CS102

C++ 算法

深度优先搜索
Depth-first Search

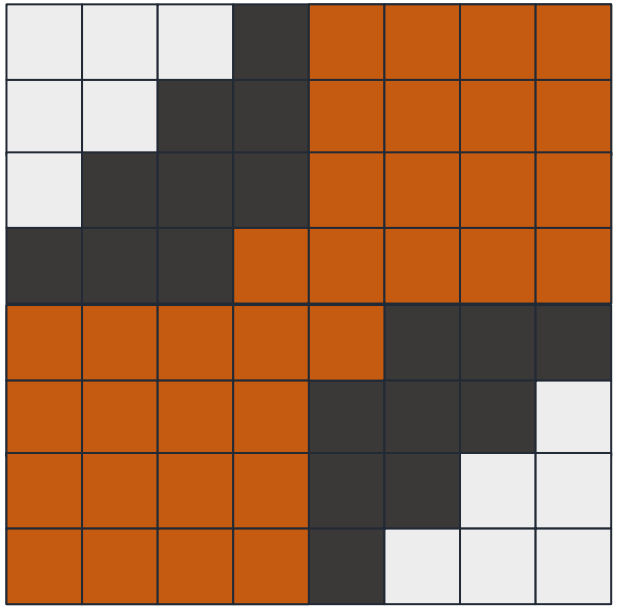
洪水填充算法
Flood Fill

上色：填充颜色

上色：填充颜色

Flood Fill

Flood fill, also called seed fill, is an algorithm that determines the area connected to a given node in a multi-dimensional array



连通方向

四个方向

	(x-1,y)	
(x,y-1)	(x,y)	(x,y+1)
	(x+1,y)	

```
int dx[4]={0,1,0,-1};
int dy[4]={1,0,-1,0};
```

八个方向

(x-1,y-1)	(x-1,y)	(x-1,y+1)
(x,y-1)	(x,y)	(x,y+1)
(x+1,y-1)	(x+1,y)	(x+1,y+1)

```
int dx[8]={1,1,1,0,0,-1,-1,-1};
int dy[8]={1,0,-1,1,-1,1,0,-1};
```

四方向连通

输入第x行第y列的格子坐标。
输出共4行：(x,y)的四个方向连通的坐标

输入样例
3 5
输出样例
(3,6)
(4,5)
(3,4)
(2,5)

```
#include<iostream>
using namespace std;
int dx[4]={0,1,0,-1};
int dy[4]={1,0,-1,0};
int x,y;
int main(){
    cin>>x>>y;
    for(int k=0;k<4;k++){
        int nx=x+dx[k],ny=y+dy[k];
        cout<<"("<<nx<<","<<ny<<")"<<endl;
    }
    return 0;
}
```

八方向连通

输入第x行第y列的格子坐标。
输出共8行：(x,y)的八个方向连通的坐标

输入样例
3 5
输出样例
(4,6)
(4,5)
(4,4)
(3,6)
(3,4)
(2,6)
(2,5)
(2,4)

```
1  #include<iostream>
2  using namespace std;
3  int dx[8]={1,1,1,0,0,-1,-1,-1};
4  int dy[8]={1,0,-1,1,-1,1,0,-1};
5  int x,y;
6  int main(){
7      cin>>x>>y;
8      for(int k=0;k<8;k++){
9          int nx=x+dx[k],ny=y+dy[k];
10         cout<<"("<<nx<<","<<ny<<")"<<endl;
11     }
12     return 0;
13 }
```

发洪水

在n*m格的地图上发洪水了，水从第a行第b列涌入。'-'代表空地，'+'代表墙体。如果两块空地是八向连通的，那么他们同时被淹或者同时不被淹。输入n,m,a,b和原始地图。输出被淹后的地图，'@'代表被淹的格子。n,m<=100

输入样例
8 8 8 1
---+----
--++----
-+++----
+++-----
-----+++
----+++-
----++--
----+---

