Week 9 - Wednesday

COMP 2100

Last time

- What did we talk about last time?
- Symbol tables (maps) in the Java Collection Framework
- Graph definitions
- Started graph representations

Questions?

Project 3

Assignment 4

Graph Representations

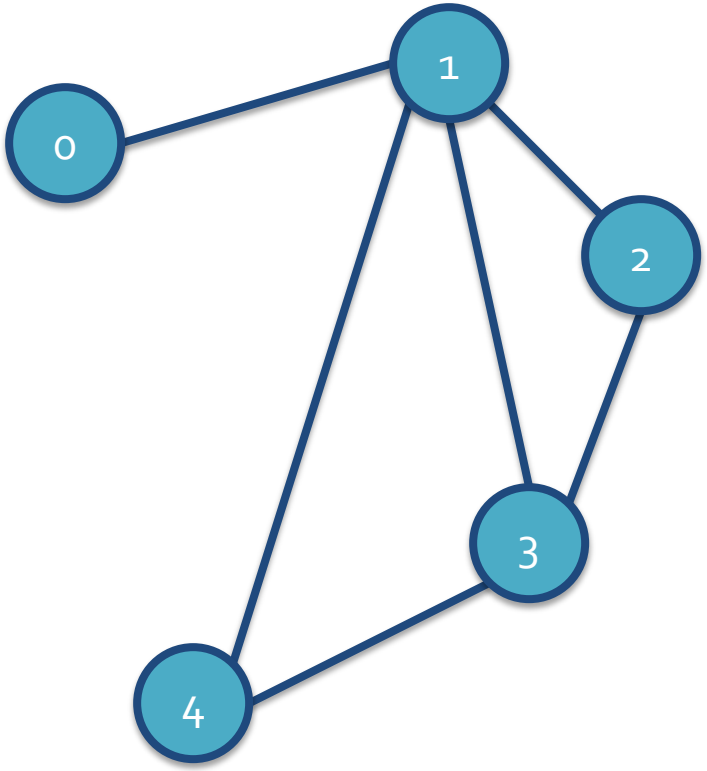
Implementing the graph ADT

- The book mentions four implementations:
  - **Adjacency matrix**
  - Array of edges
  - **Adjacency lists**
  - Adjacency sets
- We will talk about adjacency matrices and adjacency lists

Adjacency matrix

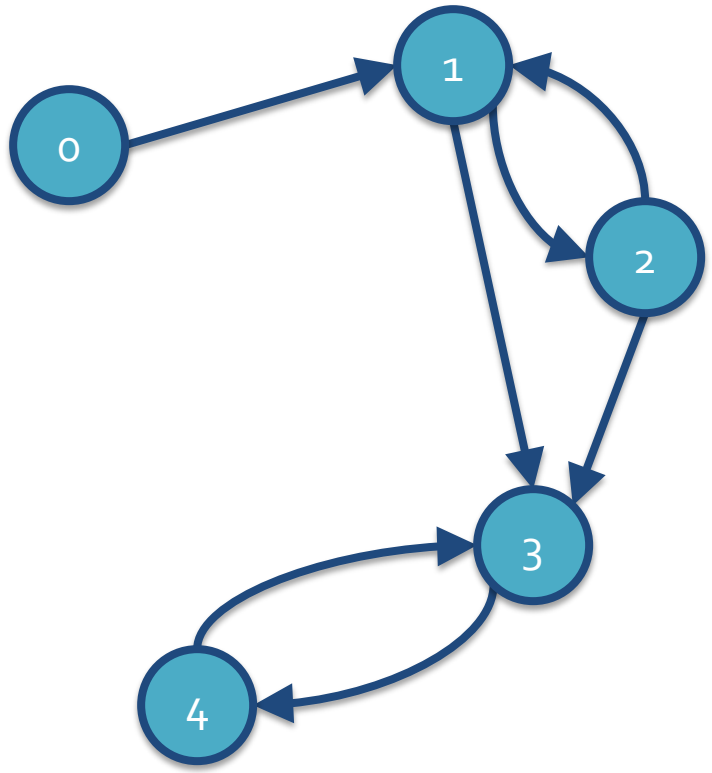
- A simple way of keeping track of the edges in a graph is an **adjacency matrix**
- An adjacency matrix is an $n \times n$ matrix where n is the number of nodes
- The number in row i column j is the number of edges between node i and node j
- Undirected graphs have symmetrical adjacency matrices
- The weakness of an adjacency matrix is that it uses $\Theta(n^2)$ space, even for sparse graphs

