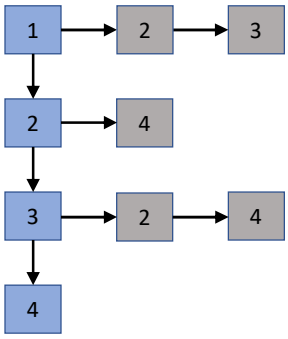
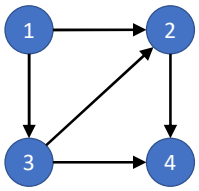
Adjacency list implementation #1

Linked list, where each node contains vertex label and linked list of adjacent vertex labels





1

Exercise

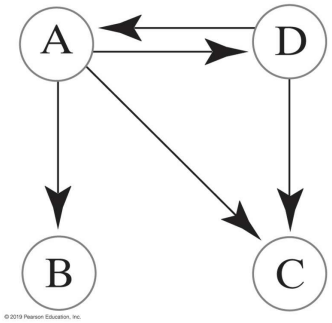
What are the adjacency lists that represent this graph?

A: ?

B: ?

C: ?

D: ?

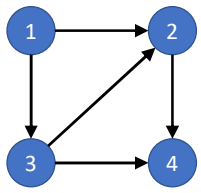


2

Adjacency "matrix"

- Given integer labels and bounded # of vertices…
- Maintain a 2D Boolean array **`b`**
- Invariant: element **`b[i][j]`** is **`true`** iff there is an edge from vertex **`i`** to vertex **`j`**

	0	1	2	3	4
0	F	F	F	F	F
1	F	F	**T**	**T**	F
2	F	F	F	F	**T**
3	F	F	**T**	F	**T**
4	F	F	F	F	F



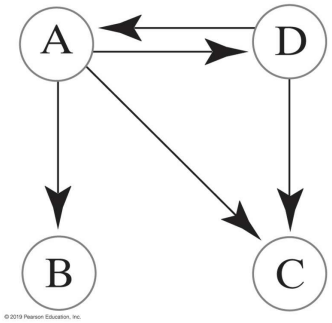
3

Exercise

What is the adjacency matrix that represents this graph?

(converting vertex labels to 0-based position in alphabet)

	0	1	2	3
0				
1				
2				
3				



4

Adjacency list *vs.* Adjacency matrix

List	Property	Matrix
	Space	
	Time to visit all edges	
	Time to find edge (v_1, v_2)	
	Better for	

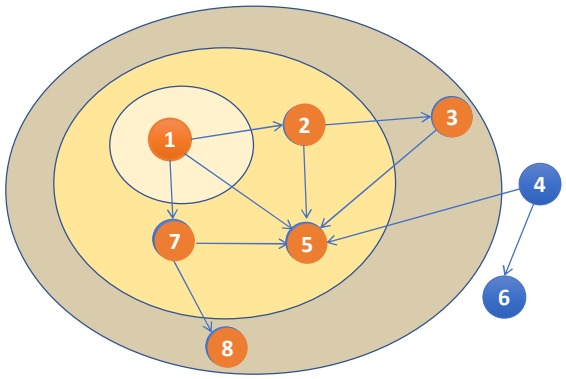
Max # edges = $|V|^2$

Sparse: $|E| \ll |V|^2$ Dense: $|E| \approx |V|^2$

5

BFS Algorithm

Key idea: Iteratively visit each unvisited "layer"



Many possible BFS orders depending on how nodes in layer are chosen:

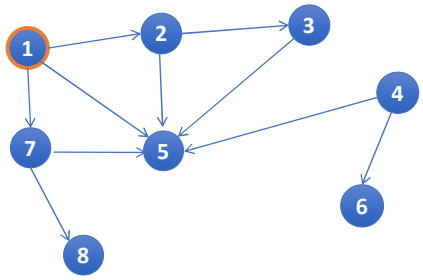
1, 2, 5, 7, 3, 8; 1, 7, 5, 2, 8, 3; 1, 5, 7, 2, 3, 8; …

(min) (max) (odds first)

6

BFS exercise

While frontier is not empty:
Remove next vertex from frontier
For each of its neighbors:
If neighbor has not been discovered:
Mark neighbor as discovered and add to frontier
Neighbor's layer is current layer + 1



Frontier:

1							

Vertex	1	2	3	4	5	6	7	8
Discovered	Y							
Layer	0							

7

Time complexity of BFS (exercise)

```
while (!frontier.isEmpty()) {
  Vertex v = frontier.remove();
  for (Vertex n : v.successors) {
    if (!n.discovered) {
      n.discovered = true;
      frontier.add(n);
    }
  }
}
```

8