

CS101

A Mars rover, likely a Curiosity rover, is shown on a rocky, reddish-brown landscape. The rover is white with various instruments and cameras. It has six large, treaded wheels. The background shows a hazy, orange-tinted sky and distant hills. The overall scene is a typical Mars surface environment.

信奥
算法

课件下载地址:

<http://pan.baidu.com/s/1o885tz0>

作业网站:

<http://120.132.18.213:8080/thrall-web/main#home>

二维递推问题

综合练习

棋盘路径计数 - 无障碍

在棋盘格上，小明站在第1行第1列的位置（左上角），罗马在第n行第m列的位置（右下角）。小明每一步只可以向右走一格或者向下走一格。输入n和m ($1 \leq m, n \leq 20$)，输出小明有多少种方法可以走到罗马。

输入样例：

2 2

输入样例：

4 3

输出样例：

2

输出样例：

10

棋盘路径计数 - 无障碍

$f[i][j]$ 表示 到第*i*行第*j*列有多少种走法

棋盘路径计数 - 无障碍

$f[i][j]$ 表示 到第*i*行第*j*列有多少种走法

	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	1	1	1	1
i=2	0	1	2	3	4
i=3	0	1	3	6	10
i=4	0	1	4	10	20

缓冲带

缓冲带

棋盘路径计数 - 无障碍

$f[i][j]$ 表示 到第*i*行第*j*列有多少种走法

	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	1	1	1	1
i=2	0	1	2	3	4
i=3	0	1	3	6	10
i=4	0	1	4	10	20

缓冲带

缓冲带

$$f[1][1] = 1$$

(i,j)不是(1,1)时 $f[i][j] = f[i-1][j] + f[i][j-1]$

初始
条件
递推
方程

```
1 #include<iostream>
2 #define M 21
3 using namespace std;
4 long long f[M][M],n,m,i,j;
5 int main(){
6     cin>>n>>m;
7     for(i=1;i<=n;i++)
8         for(j=1;j<=m;j++)
9             if(i==1&&j==1) f[1][1]=1;
10            else f[i][j]=f[i-1][j]+f[i][j-1];
11     cout<<f[n][m]<<endl;
12     return 0;
13 }
```


棋盘路径计数 - 有障碍

在棋盘格上，小明站在第1行第1列的位置（左上角），目的地在第5行第5列的位置（右下角）。小明每一步只可以向右走一格或者向下走一格。

输入5行5列的棋盘，0表示无障碍可以通行，#表示此处有障碍，输出小明有多少种方法可以走到目的地。

输入样例：

```
00000
0###0#
0###00
0###0
00000
```

输出样例：

2

输入样例：

```
00000
##0##
##0##
##000
##000
```

输出样例：

3

棋盘路径计数 - 有障碍

在棋盘格上，小明站在第1行第1列的位置（左上角），目的地在第5行第5列的位置（右下角）。小明每一步只可以向右走一格或者向下走一格。

输入5行5列的棋盘，0表示无障碍可以通行，#表示此处有障碍，输出小明有多少种方法可以走到目的地。

输入样例：

```
00000
0##0#
0##00
0###0
00000
```

输入样例：

```
00000
##0##
##0##
##000
##000
```

输出样例：

2

输出样例：

3

```
char d[6][6];
数组d保存整张地图
```

`d[i][j]`表示 第*i*行第*j*列的字符

棋盘路径计数 - 有障碍

$f[i][j]$ 表示 到第*i*行第*j*列有多少种走法

棋盘路径计数 - 有障碍

$f[i][j]$ 表示 到第*i*行第*j*列有多少种走法

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

缓冲带

缓冲带

$$f[1][1] = (d[1][1] == '0')$$

初始
条件

棋盘路径计数 - 有障碍

$f[i][j]$ 表示 到第*i*行第*j*列有多少种走法

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

缓冲带

缓冲带

$$f[1][1] = (d[1][1] == '0')$$

初始
条件

(i,j)不是(1,1)

若 $d[i][j] = \text{'\#'}'$

$$f[i][j] = 0$$

若 $d[i][j] = \text{'0'}$

$$f[i][j] = f[i-1][j] + f[i][j-1]$$

递推
方程

```
1  #include<iostream>
2  #define N 5
3  using namespace std;
4  int f[6][6],i,j;
5  char d[6][6];
6  int main(){
7      for(i=1;i<=N;i++)
8          for(j=1;j<=N;j++)
9              cin>>d[i][j];
10     for(i=1;i<=N;i++)
11         for(j=1;j<=N;j++)
12             if(i==1&&j==1) f[1][1]=(d[i][j]=='0');
13             else if(d[i][j]=='#') f[i][j]=0;
14             else f[i][j]=f[i-1][j]+f[i][j-1];
15     cout<<f[N][N]<<endl;
16     return 0;
17 }
```

棋盘最优路径 - 无障碍

在棋盘格子里有些金币。小明站在第1行第1列（左上角），目的地在第5行第5列（右下角），可以取走路过每一格里的金币。小明每一步只可以向右走一格或者向下走一格。输入5行5列的棋盘，每格数字表示金币个数。输出小明走到目的地最多能拿多少金币。

输入样例：

```
00000
01104
01100
01110
00000
```

输出样例：

6

输入样例：

```
10000
00000
00202
03000
00001
```

输出样例：

6

棋盘最优路径 - 无障碍

在棋盘格子里有些金币。小明站在第1行第1列（左上角），目的地在第5行第5列（右下角），可以取走路过每一格里的金币。小明每一步只可以向右走一格或者向下走一格。输入5行5列的棋盘，每格数字表示金币个数。输出小明走到目的地最多能拿多少金币。

输入样例：

```
00000
01104
01100
01110
00000
```

输入样例：

```
10000
00000
00202
03000
00000
```

输出样例：

6

输出样例：

5

```
int d[6][6];
```

数组d保存整张地图

$d[i][j]$ 表示 第i行第j列金币数

棋盘最优路径 - 无障碍

$g[i][j]$ 表示 到第*i*行第*j*列时最多拿走多少金币

	j=0	j=1	j=2	j=3	j=4	
i=0	0	0	0	0	0	← 缓冲带
i=1	0					
i=2	0		↓	↓	↓	
i=3	0		↓	↓	↓	

↑
缓冲带

$$g[1][1] = d[1][1]$$

初始
条件

(i,j)不是(1,1)

$$g[i][j] = \max(g[i-1][j], g[i][j-1]) + d[i][j]$$

递推
方程

```
1 #include<iostream>
2 #define N 5
3 using namespace std;
4 int d[6][6],g[6][6],i,j;
5 int main(){
6     for(i=1;i<=N;i++)
7         for(j=1;j<=N;j++){
8             char ch; cin>>ch;
9             d[i][j]=ch-'0';
10        }
11    for(i=1;i<=N;i++)
12        for(j=1;j<=N;j++){
13        if(i==1&&j==1) g[1][1]=d[i][j];
14        else g[i][j]=max(g[i-1][j],g[i][j-1])+d[i][j];
15    cout<<g[N][N]<<endl;
16    return 0;
17 }
```