第 9 章 广度优先搜索（BFS）

9.1 什么是广度优先搜索（BFS）

广度优先搜索算法(Breadth First Search)，又称为"宽度优先搜索"或"横向优先搜索"，简称 BFS。

其思路为从图上一个节点出发，先访问其直接相连的子节点，若子节点不符合，再问其子节点的子节点，按级别顺序依次访问，直到访问到目标节点。

BFS 可用于解决 2 类问题：

- ✓ 从 A 出发是否存在到达 B 的路径；
- ✓ 从 A 出发到达 B 的最短路径(这个应该叫最少步骤合理)；

深搜和广搜的区别：

遍历分为：

1.深度（Depth）优先搜索 DFS：一个递归过程，有回退过程。尽可能“深”地搜索图。在深度优先搜索中，对于最新发现的顶点，如果它还有以此为起点而未探测到的边，就沿此边继续搜索下去。当结点 V 的所有边都已被探寻过，搜索将回溯到发现结点 V 有那条边的始结点，则选择其中一个作为源结点并重复以上过程，整个进程反复进行直到所有结点都被发现为止。

2.广度（Breath）优先搜索 BFS：一个分层的搜索过程，没有回退过程，是非递归的。只是每次都尽可能地扩展当前节点的邻居节点，之后再向其子结点进行扩展。

【例子】从一个 n * m 的迷宫的左上角走到右下角，其中“.”表示该点能走通，“#”表示该点不能走通。

要求按照右、下、左、上的方向遍历，请问：

① 从左上角走到右下角至少需要多少步？

② 从左上角走到右下角的最短路径是哪一条路径？

请分别思考，在深搜和广搜下分别如何实现，有什么区别？

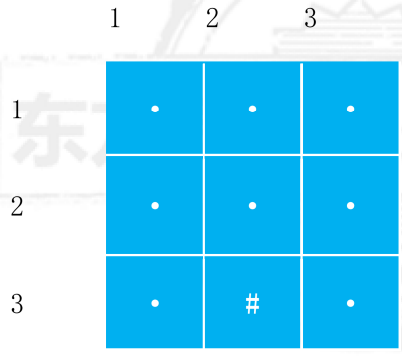


图 1

1 2 3 4

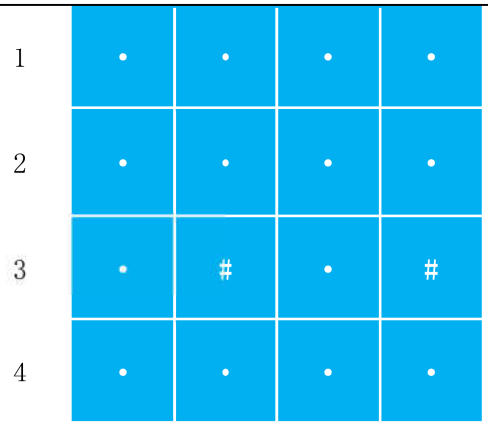


图 2

应用上的区别：

✓ BFS：对于 解决最短或最少问题特别有效（因为 BFS 只要访问到某个点，一定是最短路径到达的点），而且寻找深度小，但缺点是内存耗费量大（需要开大量的数组单元用来存储状态）。

✓ DFS：对于 解决遍历和求所有问题有效 ，对于问题搜索深度小的时候处理速度迅速，然而在深度很大的情况下效率不高。

实现时用到的数据结构：

广度优先搜索使用队列（queue）来实现，整个过程可以看做一个倒立的树形：

1、把根节点放到队列的末尾。

2、每次从队列的头部取出一个元素，查看这个元素所有的下一级元素，把它们放到队列的末尾。并把这个元素记为它下一级元素的前驱。

3、找到所要找的元素时结束程序。

4、如果遍历整个树还没有找到，结束程序。

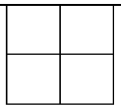
9.2 广搜的应用

1751：【入门】快乐的马里奥

思路：用一个二维数组（队列）存储广搜遍历的每个点，将 head 四个方向的可访问的点（没有出矩阵，且点没有访问过）遍历标记并将这些点存入队列，一个点四个方向都尝试访问结束后，head++，尝试新的点，直到 head>tail，表示对列中每个点四个方向的点都标记结束了。

