20220223-田船长

841 钥匙和房间

```
class Solution {
public:
    bool canVisitAllRooms(vector<vector<int>>& rooms) {
        int n = rooms.size(), cnt = 1, mark[1005] = {0};
        queue<int> que;
        que.push(0);
        mark[0] = 1;
        while (!que.empty()) {
            int temp = que.front();
            que.pop();
            for (int i = 0; i < rooms[temp].size(); i++) {
                int e = rooms[temp][i];
                if (mark[e] == 0) {
                    mark[e] = 1;
                    cnt++;
                    que.push(e);
                }
            }
        }
        return cnt == n;
    }
};
```

994 腐烂的橘子

```
class Solution {
public:
    struct node {
        int x, y, step;
    };
    int n, m, cnt = 0, ans = 0;
    int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
    int orangesRotting(vector<vector<int>>& grid) {
        n = grid.size(), m = grid[0].size();
        queue<node> que;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                if (grid[i][j] == 1) {
                    cnt++;
                } else if (grid[i][j] == 2) {
                    que.push((node){i, j, 0});
                }
            }
        }
        while (!que.empty()) {
            node temp = que.front();
            ans = temp.step;
            que.pop();
            for (int i = 0; i < 4; i++) {
                int x = temp.x + dir[i][0];
                int y = temp.y + dir[i][1];
                if (x < 0 || y < 0 || x == n || y == m || grid[x][y] != 1)
continue;
                grid[x][y] = 2;
                cnt--;
                que.push((node){x, y, temp.step + 1});
            }
        }
        return cnt == 0 ? ans : -1;
    }
};
```

417 太平洋大西洋水流问题

```
class Solution {
public:
    int n, m, mark[205][205] = {0};
    int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
    void func(vector<vector<int> > &mmap, int x, int y, int val,
vector<vector<int> > &ans) {
        if ((mark[x][y] & val) != 0) return ;
        mark[x][y] |= val;
        if (mark[x][y] == 3) ans.push_back(vector<int>{x, y});
        for (int i = 0; i < 4; i++) {
            int xx = x + dir[i][0];
            int yy = y + dir[i][1];
            if (xx < 0 || yy < 0 || xx == n || yy == m) continue;
            if (mmap[xx][yy] >= mmap[x][y]) {
                func(mmap, xx, yy, val, ans);
            }
        }
    }
    vector<vector<int>> pacificAtlantic(vector<vector<int>>& heights) {
        vector<vector<int> > ans;
        n = heights.size(), m = heights[0].size();
        for (int i = 0; i < n; i++) {
            func(heights, i, 0, 1, ans);
            func(heights, i, m - 1, 2, ans);
        }
        for (int i = 0; i < m; i++) {
            func(heights, 0, i, 1, ans);
            func(heights, n - 1, i, 2, ans);
        }
        return ans;
    }
};
```

529 扫雷游戏

```
class Solution {
public:
    struct node {
        int x, y;
    };
    int n, m;
    int dir[8][2] = {0, 1, 1, 0, 0, -1, -1, 0, 1, 1, 1, -1, -1, 1, -1, -1};
    int func(vector<vector<char> > &mmap, node &temp) {
        int cnt = 0;
        for (int i = 0; i < 8; i++) {
            int x = temp.x + dir[i][0];
            int y = temp.y + dir[i][1];
            if (x < 0 || y < 0 || x == n || y == m) continue;
            cnt += mmap[x][y] == 'M';
        }
        return cnt;
    }
    vector<vector<char>> updateBoard(vector<vector<char>>& board, vector<int>&
click) {
        if (board[click[0]][click[1]] == 'M') {
            board[click[0]][click[1]] = 'X';
            return board;
        }
        n = board.size(), m = board[0].size();
        queue<node> que;
        que.push((node){click[0], click[1]});
        board[click[0]][click[1]] = 'B';
        while (!que.empty()) {
            node temp = que.front();
            que.pop();
            int cnt = func(board, temp);
            if (cnt != 0) {
                board[temp.x][temp.y] = '0' + cnt;
                continue;
            }
            for (int i = 0; i < 8; i++) {
                int x = temp.x + dir[i][0];
                int y = temp.y + dir[i][1];
                if (x < 0 || y < 0 || x == n || y == m || board[x][y] != 'E')
continue;
                board[x][y] = 'B';
                que.push((node){x, y});
            }
        }
        return board;
    }
};
```

