

3.6 课堂笔记 2

课后作业：（直接发群里）

1. 最短路径（熟练使用广搜算法模板）

一个迷宫由R行C列格子组成，有的格子里有障碍物，不能走；有的格子是空地，可以走。

给定一个迷宫，求从左上角走到右下角最少需要走多少步(数据保证一定能走到)。只能在水平方向或垂直方向走，不能斜着走。

输入

第一行是两个整数，R和C，代表迷宫的长和宽。（ $1 \leq R, C \leq 40$ ）

接下来是R行，每行C个字符，代表整个迷宫。空地格子用 '.' 表示，有障碍物的格子用 '#' 表示。迷宫左上角和右下角都是 '.'。

输出

输出从左上角走到右下角至少要经过多少步（即至少要经过多少个空地格子）。计算步数要包括起点和终点

样例输入

```
5 5
..###
.....
.....
..###
.....
```

样例输出

9

2. 最少转弯次数：洛谷 P1649

3. 细胞个数：洛谷 P1451

广搜知识点梳理：

1、思路：

从一个节点出发，先访问其直接相连的所有子节点，再访问其子节点相连的子节点，按层次顺序访问，直到访问到目标节点。

2、深搜和广搜的区别：

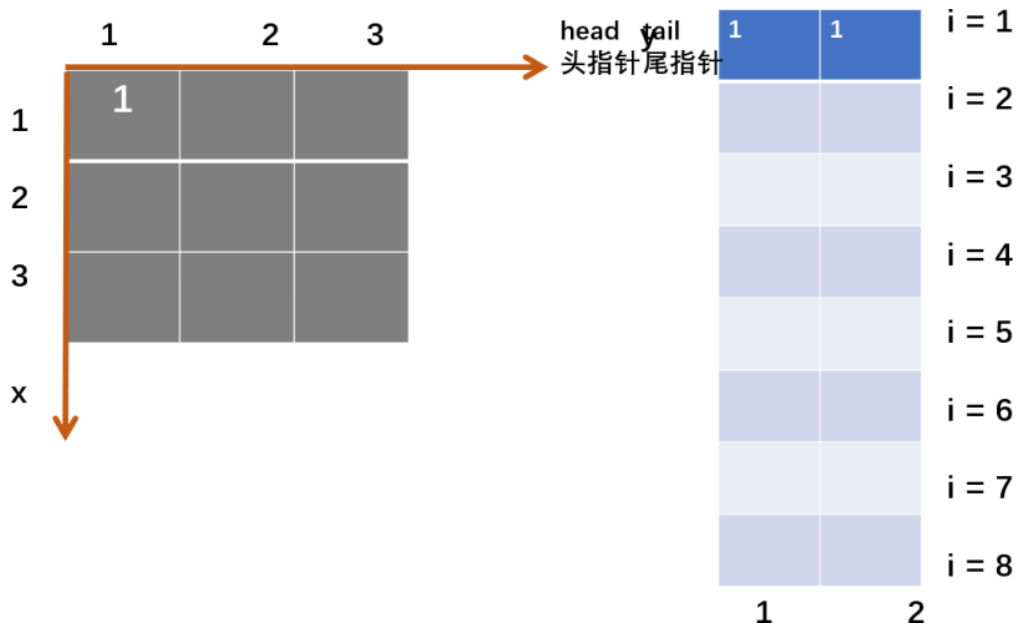
BFS 关注：解决最短或最少问题

DFS 关注：解决路径能否到达或所有路径有效

3、队列

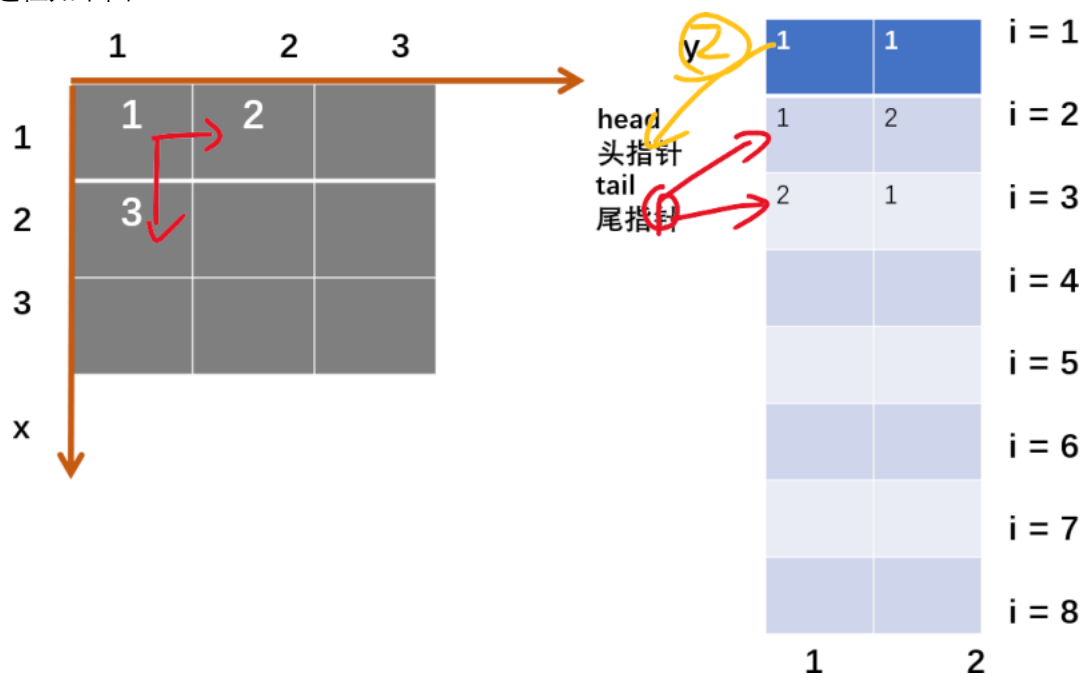
队列的规则：先进先出。同广搜相结合是记录搜索的次序。初始化：

head 和 tail 均指向 1，并将 que 的第一个点初始化为 1,1（坐标）



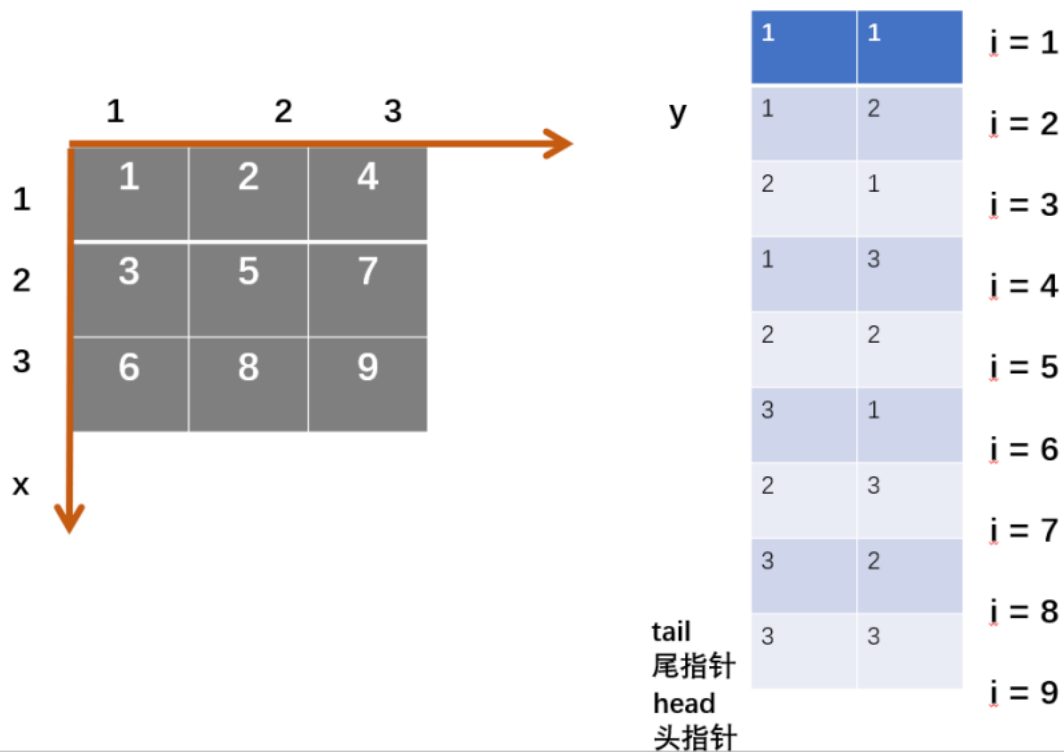
第一步：将头指针指向的位置 (1,1) 按照方向进行结点遍历，将结点存入到 que 中
 向右走：tail++，(1,2) 存入 que;
 向下走：tail++，(2,1) 存入 que;
 向左走：出界
 向上走：出界

第二步：将 (1,1) 踢出队列，即 head++
 过程如下图：



...