	j=1	j=2	j=3
i=1	1	2	
i=2	3		
i=3			

	x	y		头指针	尾指针
q	1	1	i=1		
	1	2	i=2	head	
	2	1	i=3		tail
			i=4		
			i=5		
			i=6		
			i=7		




```
#include <bits/stdc++.h>
using namespace std;
//a:存储矩阵,q:存储队列（遍历过的点）
int a[110][110],q[10100][3];

//探测每个点的四个方向
int fx[5] = {0,0,1,0,-1};
int fy[5] = {0,1,0,-1,0};
int tx,ty;//表示尝试访问的点
int k = 1;//每个点标记的值，每标记一次，自增 1
int n,m,i,j;
//head:头指针，指向头的位置,tail:尾指针，指向尾的位置
int head=1,tail=1;

int main(){
    cin>>n>>m;
    //起点
    a[1][1] = k;
    k++;
    //1,1 点存入队列，先以 1,1 点为 head，访问其四个方向
    q[1][1] = 1;
    q[1][2] = 1;

    //当 head<=tail 时，说明队列还有点没有尝试标记过四个方向
    while(head <= tail){
        //尝试 head 对应的点的四个方向
        for(i = 1;i <= 4;i++){
            //从 head 对应的点开始，加上方向值的变化，得到四个新点
            tx = q[head][1] + fx[i];
            ty = q[head][2] + fy[i];
            //如果新点在矩阵内，且新点没有访问过
            if(tx>=1&&tx<=n&&ty>=1&&ty<=m&&a[tx][ty]==0){
                //讲新点存入队列
                tail++;
                q[tail][1] = tx;
                q[tail][2] = ty;
                a[tx][ty] = k;//标记访问的点对应的值
                k++;
            }
        }

        //head++，使得刚刚标记过四个方向的点出队列，取下一个点
        head++;
    }

    //输出矩阵
    for(i = 1;i <= n;i++){
        for(j = 1;j <= m;j++){
            cout<<a[i][j]<<" ";
        }
        cout<<endl;
    }

    return 0;
}
```

1443：【提高】泉水

j=1 j=2 j=3 j=4 j=5 x y 头指针 尾指针

i=1	3	4	1	5	1
i=2	2	3	3	4	7
i=3	4	1	4	1	1

泉眼：2, 3

	j=1	j=2	j=3	j=4	j=5
i=1	F	F	T	F	F
i=2	T	T	T	F	F
i=3	F	T	F	F	F

q

2	3	i=1
2	2	i=2
1	3	i=3
3	2	i=4
2	1	i=5
1	1	i=6 tail
		i=7 head
		i=8
		i=9
		i=10
		i=11
		i=12
		i=13
		i=14
		i=15

```
#include <iostream>
using namespace std;
int n,m,p1,p2;
int a[1010][1010];
bool f[1010][1010];//标记某个点是否走过
int que[1000100][3];
int fx[] = {0,1,0,-1};//方向数组
int fy[] = {1,0,-1,0};

int main(){
    int i,j,head = 1,tail = 1;
    int x,y;
    cin>>n>>m>>p1>>p2;
    for(i = 1;i <= n;i++){
        for(j = 1;j <= m;j++){
            cin>>a[i][j];
        }
    }

    que[head][1] = p1;
    que[head][2] = p2;
    f[p1][p2] = true;//标记走过了
    while(head <= tail){
        for(i = 0;i < 4;i++){
            x = que[head][1] + fx[i];
            y = que[head][2] + fy[i];
            //判断边界
            if(x <= n && x >= 1 && y <= m && y >= 1 && a[x][y] <= a[p1][p2] &&
f[x][y] == false){
                tail++;
                que[tail][1] = x;
                que[tail][2] = y;
                f[x][y] = true;//标记走过了
            }
        }

        head++;//统计下一个点的可行路径
    }
```

```
//   for(i = 1;i <= tail;i++){
//       cout<<i<<":"<<que[i][1]<<","<<que[i][2]<<" ";
//   }
//   cout<<endl;

    cout<<tail<<endl;
}
```