输入样例
4 4 1 1
---+
--+-
-+--
+---

输入样例
4 5 4 5
---++
-+-+-
-++++

输出样例
---+@@@@
--++@@@@
-+++@@@@
+++@@@@@
@@@@@+++
@@@@+++-
@@@@++--
@@@@+---

输出样例
@@@+
@@+@
@+@@
+@@@

输出样例
@@@++
@+@+-
@++++
@@@@@

发洪水: 深度优先搜索算法

递归方式实现深度优先搜索算法(depth-first search简称dfs)

从(x,y)开始搜索：

淹没(x,y), 修改地图中(x,y)符号为'@'

依次循环查看8个方向，对于第k个方向：

沿第k个方向，从(x,y) 走到新位置(nx,ny)

若 (nx,ny)没越界，且(nx,ny)符号为'-' ：

从(nx,ny)开始搜索

发洪水: 深度优先搜索算法

递归方式实现深度优先搜索算法(depth-first search简称dfs)

不断沿某个方向往更深处探索
直到无法再深入时退回，再尝试其他方向

```
8  void dfs(int x,int y){
9      d[x][y]='@';
10     for(int k=0;k<8;k++){
11         int nx=x+dx[k],ny=y+dy[k];
12         if(nx>=1&&nx<=n&&ny>=1&&ny<=m&&d[nx][ny]=='-')
13             dfs(nx,ny);
14     }
15 }
```

从(x,y)开始搜索：

淹没(x,y), 修改地图中(x,y)符号为'@'

依次循环查看8个方向：

走到新位置(nx,ny)

若 (nx,ny)没越界且(nx,ny)符号为'-'

从(nx,ny)开始搜索

主程序调用dfs(a,b)启动洪水

打印中间步骤

运行程序“发洪水2打印中间步骤”

