CS202 – Data Structures

LECTURE-18

Graphs

Graphs and their representation

Dr. Maryam Abdul Ghafoor
Assistant Professor
Department of Computer Science, SBASSE

Agenda

- Graph Terminology
- Representing Graphs
- Graph Trversals

Concept Check!

What are the possible use cases of Priority Queue?

What are the possible use cases of Priority Queues?

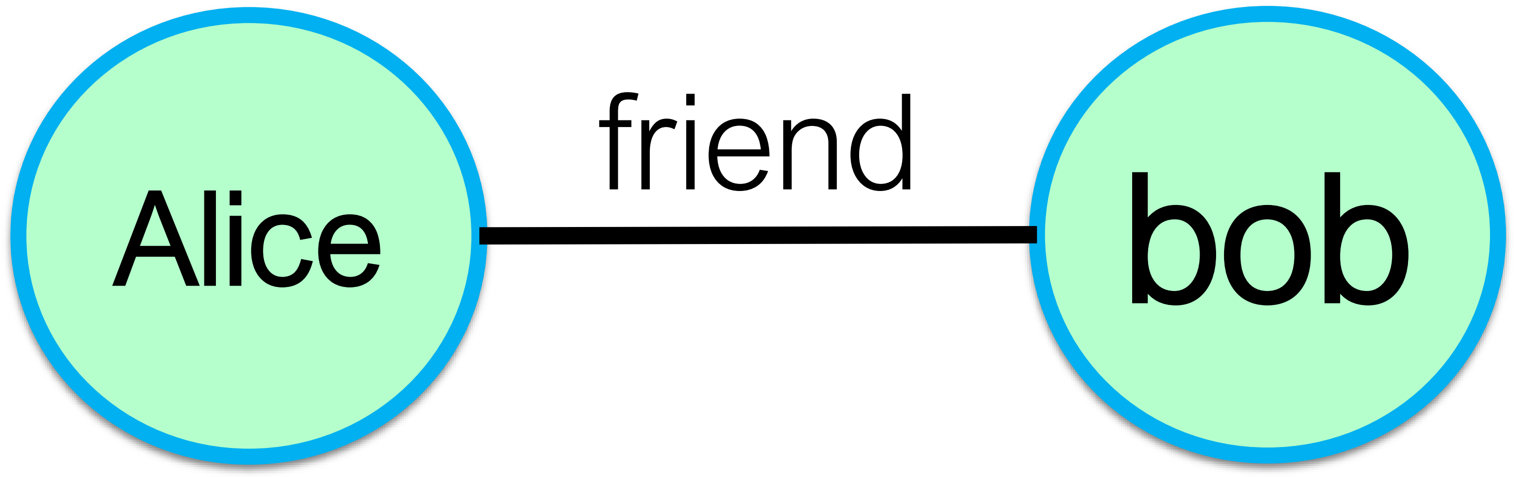
Nobody has responded yet.

Hang tight! Responses are coming in.

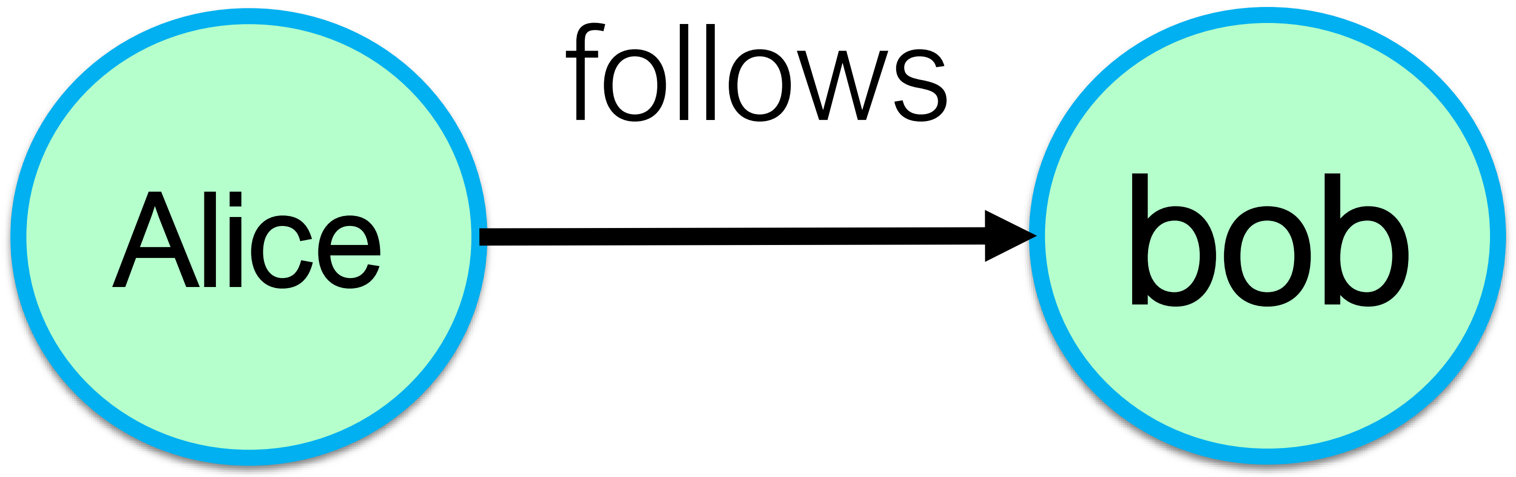
Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Graph – Basic Terminology