1432：【基础】走出迷宫的最少步数

思路：记录走到每个点需要的步数，由于 BFS 首次访问到的点一定就是最短路径，因此，只要走到终点就打印步数即可。

	j=1	j=2	j=3	j=4
i=1	.	.	.	.
i=2	.	.	.	.
i=3	.	#	.	#
i=4	.	.	.	.

q

x	y	step	
1	1	1	i=1
1	2	2	i=2
2	2	2	i=3
			i=4
			i=5
			i=6
			i=7
			i=8

```
#include <bits/stdc++.h>
using namespace std;

int n,m,i,j;
char a[50][50];//迷宫
int q[2500][4];//队列
int fx[5] = {0,0,1,0,-1};//方向数组
int fy[5] = {0,1,0,-1,0};
int tx,ty;//表示从 head 开始探索的新方向
int head = 1,tail = 1;

int main(){
    cin>>n>>m;
    for(i = 1;i <= n;i++){
        for(j = 1;j <= m;j++){
            cin>>a[i][j];
        }
    }

    //1,1 点直接存入队列，从 1,1 点开始探索
    q[1][1] = 1;
    q[1][2] = 1;
    q[1][3] = 1;//步数

    while(head <= tail){
        //从 head 开始探索四个方向
        for(i = 1;i <= 4;i++){
            tx = q[head][1] + fx[i];
            ty = q[head][2] + fy[i];

            //如果该点可行（在迷宫内，没有探索过）
            if(a[tx][ty] == '.'){
                //标记 tx，ty 走过了
                a[tx][ty] = '#';
                //将 tx，ty 存入队列
```

```
                tail++;
                q[tail][1] = tx;
                q[tail][2] = ty;
                //更新步数
                q[tail][3] = q[head][3] + 1;

                //如果到了终点
                if(tx == n && ty == m){
                    cout<<q[tail][3];
                    return 0;
                }
            }
        }

        //四个方向探索完毕，head++，探索下一个点的四个方向
        head++;
    }

    return 0;
}
```

注意：本题不能最后以 q[tail][3]作为走到右下角的最少步数，因为右下角不一定是最后一个进入队列的点。

比如：如下图所示的迷宫，最后一个进入队列的点就是 1,3 点。

	j=1	j=2	j=3
i=1	.	#	.
i=2	.	#	.
i=3	.	.	.