(重复出现这个步骤，直到如下图的状态) 直到 head > tail 的时候，结束查找



实现参考代码如下：

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 int n,m;
4 int mapp[110][110];
5 int que[10100][3]; //数组实现
6 int head,tail;
7 int cx[4]={0,1,0,-1};
8 int cy[4]={1,0,-1,0};
9
10 void bfs(){
11     int x,y,curX,curY;
12     head = 1; //初始化
13     tail = 1;
14     que[1][1] = 1;
15     que[1][2] = 1;
16     mapp[1][1] = 1;
17     while(tail >= head){
18         x = que[head][1]; //取当前队头
19         y = que[head][2];
20         for(int i = 0; i < 4; i++){ //对当前队头进行广搜
21             curX = x + cx[i];
22             curY = y + cy[i];
23             if(curX >= 1 && curX <= n && curY >= 1 && curY <= m && mapp[curX][curY] == 0){ //走过的不能再走
24                 tail++; //队尾往后移
25                 que[tail][1] = curX; //添加元素
26                 que[tail][2] = curY;
27                 mapp[curX][curY] = tail; //保存顺序
28             }
29         }
30         head++; //队头往后移
31     }
32 }
33
34
35 int main(){
36     cin >> n >> m;
37     bfs();
38     for(int i = 1; i <= n; i++){
39         for(int j = 1; j <= m; j++){
40             cout << mapp[i][j] << " ";
41         }
42         cout << endl;
43     }
44     return 0;
45 }

```

4、求解最短步数数组

	1	2	3
1	1	2	3
2	2	3	4
3	3	4	5
X			

上面的数组记录的是从左上到每个位置的最短步数。实现上面数组的参考代码如下：

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int n,m;
4  int mapp[110][110];
5  int que[10100][3];
6  int head,tail;
7  int cx[4]={0,1,0,-1};
8  int cy[4]={1,0,-1,0};
9
10 void bfs(){
11     int x,y,curX,curY;
12     head = 1; //初始化
13     tail = 1;
14     que[1][1] = 1;
15     que[1][2] = 1;
16     mapp[1][1] = 1;
17     while(tail >= head){
18         x = que[head][1]; //取当前队头
19         y = que[head][2];
20         for(int i = 0; i <= 3; i++){ //对当前队头进行广搜
21             curX = x + cx[i];
22             curY = y + cy[i];
23             if(curX >= 1 && curX <= n && curY >= 1 && curY <= m && mapp[curX][curY]==0){ //走过的不能再走
24                 tail++; //队尾往后移
25                 que[tail][1] = curX; //添加元素
26                 que[tail][2] = curY;
27                 mapp[curX][curY] = mapp[x][y] + 1; //最短步数
28             }
29         }
30         head++; //队头往后移
31     }
32 }
33
34
35 int main(){
36     cin >> n >> m;
37     bfs();
38     for(int i = 1; i <= n; i++){
39         for(int j = 1; j <= m; j++){
40             cout << mapp[i][j] << " ";
41         }
42         cout << endl;
43     }
44     return 0;
45 }

```

5、使用模板库的队列，
队列的具体操作方法如下：

```

// 模板库队列常见操作
queue<lst> que; // 定义一个空队列
que.push(val); // 队尾进队
que.pop(); // 队头出队
que.empty(); // 判断是否是空队列
que.front(); // 取队头元素

```

替换上面的代码:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int n,m;
4  int mapp[110][110];
5  struct lst{
6      int x;
7      int y;
8  };
9  queue<lst> que;
10 int cx[4]={0,1,0,-1};
11 int cy[4]={1,0,-1,0};
12
13 void bfs(){
14     int curX,curY;
15     mapp[1][1] = 1;
16     que.push((lst){1,1}); //初始化,插入队头
17     while(!que.empty()){
18         lst head = que.front(); //取队头
19         for(int i=0; i<=3; i++){
20             curX = head.x + cx[i];
21             curY = head.y + cy[i];
22             if(curX >= 1 && curX <= n && curY >= 1 && curY <= m && mapp[curX][curY]==0){
23                 que.push((lst){curX,curY}); //队尾插入元素
24                 mapp[curX][curY] = mapp[head.x][head.y] + 1;
25             }
26         }
27         que.pop(); //去队头
28     }
29 }
30
31 int main(){
32     cin >> n >> m;
33     bfs();
34     for(int i=1; i<=n; i++){
35         for(int j=1; j<=m; j++){
36             cout << mapp[i][j] << " ";
37         }
38         cout << endl;
39     }
40     return 0;
41 }
```

练习 1: 洛谷 P1747

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int n,m;
4  int x1,yy1,x2,y2;
5  int mapp[110][110];
6  struct lst{
7      int x;
8      int y;
9  };
10 queue<lst> que;
11 int cx[12]={1,1,2,2,2,2,-1,-1,-2,-2,-2,-2}; //12个方向
12 int cy[12]={-2,-2,-2,-1,1,2,-2,-2,-1,1,-2,2};
13
14 void bfs(int x, int y){
15     int curX,curY;
16     que.push((lst){x,y}); //初始化
17     while(!que.empty()){
18         lst head = que.front(); //取队头
19         for(int i=0; i<=11; i++){
20             curX = head.x + cx[i];
21             curY = head.y + cy[i];
22             if(curX >= 1 && curX <= 50 && curY >= 1 && curY <= 50 && mapp[curX][curY]==0){
23                 que.push((lst){curX,curY}); //队尾插入元素
24                 mapp[curX][curY] = mapp[head.x][head.y] + 1; //最短步数
25                 if(curX == 1 && curY == 1){
26                     return;
27                 }
28             }
29         }
30         que.pop(); //去队头
31     }
32 }
33
34 int main(){
35     cin >> x1 >> yy1 >> x2 >> y2;
36     bfs(x1,yy1);
37     cout << mapp[1][1] << endl;
38     memset(mapp,0,sizeof(mapp)); //清空数组
39     while(!que.empty()) que.pop(); //清空队列
40     bfs(x2,y2);
41     cout << mapp[1][1] << endl;
42     return 0;
43 }
```