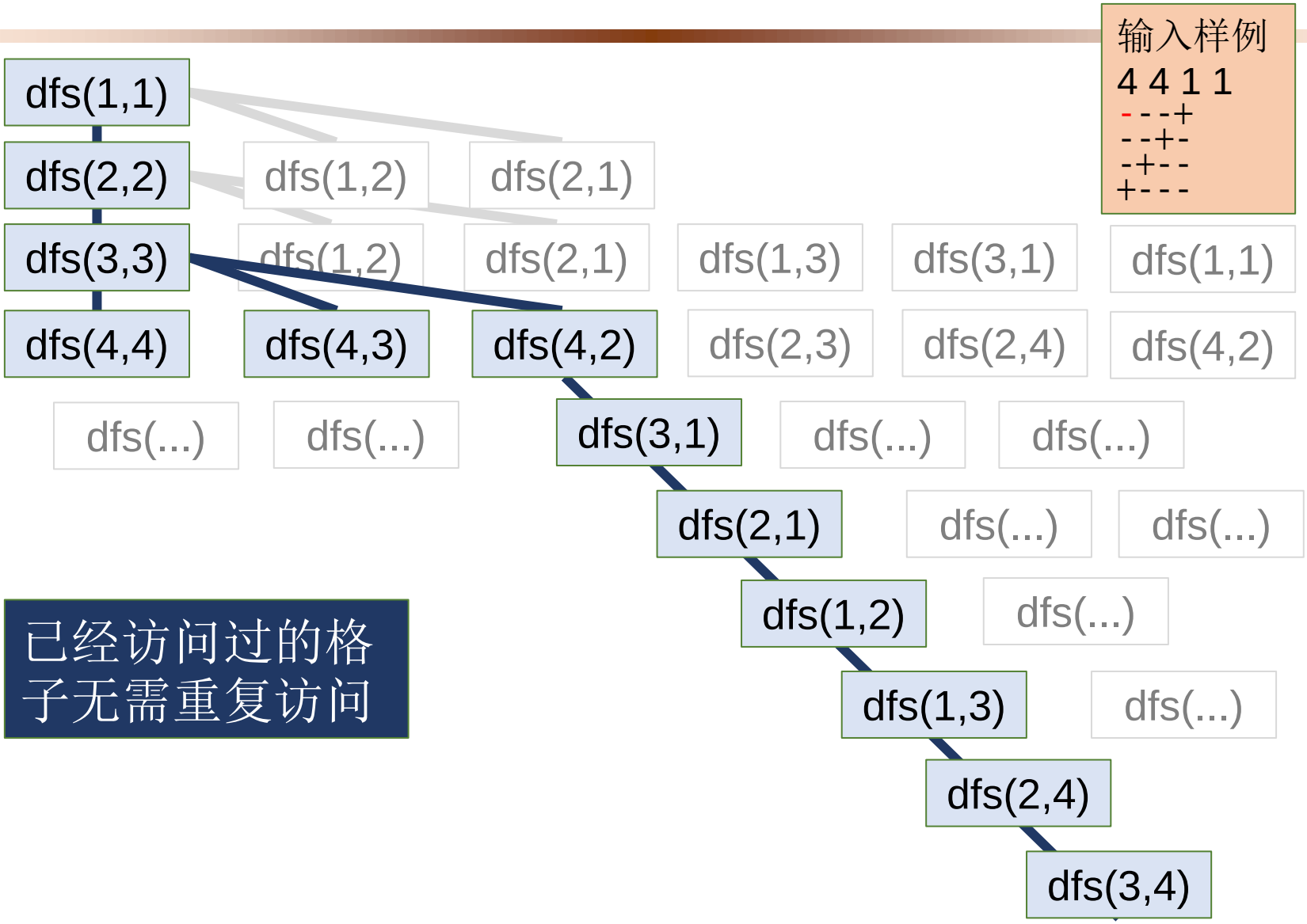
输入地图信息

观察每一步搜索步骤

思考搜索顺序

```
 9 void print(){
10     cout<<"==============="<<endl;
11     for(int i=1;i<=n;i++,cout<<endl)
12         for(int j=1;j<=m;j++)cout<<d[i][j];
13     string s;getline(cin,s);
14 }
```

递归：搜索顺序


输入样例
4 4 1 1
---+
--+-
-+--
+---
dfs(1,1)
dfs(2,2)
dfs(3,3)
dfs(4,4)
dfs(4,3)
dfs(4,2)
dfs(3,1)
dfs(2,1)
dfs(1,2)
dfs(1,3)
dfs(2,4)
dfs(3,4)
dfs(1,2)
dfs(2,1)
dfs(1,2)
dfs(2,1)
dfs(1,3)
dfs(3,1)
dfs(1,1)
dfs(2,3)
dfs(2,4)
dfs(4,2)
dfs(...)
dfs(...)
dfs(...)
dfs(...)
dfs(...)
dfs(...)
dfs(...)
dfs(...)
已经访问过的格子无需重复访问

发洪水：错误代码1

```
 8 void dfs(int x,int y){
 9     for(int k=0;k<8;k++){
10         int nx=x+dx[k],ny=y+dy[k];
11         if(nx>=1&&nx<=n&&ny>=1&&ny<=m&&d[nx][ny]=='-')
12             dfs(nx,ny);
13     }
14 }
```

错在哪?
为什么错?

发洪水：错误代码2

```
 8  void dfs(int x,int y){
 9      d[x][y]='@';
10      for(int k=0;k<8;k++){
11          int nx=x+dx[k],ny=y+dy[k];
12          if(d[nx][ny]=='-')
13              dfs(nx,ny);
14      }
15  }
```

错在哪?
为什么错?

发洪水：错误代码3

```
 8 void dfs(int x,int y){
 9     d[x][y]='@';
10     for(int k=0;k<8;k++){
11         int nx=x+dx[k],ny=y+dy[k];
12         if(nx>=1&&nx<=n&&ny>=1&&ny<=m&&d[nx][ny]!='+')
13             dfs(nx,ny);
14     }
15 }
```

错在哪?
为什么错?