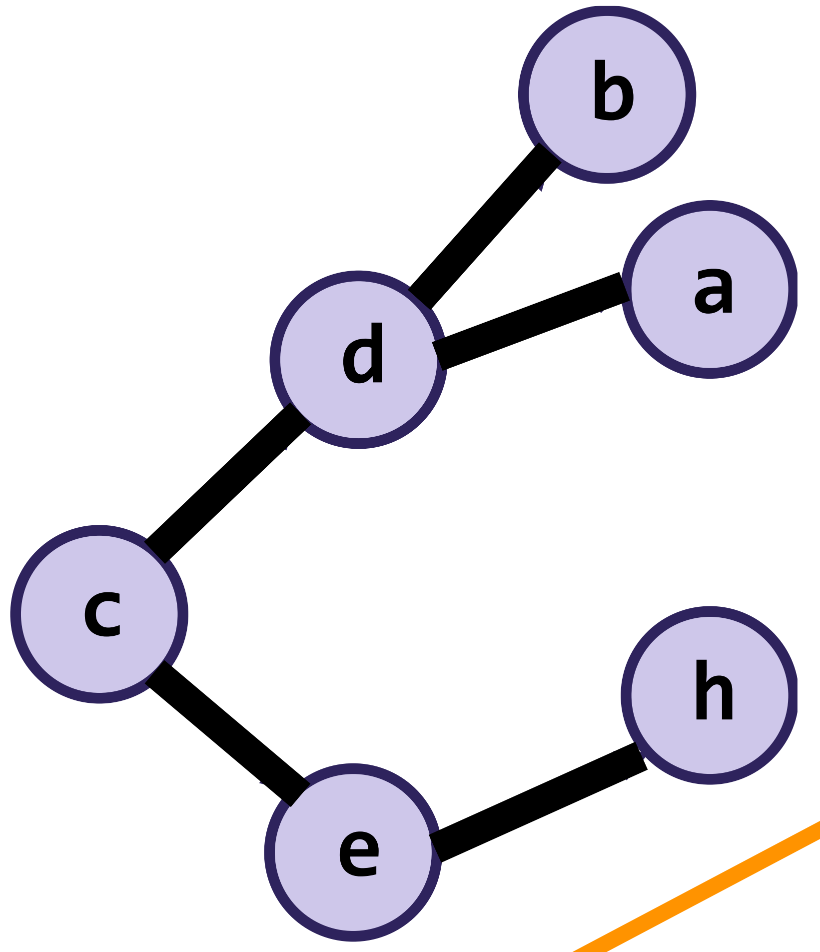
- Graph is a set of vertices and edges $G = (V, E)$
- Vertex (V)
  - A node is also known as the vertex
  - V = {Alice, Bob}



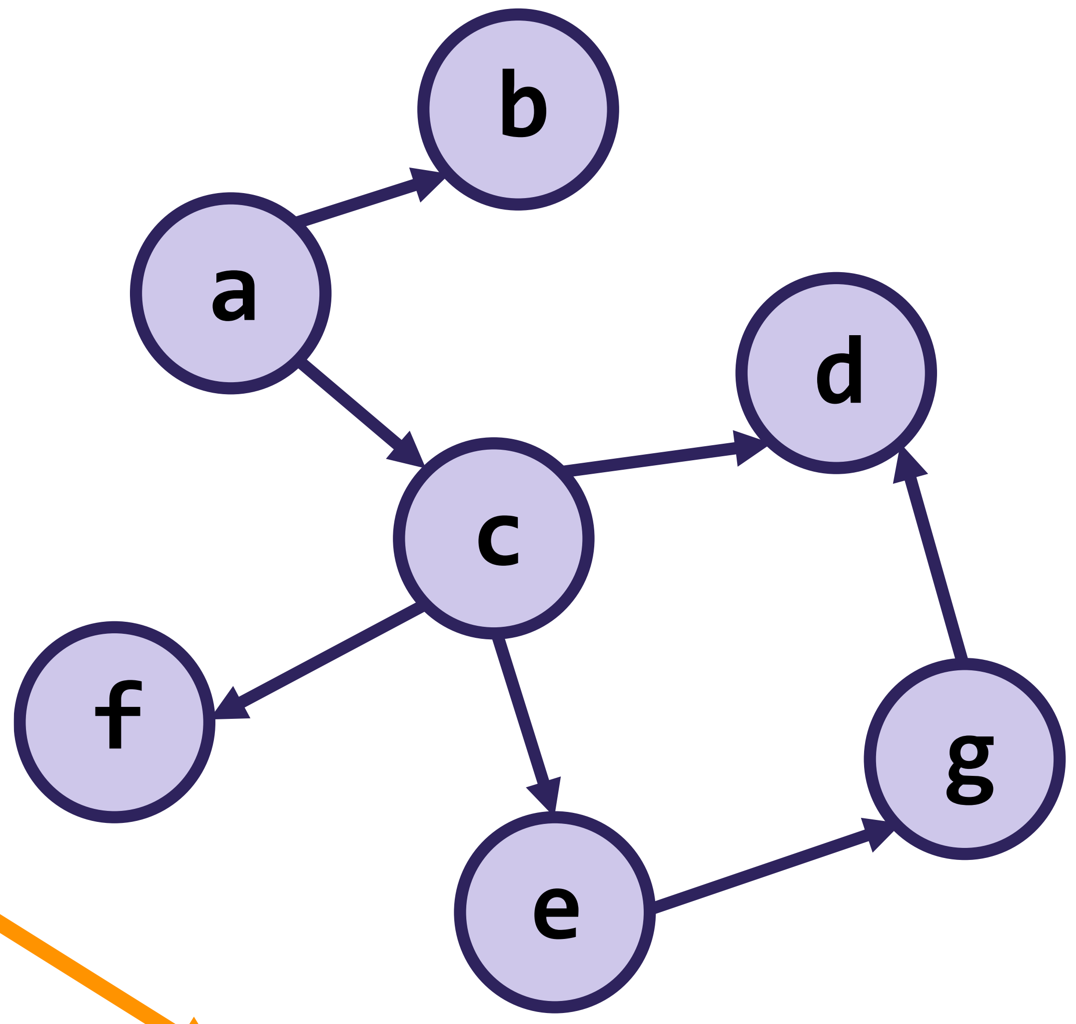
- Edges (E)
  - Connectors for vertices
  - E = {(Alice, Bob)}
  - Can be directed or undirected