127 单词接龙

```
class Solution {
public:
    struct node {
        string str;
        int step;
    };
    int ladderLength(string beginWord, string endWord, vector<string>&
wordList) {
        unordered_set<string> s;
        for (auto &str : wordList) {
            s.insert(str);
        }
        if (s.count(endWord) == 0) {
            return 0;
        }
        queue<node> que;
        que.push((node){beginWord, 1});
        while (!que.empty()) {
            node temp = que.front();
            que.pop();
            if (temp.str == endWord) {
                return temp.step;
            }
            for (int i = 0; i < temp.str.size(); i++) {
                string t = temp.str;
                for (int j = 'a'; j <= 'z'; j++) {
                    t[i] = j;
                    if (s.count(t) == 1) {
                        s.erase(t);
                        que.push((node){t, temp.step + 1});
                    }
                }
            }
        }
        return 0;
    }
};
```

1631 最小体力消耗路径

```
class Solution {
public:
    struct node {
        int x, y;
    };
    int n, m;
    int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
    int func(int dis, vector<vector<int> > &mmap) {
        int mark[105][105] = {0};
        queue<node> que;
        que.push((node){n - 1, m - 1});
        mark[n - 1][m - 1] = 1;
        while (!que.empty()) {
            node temp = que.front();
            que.pop();
            if (temp.x == 0 && temp.y == 0) {
                return 1;
            }
            for (int i = 0; i < 4; i++) {
                int x = temp.x + dir[i][0];
                int y = temp.y + dir[i][1];
                if (x < 0 || y < 0 || x == n || y == m || mark[x][y] == 1)
continue;
                if (abs(mmap[temp.x][temp.y] - mmap[x][y]) <= dis) {
                    mark[x][y] = 1;
                    que.push((node){x, y});
                }
            }
        }
        return 0;
    }
    int bs(vector<vector<int> > &mmap) {
        int l = 0, r = 1000000;
        while (l != r) {
            int mid = (l + r) / 2;
            if (func(mid, mmap)) {
                r = mid;
            } else {
                l = mid + 1;
            }
        }
        return l;
    }
    int minimumEffortPath(vector<vector<int>>& heights) {
        n = heights.size(), m = heights[0].size();
        return bs(heights);
    }
};
```

864 获取所有钥匙的最短路径

```
class Solution {
public:
    struct node {
        int x, y, step, status;
    };
    int n, m, cnt, mark[35][35][256] = {0};
    int dir[4][2] = {0, 1, 1, 0, 0, -1, -1, 0};
    int b2[10] = {1, 2, 4, 8, 16, 32, 64, 128, 256};
    int shortestPathAllKeys(vector<string>& mmap) {
        n = mmap.size(), m = mmap[0].size();
        queue<node> que;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                if (mmap[i][j] == '@') {
                    mmap[i][j] = '.';
                    que.push((node){i, j, 0, 0});
                    mark[i][j][0] = 1;
                } else if (mmap[i][j] >= 'a' && mmap[i][j] <= 'z') {
                    cnt++;
                }
            }
        }
        while (!que.empty()) {
            node temp = que.front();
            que.pop();
            if (temp.status == b2[cnt] - 1) {
                return temp.step;
            }
            for (int i = 0; i < 4; i++) {
                int x = temp.x + dir[i][0];
                int y = temp.y + dir[i][1];
                if (x < 0 || y < 0 || x == n || y == m || mark[x][y]
[temp.status]) continue;
                if (mmap[x][y] == '.') {
                    mark[x][y][temp.status] = 1;
                    que.push((node){x, y, temp.step + 1, temp.status});
                } else if (mmap[x][y] >= 'a' && mmap[x][y] <= 'z') {
                    mark[x][y][temp.status] = 1;
                    mark[x][y][temp.status | b2[mmap[x][y] - 'a']] = 1;
                    que.push((node){x, y, temp.step + 1, temp.status |
b2[mmap[x][y] - 'a']});
                } else if (mmap[x][y] >= 'A' && mmap[x][y] <= 'Z' &&
(temp.status & b2[mmap[x][y] - 'A'])) {
                    mark[x][y][temp.status] = 1;
                    que.push((node){x, y, temp.step + 1, temp.status});
                }
            }
        }
        return -1;
    }
};
```