Adjacency matrix example



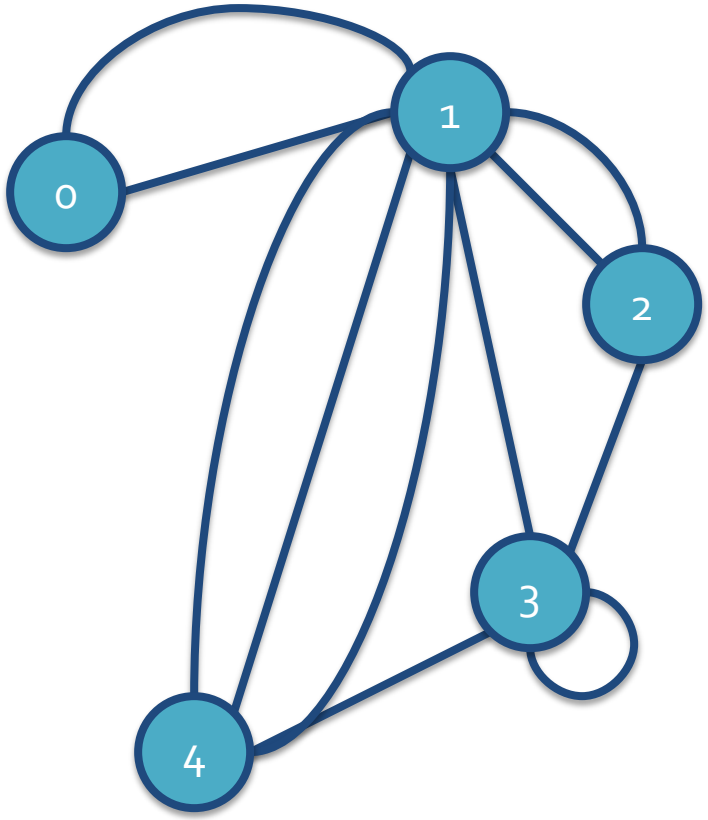
	0	1	2	3	4
0	0	1	0	0	0
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	0	1	0	1	0

Directed graph example



	0	1	2	3	4
0	0	1	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	0	0	0	1
4	0	0	0	1	0

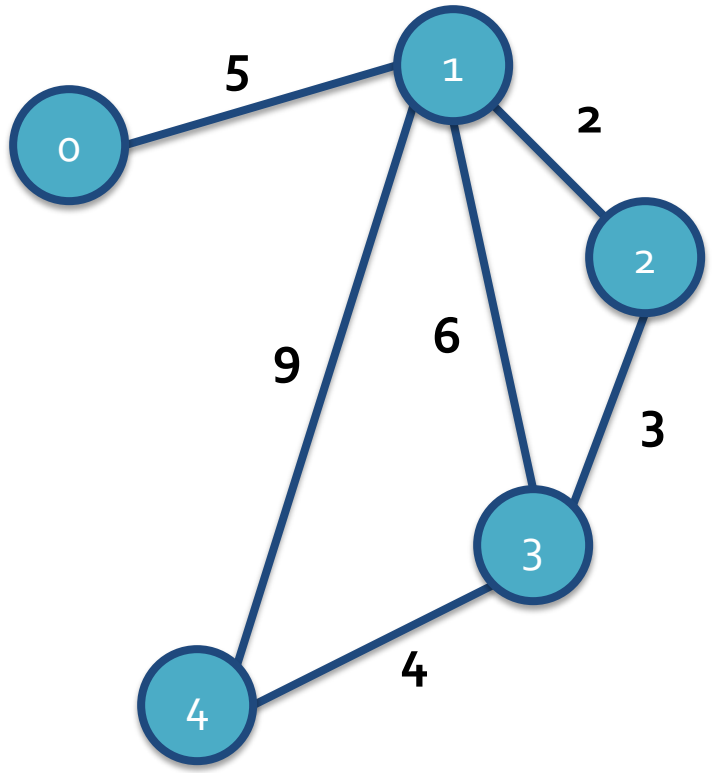
Multigraph example



	0	1	2	3	4
0	0	2	0	0	0
1	2	0	2	1	3
2	0	2	0	1	0
3	0	1	1	1	1
4	0	3	0	1	0

Weighted graph example

Alternatively, the numbers in the matrix can represent the weights on edges.