Graph Vocabulary



Graph Direction



Undirected Graphs

Edges have no direction and are two way

E = {(e, c), (c, e), (d, b)…..}

Directed graphs (Diagraphs)

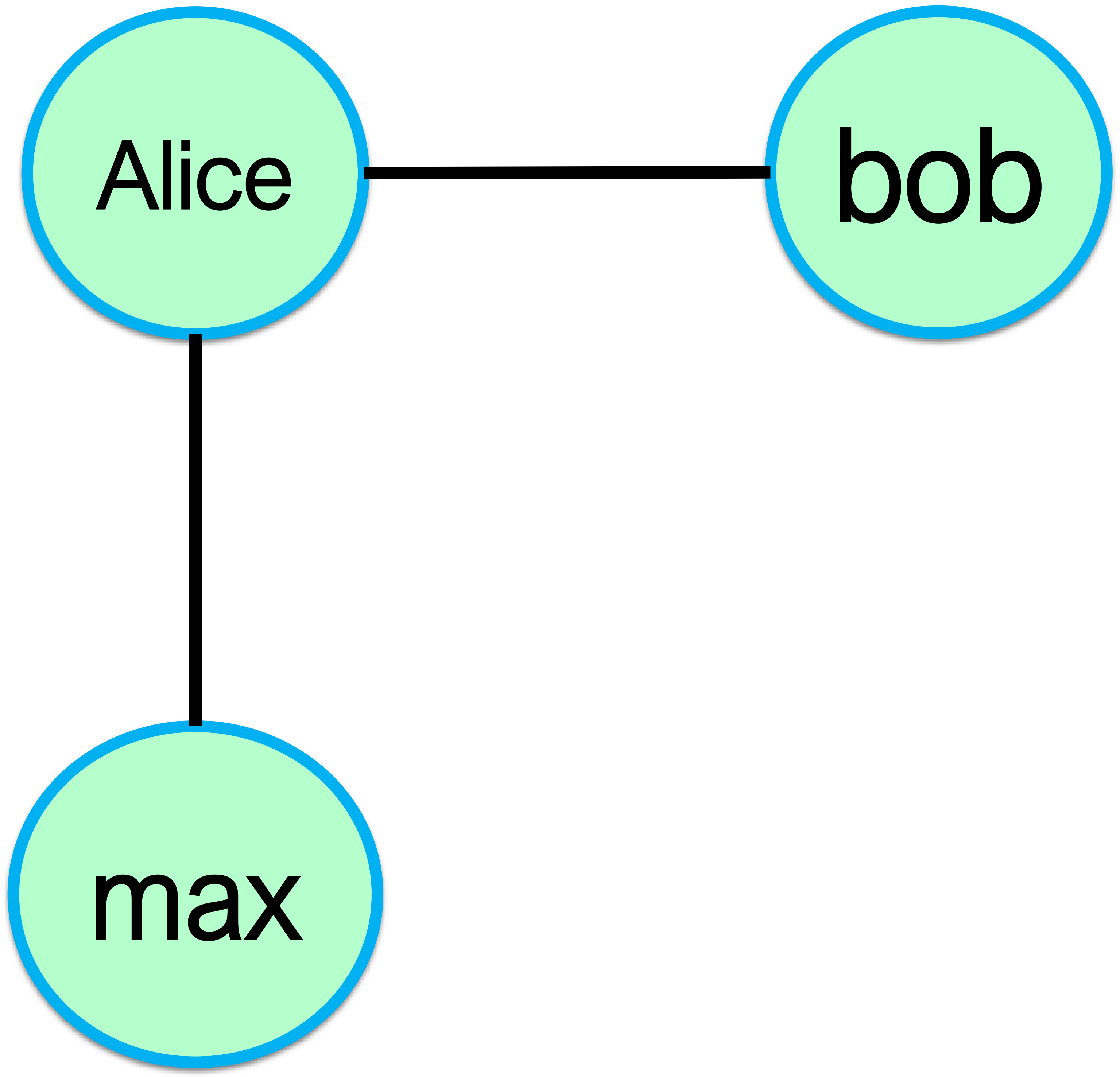
Edges have direction and are one way

E = {(a, b), (a, c), (c, f), (c, e), (c, d), (g, d)}

Basic Terminology

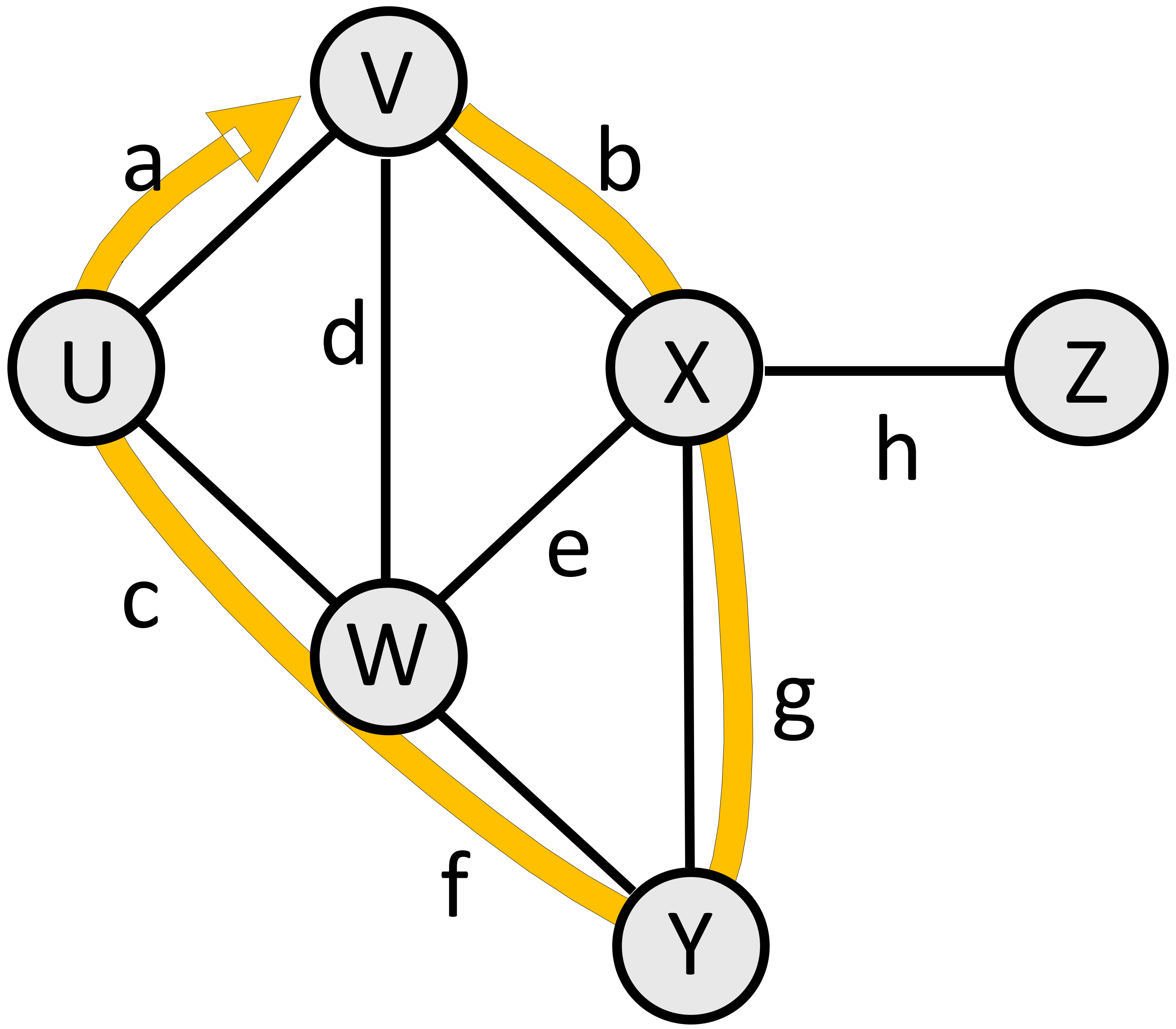
- Adjacent nodes (aka **neighbors**)
  - Two nodes are **adjacent**, if an edge connects them together.