练习 2：洛谷 P6207

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int n,m;
4  char mapp[120][120];
5  int que[10100][3];
6  int head,tail;
7  int cx[4]={0,1,0,-1};
8  int cy[4]={1,0,-1,0};
9
10 void print(int x){ //输出路径
11     if(x == 1){
12         cout << que[1][1] << " " << que[1][2] << endl;
13     }
14     else{
15         print(que[x][0]);
16         cout << que[x][1] << " " << que[x][2] << endl;
17     }
18 }
19 void bfs(){
20     int x,y,curX,curY;
21     head = 1; //初始化
22     tail = 1;
23     que[1][0] = 0;
24     que[1][1] = 1;
25     que[1][2] = 1;
26     mapp[1][1] = '*';
27     while(tail >= head){
28         x = que[head][1]; //取当前队头
29         y = que[head][2];
30         for(int i = 0; i <= 3; i++){ //对当前队头进行广搜
31             curX = x + cx[i];
32             curY = y + cy[i];
33             if(curX >= 1 && curX <= n && curY >= 1 && curY <= m && mapp[curX][curY] != '.'){ //走过的不能再走
34                 mapp[curX][curY] = '*';
35                 tail++; //队尾往后移
36                 que[tail][0] = head; //保存上一个结点
37                 que[tail][1] = curX; //添加元素
38                 que[tail][2] = curY;
39                 if(curX == n && curY == m){
40                     return;
41                 }
42             }
43         }
44         head++; //队头往后移
45     }
46 }
47
48 int main(){
49     cin >> n >> m;
50     for(int i = 1; i <= n; i++){
51         for(int j = 1; j <= m; j++){
52             cin >> mapp[i][j];
53         }
54     }
55     bfs();
56     print(tail);
57     return 0;
58 }

```

练习 3：洛谷 P2895

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn = 310;
4  struct lst{
5      int x;
6      int y;
7      int step;
8  };
9  int cx[4]={0,1,0,-1};
10 int cy[4]={1,0,-1,0};
11 int mapp[maxn][maxn];
12 bool vis[maxn][maxn];
13 int m;
14 queue<lst> que;
15
16 int bfs(){
17     que.push((lst){0,0,0});
18     vis[0][0] = 1;
19     while(!que.empty()){
20         lst head = que.front();
21         for(int i=0;i<4;i++){
22             int curX = head.x + cx[i];
23             int curY = head.y + cy[i];
24             if(!(curX>=0 && curY>=0)) { //超过范围, 直接返回
25                 continue;
26             }
27             if(mapp[curX][curY] == 0x3f3f3f3f){

```

```

28         return head.step + 1; //如果这个点为无穷大,表示永远不会被摧毁,已经安全了
29     }
30
31     if (head.step + 1 < mapp[curX][curY] && !vis[curX][curY]) { //如果到达这个时间点的时间早于最早摧毁时间, 可以走
32         vis[curX][curY] = 1;
33         que.push(lst){curX, curY, head.step + 1};
34     }
35 }
36 que.pop();
37 }
38 return -1;
39 }
40
41 int main(){
42     //初始化为无穷大
43     for(int i=0; i<maxn; i++){ //maxn=310
44         for(int j=0; j<maxn; j++){
45             mapp[i][j] = 0x3f3f3f3f; //无穷大
46         }
47     }
48     cin >> m;
49     for(int i=0; i<m; i++){
50         int x, y, t;
51         cin >> x >> y >> t;
52         //求出该点和周围点的最早摧毁时间
53         mapp[x][y] = min(mapp[x][y], t); //该点
54         for(int j=0; j<4; j++){
55
56             int xx = x + cx[j];
57             int yy = y + cy[j];
58             if(xx>=0 && yy>=0){
59                 mapp[xx][yy] = min(mapp[xx][yy], t); //周围点
60             }
61         }
62     }
63     cout << bfs();
64     return 0;
65 }
66

```