淹了吗：判断连通性

在n*m格的地图上发洪水了，水从第a行第b列涌入。'-'代表空地，'+'代表墙体。如果两块空地是八向连通的，那么他们同时被淹或者同时不被淹。你家住在第n行第m列。输入n,m,a,b和原始地图。输出你家是否被淹。n,m<=100

输入样例
4 4 1 1
---+
--+-
-+--
+---

输入样例
4 5 1 5
---+-
-+-+-
-++++

输出样例
Sadly yes

输出样例
Luckily no

如何利用dfs判断连通性?

淹了吗：判断连通性

vst[i][j]记录(i,j)是否访问过

发现连通后马上逐步退出

```
 8  bool vst[N][N];
 9  void dfs(int x,int y){
10      vst[x][y]=1;
11      if(vst[n][m])return;
12      for(int k=0;k<8;k++){
13          int nx=x+dx[k],ny=y+dy[k];
14          if(nx>=1&&nx<=n&&ny>=1&&ny<=m&&d[nx][ny]=='-'&&!vst[nx][ny])
15              dfs(nx,ny);
16      }
17  }
```

连通性DFS演示

请打开程序
“连通性DFS演示”

请运行程序

观察搜索步骤

输入
4 5
.....
.+.+.
.++++
.....

输入
7 7
...+...
.+.+.+.
.+...+.
.++++++
....
.+.+.+.
....

受灾面积：累计连通区域面积

在n*m格的地图上发洪水了，水从第a行第b列涌入。'-'代表空地，'+'代表墙体。如果两块空地是八向连通的，那么他们同时被淹或者同时不被淹。每一格面积为1，请求出受灾面积。输入n,m,a,b和原始地图。输出受灾面积。n,m<=100

输入样例
4 4 1 1
---+
--+-
-+--
+---

输入样例
4 5 4 5
---+-
-+-+-
-++++

输出样例
12

输出样例
11

如何利用dfs计算连通面积?

受灾面积：累计连通区域面积

```
 6  char d[N][N];
 7  bool vst[N][N];
 8  int n,m,a,b,area;
 9  void dfs(int x,int y){
10      vst[x][y]=1;
11      area++;
12      for(int k=0;k<8;k++){
13          int nx=x+dx[k],ny=y+dy[k];
14          if(nx>=1&&nx<=n&&ny>=1&&ny<=m&&d[nx][ny]=='-'&&!vst[nx][ny])
15              dfs(nx,ny);
16      }
17  }
```

vst[i][j]记录(i,j)是否访问过

第一次访问(x,y)格子时累积面积

数连通区域个数

在n*m格的地图上发洪水了，水从某些格子涌入。'-'代表空地，'+'代表墙体。如果两块空地是八向连通的，那么他们同时被淹或者同时不被淹。要使整张地图的空地受灾，需要几格涌入洪水。输入n,m和原始地图。输出需要几个涌入格子。n,m<=100

输入样例
4 4 1 1
---+
--+-
-+--
+---

输入样例
4 5
-+-+-
-+-+-
-++++

输出样例
1

输出样例
3

如何利用dfs数连通区域个数?

数连通区域个数

```
22      for(int i=1;i<=n;i++)
23          for(int j=1;j<=m;j++)
24              if(d[i][j]=='-'&&!vst[i][j]){
25                  cnt++;
26                  dfs(i,j);
27              }
28      cout<<cnt<<endl;
```

cnt记录连通区域个数

每次dfs把连通区域内都访问完

记忆化搜索

搜索时，记住搜索过的情况

直接修改地图

vst[][]数组标记是否走过

dfs总结

填充颜色

判断连通性

累计连通区域面积

数连通区域个数

课件下载链接：
链接：https://pan-baidu-com/s/1ei7f7w
密码：q66i

作业网站：
http://120-132-18-213:8080/thrall-web/main#home