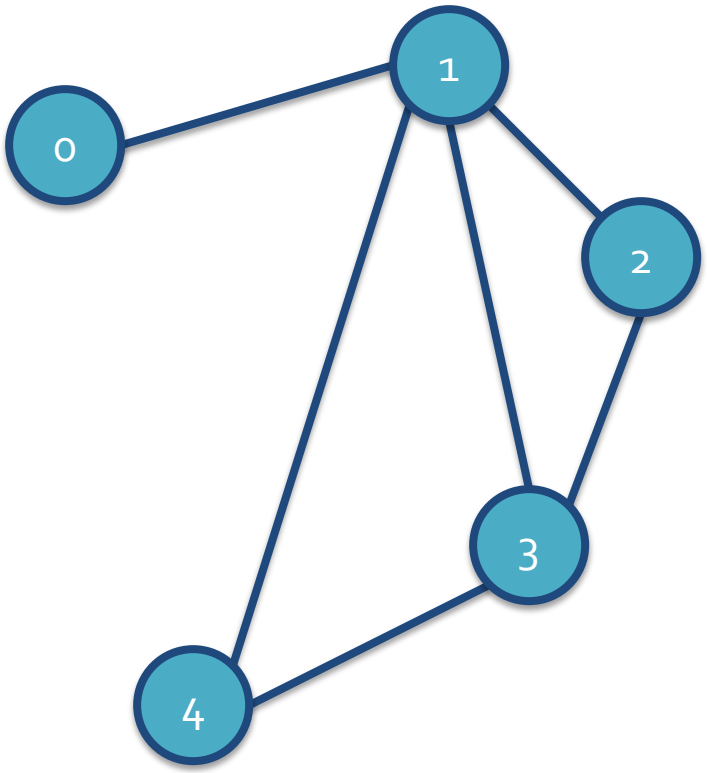


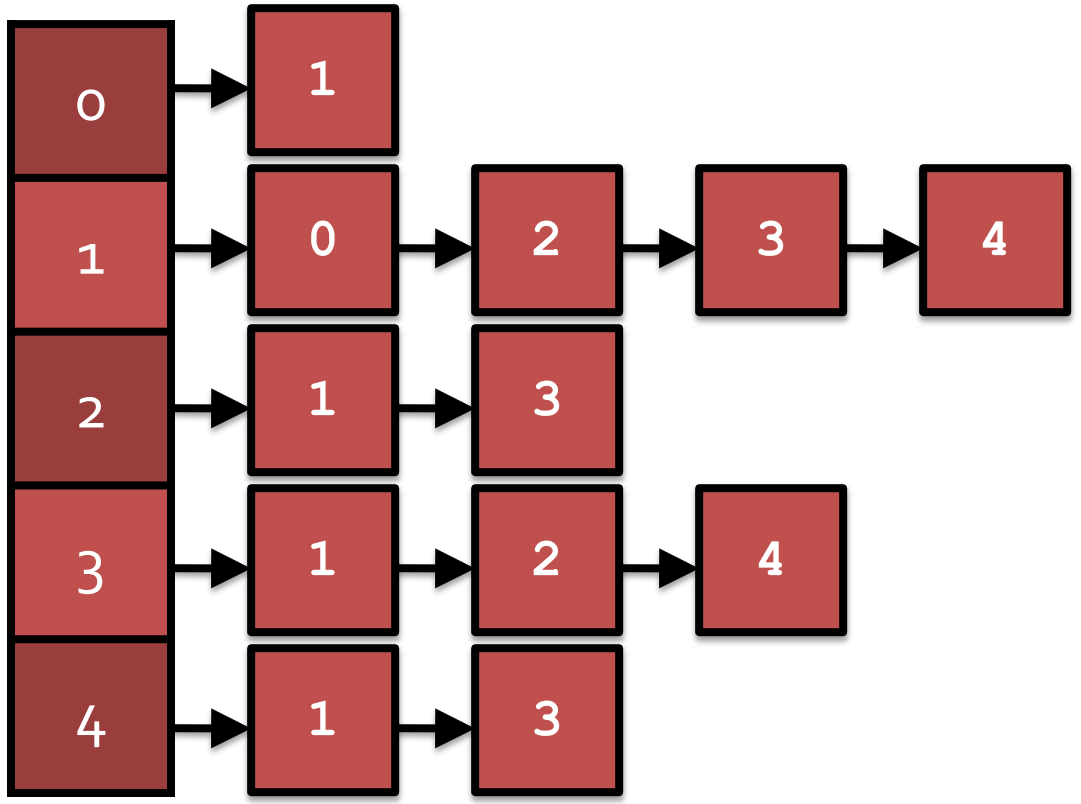
	0	1	2	3	4
0	0	5	0	0	0
1	5	0	2	6	9
2	0	2	0	3	0
3	0	6	3	0	4
4	0	9	0	4	0

