

Graph Traversals

Dr. Mattox Beckman

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
DEPARTMENT OF COMPUTER SCIENCE



Objectives

Your Objectives:

- ▶ Implement DFS and BFS
- ▶ Show how to use these to solve some classic graph problems:
 - ▶ connected components

DFS Basics

- ▶ Step 1: Mark self as visited
- ▶ Step 2: Visit all unvisited children
- ▶ Step 3: ???
- ▶ Step 4: Profit!

Recursive DFS Code (from the book)

```
0 typedef pair<int, int> ii;
1 typedef vector<ii> vii; // edge is (neighbor, weight) pair
2 typedef vector<int> vi;
3
4 vi dfs_num;
5
6 void dfs(int u) {
7     // we mark the vertex as visited
8     dfs_num[u] = VISITED; // == 1, UNVISITED == -1
9     for (int j = 0; j < (int)AdjList[u].size(); j++) {
10         ii v = AdjList[u][j];
11         if (dfs_num[v.first] == UNVISITED)
12             dfs(v.first);
13 } }
```

BFS Basics

- ▶ Step 1: Mark self as visited
- ▶ Step 2: Enqueue all unvisited children
- ▶ Step 3: Dequeue next child and visit
- ▶ Step 4: ???
- ▶ Step 5: Profit!

BFS Code

```
0 vi d(V, INF); d[s] = 0; // initialize source distance
1 queue<int> q; q.push(s); // start from source
2 while (!q.empty()) {
3     int u = q.front(); q.pop();
4     for (int j = 0; j < (int)AdjList[u].size(); j++) {
5         int v = AdjList[u][j];
6         if (d[v.first] == INF) {
7             d[v.first] = d[u] + 1;
8             q.push(v.first);
9     } } }
```

Connected Components

```
0 numCC = 0;
1 dfs_num.assign(V, UNVISITED);
2 for (int i = 0; i < V; i++)
3     if (dfs_num[i] == UNVISITED) {
4         printf("CC %d:", ++numCC);
5         dfs(i);
6         printf("\n");
7     }
```

Flood Fill

```
0 int dr[] = {1,1,0,-1,-1,-1, 0, 1};
1 int dc[] = {0,1,1, 1, 0,-1,-1,-1};
2
3 int floodfill(int r, int c, char c1, char c2) {
4     if (r < 0 || r >= R || c < 0 || c >= C) return 0;
5     if (grid[r][c] != c1) return 0;
6     int ans = 1;
7     grid[r][c] = c2;
8     for (int d = 0; d < 8; d++)
9         ans += floodfill(r + dr[d], c + dc[d], c1, c2);
10    return ans;
11 }
```


Topological Sorting

```
0 vi ts; // the toposort vector
1
2 void toposort(int u) {
3     dfs_num[u] = VISITED;
4     for (int j = 0; j < (int)AdjList[u].size(); j++) {
5         vi v = AdjList[u][j];
6         if (dfs_num[v.first] == UNVISITED)
7             toposort(v.first);
8     }
9     ts.push_back(u);
10 }
```

Calling It

```
0 // in main
1
2 ts.clear();
3 memset(dfs_num, UNVISITED, sizeof dfs_num);
4 for (int i = 0; i < V; i++)
5     if (dfs_num[i] == UNVISITED)
6         dfs2(i);
```