```cpp
//日期: 2018/ 时间:
#include <stdio.h>
#include <stdlib.h>
#include <queue>
using namespace std;
//#define p1
#define p2

//**********************************************************************
***
#ifdef p2
const int maxn = 100;
struct node{
    int x,y;
    int step;
}S,T,Node;                  //S为起点，T为中点，Node为临时节点

int n,m;
char maze[maxn][maxn];      //迷宫信息
bool inq[maxn][maxn] = {false}; //记录位置(x,y)是否已入过队
int X[4] = {0,0,1,-1};
int Y[4] = {1,-1,0,0};

//检测位置(x,y)是否有效
bool test(int x,int y){
    if(x>=n || x<0 || y>=m || y<0)  return false;
    if(maze[x][y]=='*') return false;
    if(inq[x][y] == true)   return false;   //已经入过队
    return true;
}

int BFS(){
    queue<node> q;
    q.push(S);          //起点
    while(!q.empty()){
        node top = q.front();       //取出队首元素
        q.pop();
        if(top.x == T.x && top.y == T.y){
            return top.step;        //终点，直接返回最少步数
        }
        for(int i=0;i<4;i++){
            int newx = top.x + X[i];
            int newy = top.y + Y[i];
            if(test(newx,newy)){    //位置()有效
                Node.x = newx,Node.y = newy;
                Node.step = top.step + 1;   //Node层数为top的层数加一
                q.push(Node);
                inq[newx][newy] = true;
            }
        }
    }

    return -1;  //无法到达
}

```

```
55  int main(){
56      scanf("%d%d",&n,&m);
57      for(int i=0;i<n;i++){
58          getchar();          //过滤上一行的换行符
59          for(int j=0;j<m;j++){
60              maze[i][j] = getchar();
61          }
62          maze[i][m+1]='\0';  //补上结束符
63      }
64      scanf("%d%d%d%d",&S.x,&S.y,&T.x,&T.y);      //起点和终点坐标
65      S.step=0;
66      printf("%d\n",BFS());
67
68
69      return 0;
70  }
71  /*
72  测试数据:
73  5 5
74  .....
75  .*.*.
76  .*S*.
77  .***.
78  ...T*
79  2 2 4 3
80  */
81
82
83  #endif
84  //  ********************************************************************************
    ***
85  #ifdef p1
86  const int maxn = 100;
87  struct node{
88      int x,y;    //位置(x,y)
89  }Node;
90
91  int n,m;        //矩阵大小为n*m
92  int matrix[maxn][maxn];
93  bool inq[maxn][maxn] = {false};     //记录位置(x,y)是否已入队列
94  int X[4] = {0,0,1,-1};
95  int Y[4] = {1,-1,0,0};
96
97  bool judge(int x,int y){            //判断坐标(x,y)是否需要访问
98      //越界
99      if(x>=n || x<0 || y>m || y<0) return false;
100     //当前位置为0，或(x,y)已入过队列，返回false
101     if(matrix[x][y] == 0 || inq[x][y] == true) return false;
102
103     //以上都不满足，返回true
104     return true;
105 }
106
107 //BFS函数访问位置(x,y)所在块，将该块中所有'1'的inq都设置为true
108 void BFS(int x,int y){
```

```
109     queue<node> q;
110     Node.x = x,Node.y = y;      //当前节点坐标
111     q.push(Node);       //将节点Node入队
112     inq[x][y] = true;
113     while(!q.empty()){
114         node top = q.front();       //取出队首元素
115         q.pop();                //队首元素出队
116         for(int i=0;i<4;i++){
117             int newx = top.x + X[i];
118             int newy = top.y + Y[i];
119             if(judge(newx,newy)){   //如果新位置(newX,newY)需要访问
120                 //设置Node坐标为(newX,newY)
121                 Node.x = newx,Node.y = newy;
122                 q.push(Node);       //将节点Node加入队列
123                 inq[newx][newy] = true; //设置位置(newx,newy)已入队列
124             }
125         }
126     }
127 }
128
129 int main(){
130     scanf("%d%d",&n,&m);
131     for(int x=0;x<n;x++){
132         for(int y=0;y<m;y++){
133             scanf("%d",&matrix[x][y]);
134         }
135     }
136     int ans = 0;        //存放块数
137
138     for(int x=0;x<n;x++){       //枚举每一个位置
139         for(int y=0;y<m;y++){
140             //如果元素为1，且未入过队
141              if(matrix[x][y] == 1 && inq[x][y] == false){
142                 ans++;      //块数加一
143                 BFS(x,y);   //访问整个块，将该块所有"1"的inq都标记为true
144              }
145         }
146     }
147     printf("%d\n",ans);
148
149     return 0;
150 }
151 #endif
152
```