- **Path** is a particular permutation of edges in the graph.
  - For example, Max – Alice – Bob



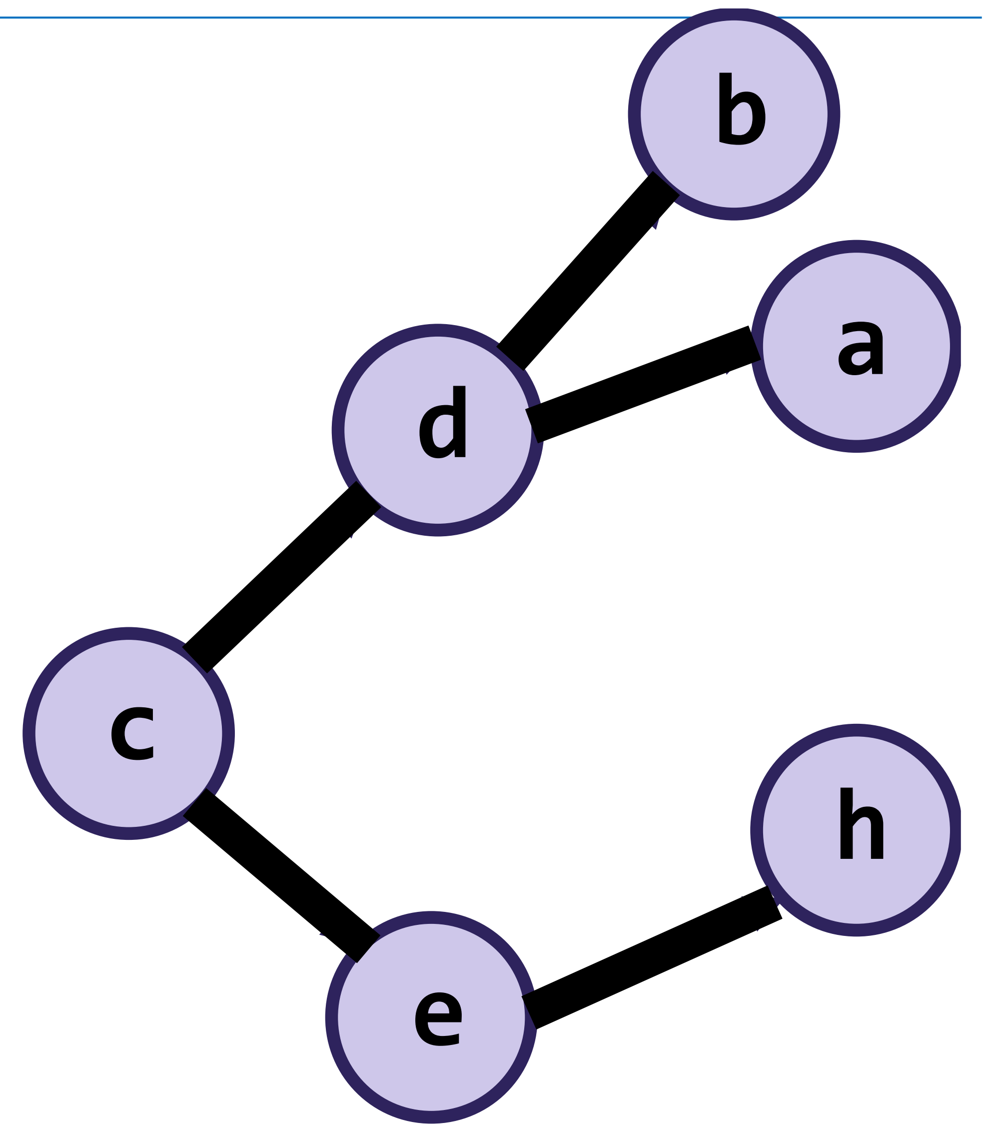
Cycles in Graphs

- Graphs can have **loops** in them, a graph with a loop is known as cyclic graph.
- A **cycle** is a path whose first and last vertices are the same
  - Ex: {V, X, Y, W, U, V}
- **Acyclic graph:** One that **does not** contain any cycles