Adjacency lists

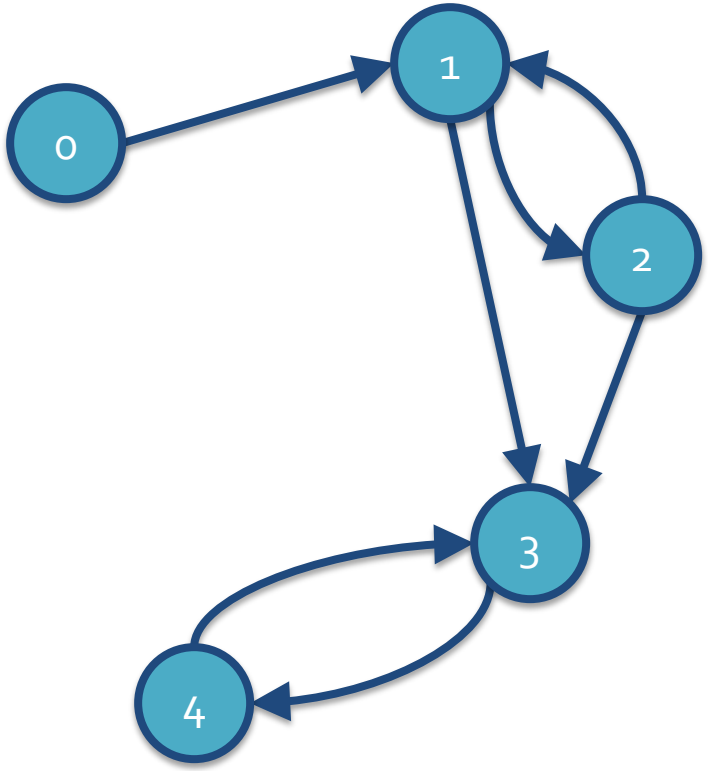
- An adjacency matrix wastes a lot of space if the graph is not very dense
- An alternative is an adjacency list
- The form of an adjacency list is an array of length ***n*** where the ***i***th element is a pointer to a linked list (or dynamically allocated array) of the nodes adjacent to node ***i***
- This is the approach the book focuses on, since most graphs are not dense

