P1508 Likecloud-吃、吃、吃

题目背景

问世间，青春期为何物？

答曰：“甲亢，甲亢，再甲亢；挨饿，挨饿，再挨饿！”

题目描述

正处在某一特定时期之中的李大水牛由于消化系统比较发达，最近一直处在饥饿的状态中。某日上课，正当他饿得头昏眼花之时，眼前突然闪现出了一个n* m(n and m<=200)的矩型的巨型大餐桌，而自己正处在这个大餐桌的一侧的中点下边。餐桌被划分为了n*m个小方格，每一个方格中都有一个圆形的巨型大餐盘，上面盛满了令李大水牛朝思暮想的食物。李大水牛已将餐桌上所有的食物按其所能提供的能量打了分（有些是负的，因为吃了要拉肚子），他决定从自己所处的位置吃到餐桌的另一侧，但他吃东西有一个习惯——只吃自己前方或左前方或右前方的盘中的食物。

由于李大水牛已饿得不想动脑了，而他又想获得最大的能量，因此，他将这个问题交给了你。

每组数据的出发点都是最后一行的中间位置的下方！

输入输出格式

输入格式：

[输入数据：]

第一行为m n.(n为奇数)，李大水牛一开始在最后一行的中间的下方

接下来为m*n的数字距阵.

共有m行,每行n个数字.数字间用空格隔开.代表该格子上的盘中的食物所能提供的能量.

数字全是整数.

输出格式：

[输出数据：]

一个数,为你所找出的最大能量值.

输入输出样例

输入样例#1：

复制

```
6 7
16 4 3 12 6 0 3
4 -5 6 7 0 0 2
6 0 -1 -2 3 6 8
5 3 4 0 0 -2 7
-1 7 4 0 7 -5 6
0 -1 3 4 12 4 2
```

输出样例#1：

复制

```
41
```

说明

快吃！快吃！快吃！

思路

//金子扬

方法：用记忆化搜索。

思路：皇帝派3个大臣去收集钱（$），每个大臣都会拍（小）大臣，然后，大臣会选择最好的记录下来，返回上级，知道皇帝。

（用动归更方便）

//李明烨

用“数字三角形”那题进行建模

代码

```
//周子涵
#include<bits/stdc++.h>
using namespace std;
int a[205][205],n,m;
int mm[205][205];
bool vis[205][205];//存放这个点有没有访问过
int dfs(int x,int y){
    if(vis[x][y]) return mm[x][y];
    for(int i=-1;i<=1;i++)
        if(y+i>0&&y+i<=n&&x-1>0)//判断是否越界
            mm[x][y]=max(mm[x][y],dfs(x-1,y+i)+a[x][y]);//记忆化从mm[x][y]和 dfs(x-1,y+i)+a[x][y]选一个
    vis[x][y]=true;
    return mm[x][y];
}
int main(){
    memset(a,-9999,sizeof(a));
    memset(mm,-0x3f,sizeof(mm));
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            cin>>a[i][j];
```

```
    for(int i=1;i<=m;i++){//第一排无法再向下搜索,标true
        mm[1][i]=a[1][i];
        vis[1][i]=true;
    }
    dfs(n+1,m/2+1);//开始dfs,因为李大水牛一开始在最后一行的中间的下方，所以要在n+1,m/2+1开始搜
    cout<<max(mm[n][m/2],max(mm[n][m/2+1],mm[n][m/2+2]));//答案只可能在这三个点里
    return 0;
}
```

```
//金子扬
#include<iostream>
#include<cmath>
using namespace std;
int n,m,map[205][205],mm[205][205],t,maxn;
bool vis[205][205];
int dx[4]={0,-1,-1,-1},
    dy[4]={0,0,-1,1};
int dfs(int x,int y)
{
    if(vis[x][y]) return mm[x][y];
    for(int i=1;i<=3;i++)
    {
        int tx=x+dx[i];
        int ty=y+dy[i];
        if(tx<1 || tx>n || ty<1 || ty>m) continue;
        mm[x][y]=max(mm[x][y],dfs(tx,ty)+map[x][y]);//记忆化，记忆小臣的最佳答案
    }
    vis[x][y]=true;
    return mm[x][y];
}
int main()
{
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            cin>>map[i][j];
    for(int i=1;i<=m;i++)
        mm[1][i]=map[1][i];
    dfs(n+1,m/2+1);
    for(int i=m/2;i<=m/2+2;i++)
        maxn=max(maxn,mm[n][i]);
    cout<<maxn<<endl;
    return 0;
}
```

```
//殷学楷
#include<bits/stdc++.h>
using namespace std;
long long mm[201][201],g[201][201],n,m;
bool eat_plate[201][201];//标记吃过的；
int d[5]={-1,0,1};//三个方向；
long long dfs(int x,int y){
    if(eat_plate[x][y]==true) return mm[x][y];//如果吃过，就不走；
    if(x==1){//如果吃到了第一行，return；
        mm[x][y]=g[x][y];
```

```
        return mm[x][y];
    }
    for(int i=0;i<3;i++)//搜索；
    {
        if ((y+d[i]>0)&&(y+d[i]<=n)) mm[x][y]=max(mm[x][y],dfs(x-1,y+d[i])+g[x][y]);//如果走下去不越界，更新；
    }
    eat_plate[x][y]=true;//标记吃过的；
    return mm[x][y];
}
int main() {
    //输入；
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>g[i][j];
        }
    }
    memset(mm,-0x10000f,sizeof(mm));//初始化；
    dfs(n+1,m/2+1);//搜索；
    cout<<max(mm[n][m/2],max(mm[n][m/2+1],mm[n][m/2+2]));//输出三个方向中最大的那个；
    return 0;
}
/*记忆化搜索*/
```