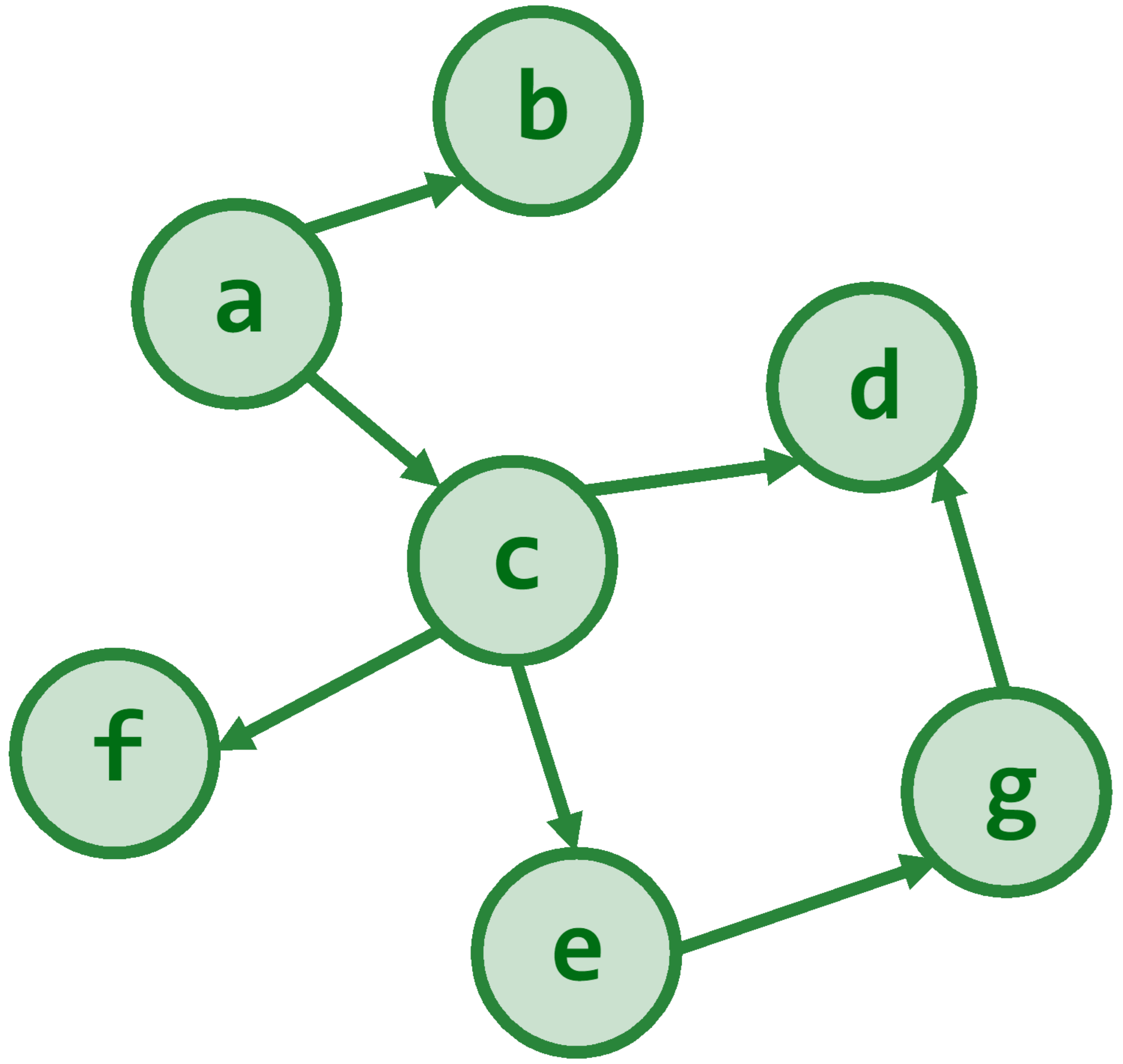


Degree of a Vertex

- Degree of a Vertex
  - Degree is the number of edges incident on a vertex deg(e) = 2