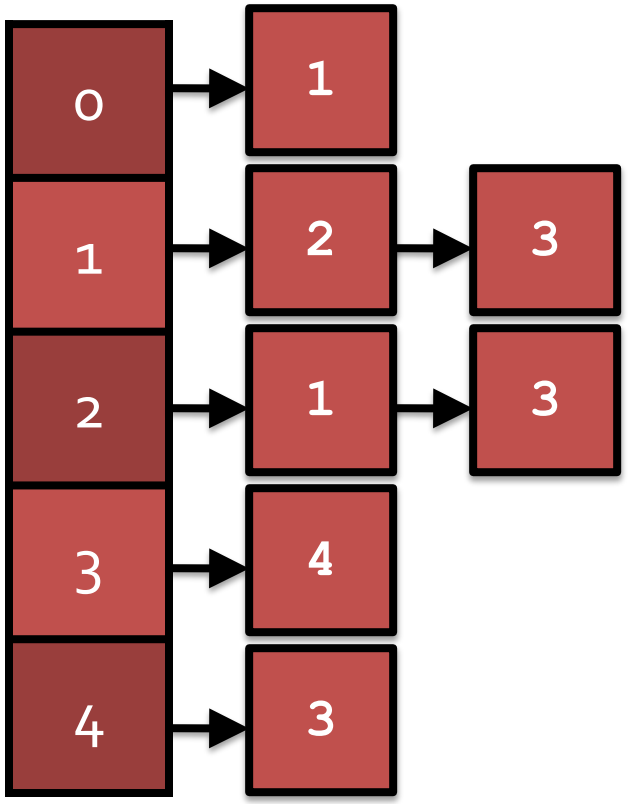
Adjacency list example



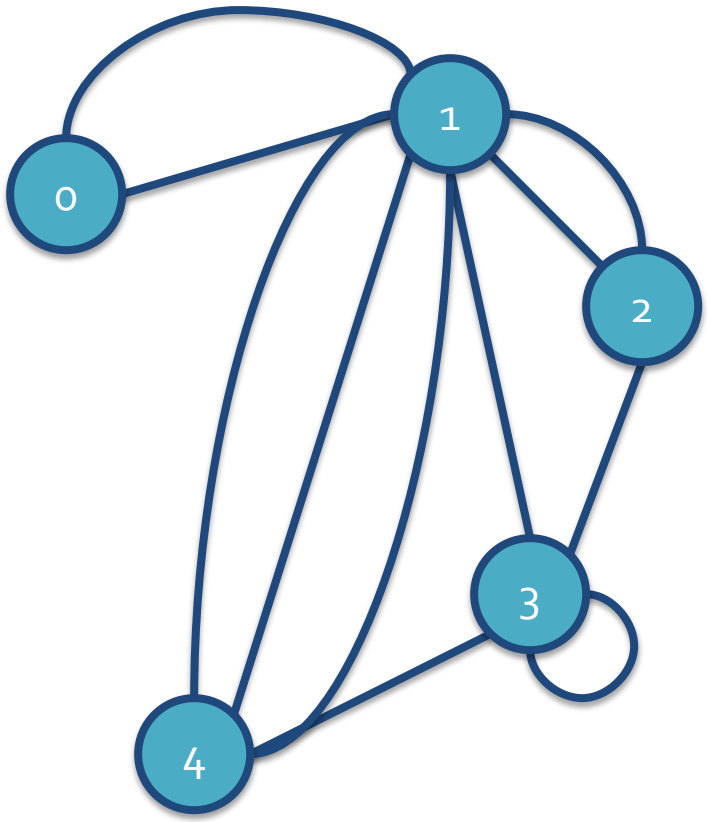


Directed graph adjacency list





Multigraph example



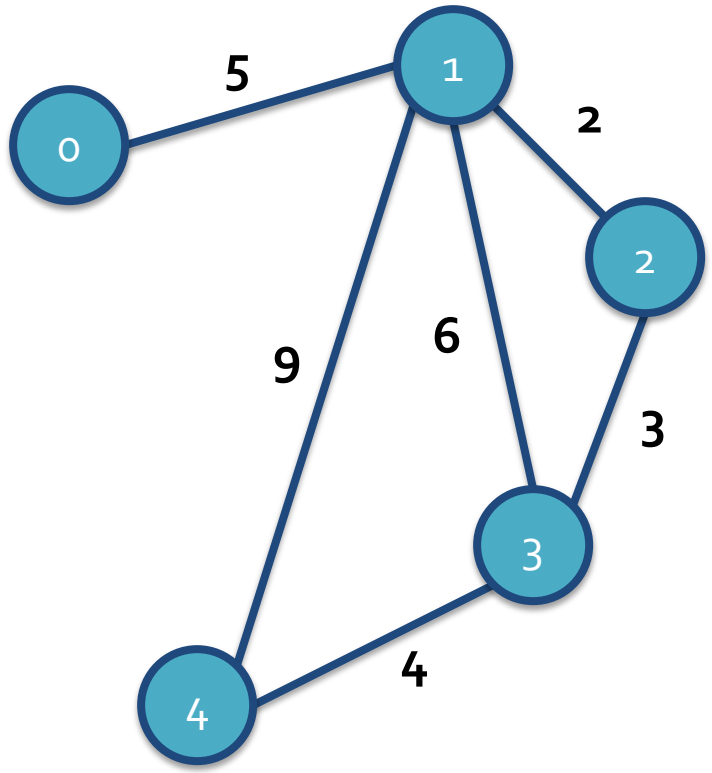
It's a trick!