动态规划的代码

```
//钱嘉欢、
#include<iostream>
using namespace std;
int n,m,a[205][205],x;
int main()
{
    cin>>m>>n;
    for(int i=1;i<=m;i++)
        for(int j=1;j<=n;j++)
            cin>>a[i][j];
    //输入
    for(int i=2;i<=m+1;i++)
        for(int j=1;j<=n;j++)
        {
            //x是max的简写
            x=a[i-1][j];
            if(x<a[i-1][j-1]) x=a[i-1][j-1];
            if(x<a[i-1][j+1]) x=a[i-1][j+1];
            //求出a[i][j]前面三个最大是那个数
            a[i][j]=a[i][j]+x;
            //a[i][j]等于前面所提供的能量再加上a[i][j]
        }
    //动态规划
    cout<<a[m+1][(n+1)/2];//输出
    return 0;
}
```

```
//李明烨
#include<cstdio>
#include<iostream>
#include<algorithm>
```

```cpp
using namespace std;
long long m,n,/*长宽*/map[1005][1005]/*类似数字三角形*/,hun/*获取的能量*/;
int main(){
    scanf("%lld%lld",&m,&n);//输入
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++){
            scanf("%lld",&map[i][j]);//输入
            map[i][j]+=max(map[i-1][j-1],max(map[i-1][j],map[i-1][j+1]));//我们将地图绕中心点顺时针旋转180°，就好像1153“数字三角形”的扩展版了（枚举三个方向：左上，正上，右上）
        }
    }
    printf("%lld\n",max(map[m][n/2],max(map[m][n/2+1],map[m][n/2+2])));//比较三个第一次到达的点：底下的中心点左边，中心点，中心点右边。
    return 0;
}
```

```cpp
//叶炜辰
#include <iostream>
using namespace std;
int a[1005][1005];
int main(){
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>a[i][j];
            a[i][j]+=max(a[i-1][j-1],max(a[i-1][j],a[i-1][j+1]));//动归，看他上面的点，右上的点，左上的点那个大 ，就加那个点
        }
    }
    cout<<max(a[n][m/2],max(a[n][m/2+1],a[n][m/2+2]));//注意是a[n][m/2+1]，因为他是从中间位置的下方开始走的！
    return 0;
}
```