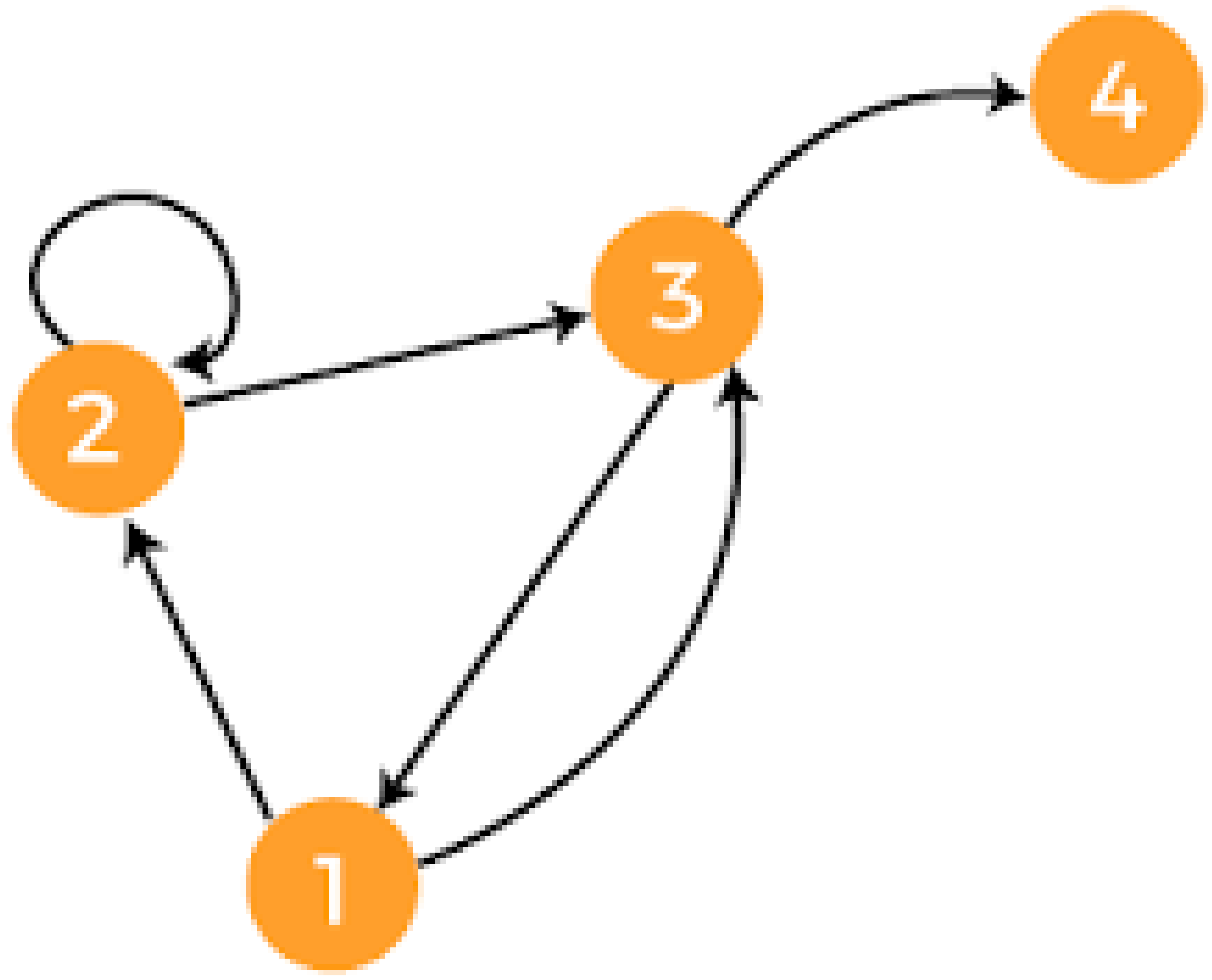


- Degree of a directed graph (diagraph)
  - In-degree is the number of edges directed towards a vertex, deg(e) = 1
  - Out-degree is the number of edges directed aways from the vertex, deg(a) = 2