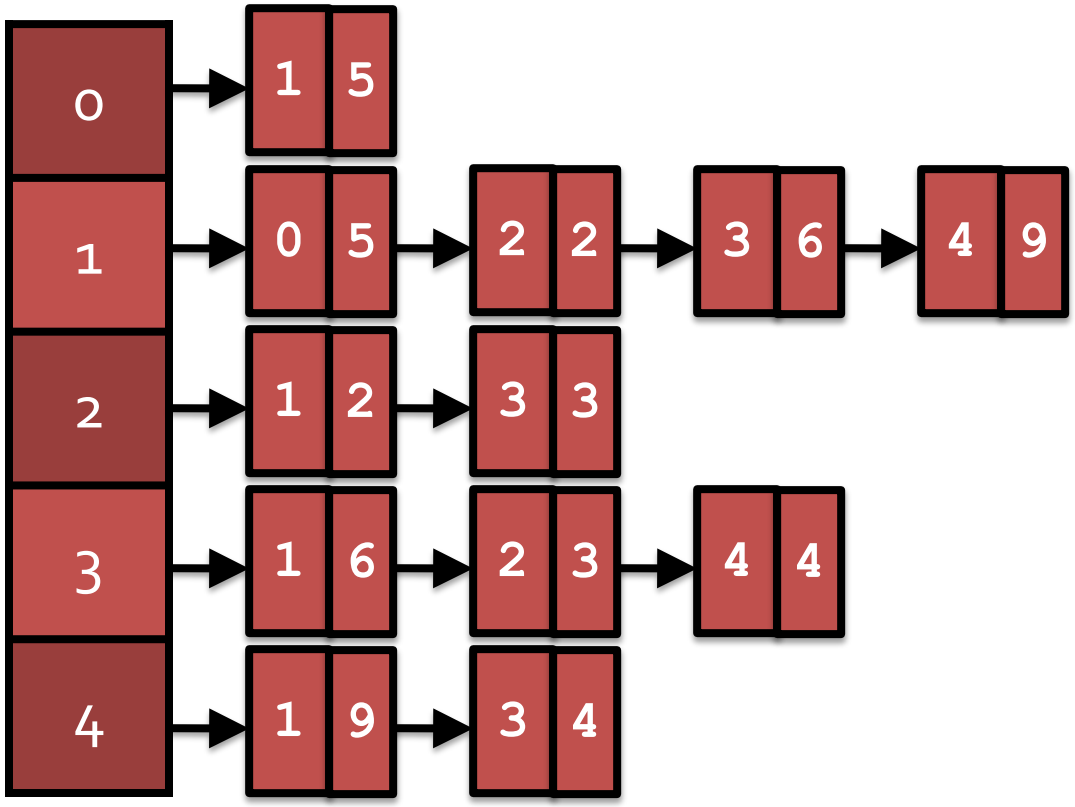
Some other steps must be taken to represent a multigraph with an adjacency list.

Each node in the linked list must contain additional information.

Weighted graph example

Again, we need extra information in the lists.



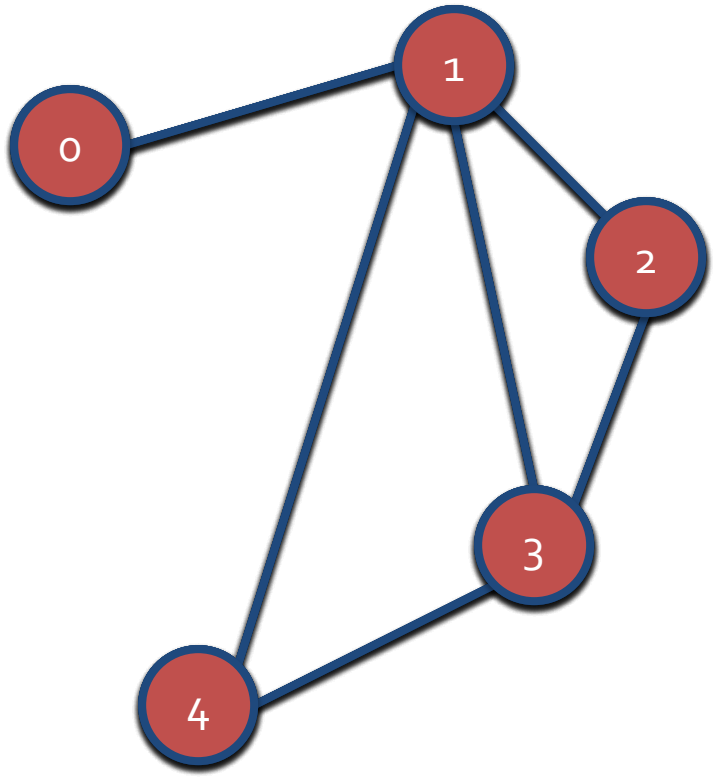


Depth First Search

Depth first search (DFS)

- Similar to a preorder traversal in a tree
- We want to visit every node once, going down as far as possible before backing up
- Issues:
  - No guarantee about ordering like a BST
  - Loops are a problem, how do we keep from repeating nodes?

