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

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


Connected Components

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


Flood Fill

```
int dr[] = {1,1,0,-1,-1,-1, 0, 1};
int dc[] = {0,1,1, 1, 0,-1,-1,-1};

int floodfill(int r, int c, char c1, char c2) {
  if (r < 0 || r >= R || c < 0 || c >= C) return 0;
  if (grid[r][c] != c1) return 0;
  int ans = 1;
  grid[r][c] = c2;
  for (int d = 0; d < 8; d++)
      ans += floodfill(r + dr[d], c + dc[d], c1, c2);
  return ans;
}
```

Topological Sorting

```
vi ts;  // the toposort vector

void toposort(int u) {
   dfs_num[u] = VISITED;
   for (int j = 0; j < (int)AdjList[u].size(); j++) {
      ii v = AdjList[u][j];
      if (dfs_num[v.first] == UNVISITED)
         toposort(v.first);
   }
   ts.push_back(u);
}
```


Calling It

```
// in main

ts.clear();
memset(dfs_num, UNVISITED, sizeof dfs_num);
for (int i = 0; i < V; i++)
    if (dfs_num[i] == UNVISITED)
       dfs2(i);
```