宽度优先搜索bfs

还有一点点队列(stl)

题目描述p1443

> 有一个n*m的棋盘(1<n,m<=400)，在某个点上有一个马,要求你计算出马到达棋盘上任意一个点最少要走几步
>
> 输入格式
>
> 一行四个数据，棋盘的大小和马的坐标
>
> 输出格式
>
> 一个n*m的矩阵，代表马到达某个点最少要走几步（左对齐，宽5格，不能到达则输出-1）

输入	输出
3 3 1 1	0 3 2 3 -1 1 2 1 4

tip：队列

```
//先进先出
#include <queue>
queue<int> q;
//也可以是char string float 结构体.....
q.push();q.pop();q.size();q.front();
q.empty(); //空为true

//优先队列
#include <queue>
priority_queue<int>q; //大根堆,即大的排在前面
priority_queue<int,vector<int>,greater<int> >q;//小根堆，小的排在前面
q.top();//与普通队列唯一不同
//时间复杂度logn
```

思路

bfs模板，理解成以自己为中心向外拓展。队列用于拓展，如访问了mapp[1][1]，便将mapp[1][1]弹出

然后将mapp[1][1]相连的节点入队，以此类推，与dfs不同，不用回溯，常用于求最短路径。

注意动态变化的值

talk is cheap,show me the code.

```
#include <cstdio>
#include<iostream>
#include <cstring>
#include <queue>
#include <stack>
using namespace std;
struct Node{
    int x,y;
};
//因为有两个变量，所以用结构体存
queue<Node>que;
int mapp[410][410];
//方向数组，马走日
int dx[8]={1,1,-1,-1,2,2,-2,-2};
int dy[8]={-2,2,-2,2,-1,1,-1,1};
int main(){
    int m,n;
    Node begin;
    cin>>n>>m;
    cin>>begin.x>>begin.y;
    memset(mapp,-1,sizeof(mapp));
    //根据题意，走不到输出-1，所以把这个mapp初始化为-1(亦表示未访问过)
    mapp[begin.x][begin.y]=0;//起点走0步
    que.push(begin);//不要忘了把起点入队
    while(!que.empty()){
        Node now;
        now=que.front();que.pop();
        for(int i=0;i<8;i++){
            int xx=now.x+dx[i];
            int yy=now.y+dy[i];
            if(xx>=1&&yy>=1&&xx<=n&&yy<=m&&mapp[xx][yy]==-1){
                mapp[xx][yy]=mapp[now.x][now.y]+1;//走一步
                Node next;
                next.x=xx;
                next.y=yy;
                que.push(next);
            }
        }
    }
```

```
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            printf("%-5d",mapp[i][j]);
            //-5d用于使输出之间有5个空格且左对齐
        }
        cout<<endl;
    }
    return 0;
}
```