DFS example



We might start at an arbitrary location.

What's a DFS look like that starts at node 4?

4, 1, 0, 2, 3

DFS pseudocode

- We use pseudocode a lot when describing graph operations, since the details depend on implementation choice
- Nodes all need some extra information, call it **number**
- Startup
  1. Set the number of all nodes to 0
  2. Pick an arbitrary node ***u*** and run DFS(***u***, 1)
- DFS(node ***v***, int ***i***)
  1. Set number(***v***) = ***i***++
  2. Do whatever other processing for ***v*** is necessary
  3. For each node ***u*** adjacent to ***v***

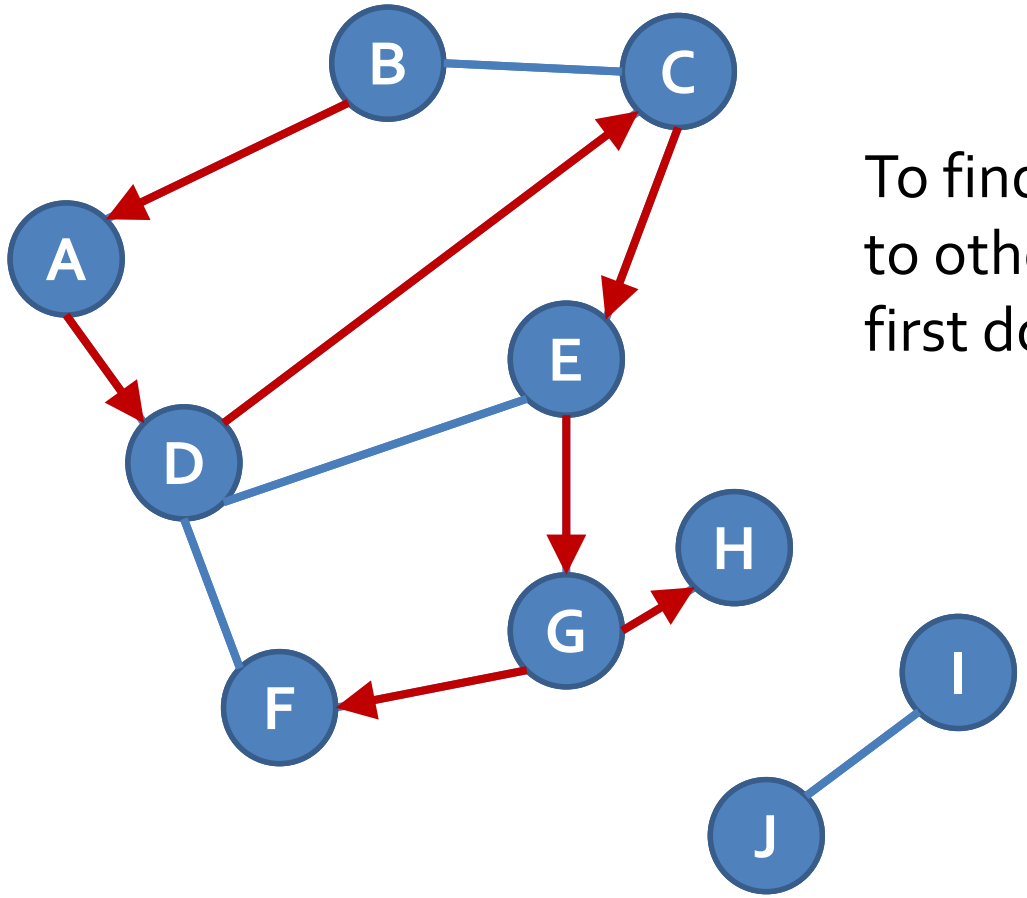
     If number(***u***) is 0

     DFS(***u***, ***i***)

