfs(x, y, board);
14    }
15    return ;
16 }
17 void solve(vector<vector<char>>& board) {
18     n = board.size(), m = board[0].size();
19     for (int i = 0; i < n; i++) {
20         if (board[i][0] == 'O') dfs(i, 0, board);
21         if (board[i][m - 1] == 'O') dfs(i, m - 1, board);
22     }
23     for (int j = 0; j < m; j++) {
24         if (board[0][j] == 'O') dfs(0, j, board);
25         if (board[n - 1][j] == 'O') dfs(n - 1, j, board);
26     }
27     for (int i = 0; i < n; i++) {
28         for (int j = 0; j < m; j++) {
29             if (board[i][j] == 'O') board[i][j] = 'X';
30             else if (board[i][j] == 'o') board[i][j] = 'O';
31         }
32     }
33     return ;
34 }
35 };

```

## 7. LeetCode 494. 目标和

```

1 class Solution {
2 public:
3     typedef pair<int, int> PII;
4     struct CMP {
5         int operator()(const PII &a) const {
6             return a.first ^ a.second;
7         }
8     };
9     unordered_map<PII, int, CMP> h;
10    int dfs(int i, int target, vector<int> &nums) {
11        if (i == nums.size()) {
12            return target == 0;
13        }
14        if (h.find(PII(i, target)) != h.end()) {
15            return h[PII(i, target)];
16        }
17        int ans = 0;
18        ans += dfs(i + 1, target - nums[i], nums); // +
19        ans += dfs(i + 1, target + nums[i], nums); // -
20        h[PII(i, target)] = ans;
21        return ans;

```

```

22     }
23     int findTargetSumWays(vector<int>& nums, int target) {
24         h.clear();
25         return dfs(0, target, nums);
26     }
27 };

```

## 8. [LeetCode 473. 火柴拼正方形](#)

```

1  class Solution {
2  public:
3      bool dfs(int ind, vector<int> &arr, vector<int> &ms) {
4          if (ind == -1) return true;
5          for (int i = 0; i < 4; i++) {
6              if (arr[i] < ms[ind]) continue;
7              if (arr[i] == ms[ind] || arr[i] >= ms[ind] + ms[0]) {
8                  arr[i] -= ms[ind];
9                  if (dfs(ind - 1, arr, ms)) return true;
10                 arr[i] += ms[ind];
11             }
12         }
13         return false;
14     }
15     bool makesquare(vector<int>& matchsticks) {
16         sort(matchsticks.begin(), matchsticks.end());
17         vector<int> arr(4);
18         int sum = 0;
19         for (auto x : matchsticks) sum += x;
20         if (sum % 4) return false;
21         for (int i = 0; i < 4; i++) arr[i] = sum / 4;
22         return dfs(matchsticks.size() - 1, arr, matchsticks);
23     }
24 };

```

## 9. [LeetCode 39. 组合总和](#)

```

1  class Solution {
2  public:
3      void dfs(int ind, int target, vector<int> &nums,
4              vector<int> &buff, vector<vector<int>> &ret
5      ) {
6          if (target < 0) return ;
7          if (target == 0) {
8              ret.push_back(buff);
9              return ;
10         }
11         if (ind == nums.size()) return ;
12         dfs(ind + 1, target, nums, buff, ret);
13         buff.push_back(nums[ind]);
14         dfs(ind, target - nums[ind], nums, buff, ret);

```

```
15         buff.pop_back();
16         return ;
17     }
18     vector<vector<int>> combinationSum(vector<int>& candidates, int target) {
19         vector<int> buff;
20         vector<vector<int>> ret;
21         dfs(0, target, candidates, buff, ret);
22         return ret;
23     }
24 };
```

## 10. LeetCode 51. N 皇后

答案在彩蛋里，彩蛋作业敬请期待~



# 开课吧