1442：【提高】走出迷宫的最短路径

思路：记录遍历过程中的每个点，以及每个点的来源下标，到终点时递归寻找每个点的来源并打印下标。

	j=1	j=2	j=3	j=4
i=1	0	0	0	0
i=2	0	1	1	0
i=3	0	1	0	1
i=4	0	0	0	0

q

	x	y	pre	
	1	1	0	i=1
	1	2	1	i=2
	2	1	1	i=3
	1	3	2	i=4
	3	1	3	i=5
	1	4	4	i=6
	4	1	5	i=7
	2	4	6	i=8
	4	2	7	i=9
	4	3	9	i=10
	4	4	10	i=11

```
#include <bits/stdc++.h>
using namespace std;

/*
```

```
  思路：从出发点开始广搜，记录每个点及每个点的前驱点（父节点）
*/
int n,m,i,j;
int a[150][150];//迷宫
int q[40000][4];//队列
int fx[5] = {0,0,1,0,-1};//方向数组
int fy[5] = {0,1,0,-1,0};
int tx,ty;//表示从 head 开始探索的新方向
int head = 1,tail = 1;
int s1,s2,e1,e2;//起止点

//打印路径
void print(int k){
    //如果前驱节点不为 0（有前驱节点，则递归找前驱节点）
    if(q[k][3] != 0){
        print(q[k][3]);
    }

    cout<<"("<<q[k][1]<<","<<q[k][2]<<")";
    if(k != tail){
        cout<<"->";
    }
}

int main(){
    cin>>n>>m;
    for(i = 1;i <= n;i++){
        for(j = 1;j <= m;j++){
            cin>>a[i][j];
        }
    }

    //读入起止点
    cin>>s1>>s2>>e1>>e2;

    //出发点直接存入队列
    q[1][1] = s1;
    q[1][2] = s2;
    //出发点没有前驱点（父节点）
    q[1][3] = 0;

    while(head <= tail){
        //遍历 head 的四个方向
        for(i = 1;i <= 4;i++){
            //得到 head 节点的四个方向的坐标
            tx = q[head][1] + fx[i];
            ty = q[head][2] + fy[i];

            //如果该方向可达（在迷宫内，且没走过）
            if(tx>=1&&tx<=n&&ty>=1&&ty<=m&&a[tx][ty]==0){
                //走过的点做标记
                a[tx][ty] = 1;
                //将点存入队列
                tail++;
                q[tail][1] = tx;
                q[tail][2] = ty;
                //记录该点的前驱点的行下标
                q[tail][3] = head;

                //如果到了终点，打印
                if(tx == e1 && ty == e2){
                    print(tail);
                    return 0;
                }
            }
        }

        head++;
    }
```

```
    cout<<"no way";
    return 0;
}
```

1441：【提高】骑士牛

```
#include<bits/stdc++.h>
using namespace std;

/*
  做题最重要的一个注意事项：仔细审题；
*/
int n,m,i,j;
char a[200][200];//迷宫
int q[40000][4];//队列
//方向数组
int fx[9] = {0,-1,-2,-2,-1,1,2,2,1};
int fy[9] = {0,-2,-1,1,2,2,1,-1,-2};
int tx,ty;//表示从 head 开始探索的新方向
int head = 1,tail = 1;
//起止点坐标
int s1,s2,e1,e2;

int main(){
    //n 行 m 列
    cin>>m>>n;
    for(i = 1;i <= n;i++){
     for(j = 1;j <= m;j++){
         cin>>a[i][j];
         if(a[i][j] == 'K'){
             s1 = i;
             s2 = j;
             }else if(a[i][j] == 'H'){
                 e1 = i;
                 e2 = j;
             }
         }
     }

     //出发点存入队列
     q[1][1] = s1;
     q[1][2] = s2;
     //记录出发点的步数
     q[1][3] = 0;

     while(head <= tail){
         //遍历 8 个方向
         for(i = 1;i <= 8;i++){
             //获得 head 周边的 8 个坐标
             tx = q[head][1] + fx[i];
             ty = q[head][2] + fy[i];

             //如果新点在迷宫内，且没走过
             if(a[tx][ty] == '.' || a[tx][ty] == 'H'){
                 //标记该点走过
                 a[tx][ty] = '*';
                 //将该点存入队列
                 tail++;
                 q[tail][1] = tx;
                 q[tail][2] = ty;
                 //记录步数
                 q[tail][3] = q[head][3] + 1;

                 //如果到了终点
                 if(tx == e1 && ty == e2){
                     cout<<q[tail][3];
                     return 0;
```

```
                }
            }
        }

        head++;
    }
}
```

9.3 广搜作业

1430：【基础】迷宫出口

1434：【基础】数池塘（四方向）

1433：【基础】走出迷宫的最少步数 2

1541：【提高】小 X 学游泳(swim)

1380：【提高】小 X 学游泳

1381：【提高】方格取数

1444：【提高】最小拐弯路径