Generating paths

- What if we wanted to find paths from node ***s*** to other nodes?
- We run DFS starting at ***s*** with an extra array of length |***V***| called ***edges***
- When we move from node u to node v, we set ***edges***[***v***] = ***u***
- Then, to find a path from ***s*** to ***t***, we backtrack by looking at ***edges***[***t***] and working backwards until we get to ***s***
- This approach will find a path if there is one, but it may not be the shortest path

Path example



To find paths from B to other nodes, we first do a DFS from B

Node	Edge From
A	B
B	
C	D
D	A
E	C
F	G
G	E
H	G
I	
J	

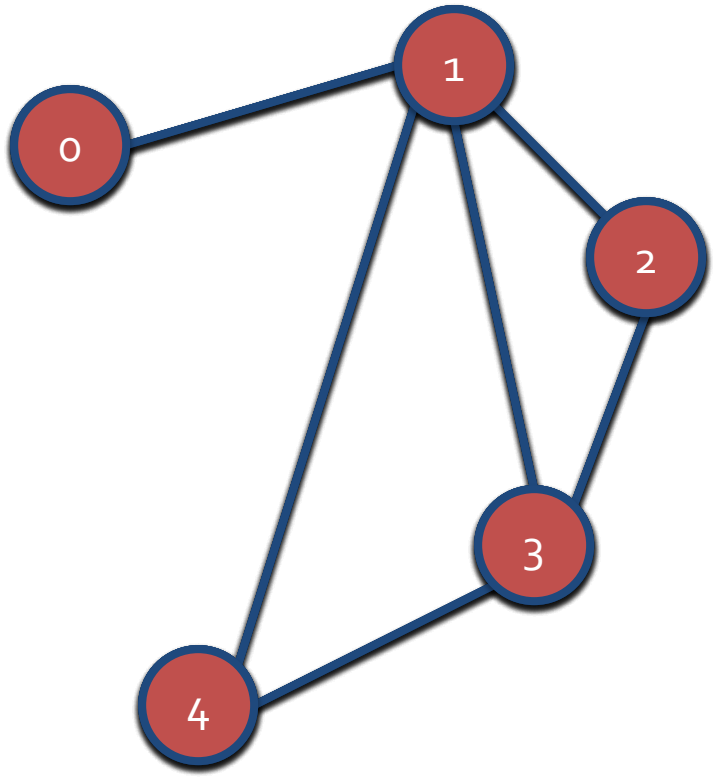
Working backwards, a path from **F** to **B** is: **F G E C D A B**
Thus, a path from **B** to **F** is: **B A D C E G F**

Breadth First Search

Breadth first search (BFS)

- Similar to a level order traversal in a tree
- We want to visit every node once, visiting all the neighbors of one node before moving on to their neighbors
- Similar issues to a DFS

BFS example



We might start at an arbitrary location.

What's a BFS look like that starts at node 4?

4, 1, 3, 0, 2

BFS pseudocode

- More pseudocode!
- Nodes all need some extra information, call it **number**
- BFS(node ***v***)
  1. Set the number of all nodes to 0
  2. Create queue ***q***
  3. Set ***i*** = 1
  4. number(***v***) = ***i***++
  5. ***q***.enqueue(***v***)
  6. While ***q*** is not empty
     a. ***v*** = ***q***.dequeue()
     b. Do whatever other processing for ***v*** is necessary
     c. For each node ***u*** adjacent to ***v***

        If number(***u***) is 0

        Set number(***u***) = ***i***++

        q.enqueue(***u***)

Running time

- Let $\boldsymbol{V}$ be the set of vertices and $\boldsymbol{E}$ be the set of edges
- Thus, $|\boldsymbol{V}|$ is the number of vertices and $|\boldsymbol{E}|$ is the number of edges
- If you are using adjacency lists then:
  - DFS is:
    - $O(|\boldsymbol{V}| + |\boldsymbol{E}|)$
  - BFS is:
    - $O(|\boldsymbol{V}| + |\boldsymbol{E}|)$
- If you are using an adjacency matrix then:
  - DFS is:
    - $O(|\boldsymbol{V}|^2)$
  - BFS is:
    - $O(|\boldsymbol{V}|^2)$

Quiz

Upcoming

Next time…

- Cycle detection
- Topological sort
- Connectivity
- Minimum spanning trees
- Shortest paths

Reminders

- Keep working on Project 3
- Finish Assignment 4
  - **Due Friday!**
- Read 4.3 and 4.4