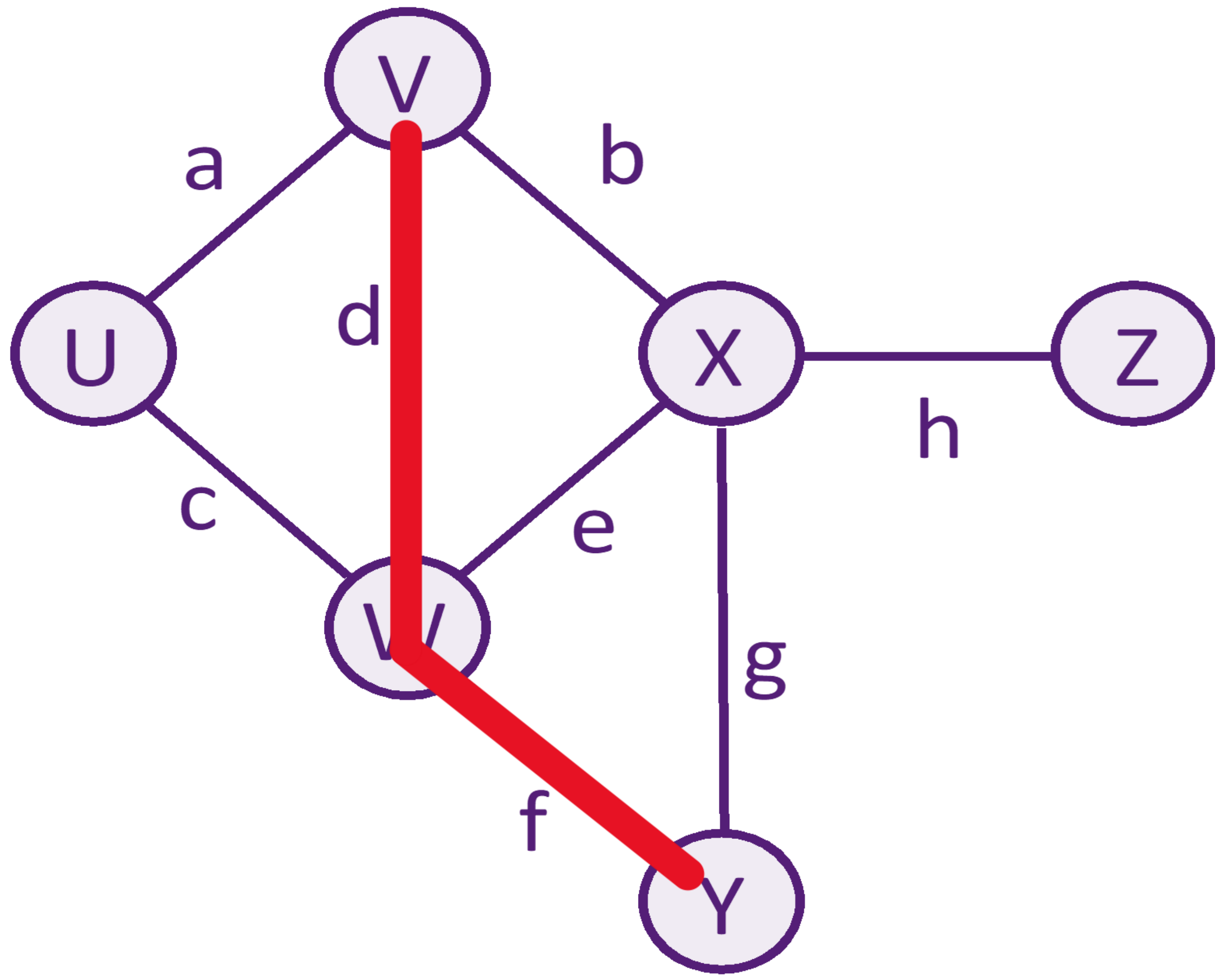
Self-edges in Graphs

- An edge from the vertex v to v(itself) is a self-edge



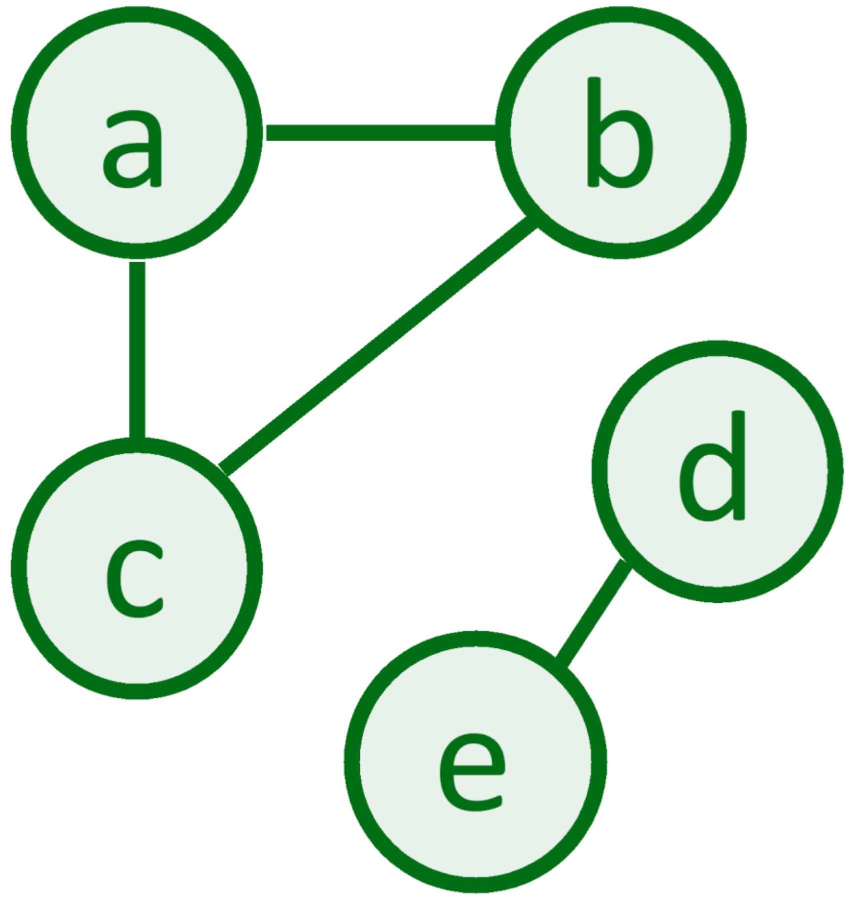
Self-edge: An edge of the from (2,2)

Reachability and Connectedness in Graph



Reachable: Vertex Y is reachable from V if a path exists from Y to V

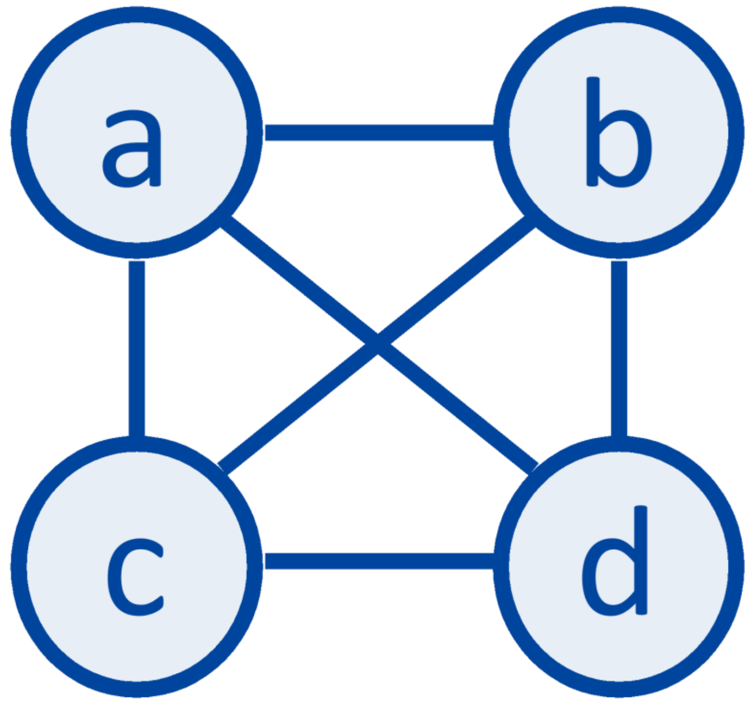
Connected: If every vertex is reachable from every other vertex



Strongly connected: if every vertex is reachable from every other vertex in diagraph

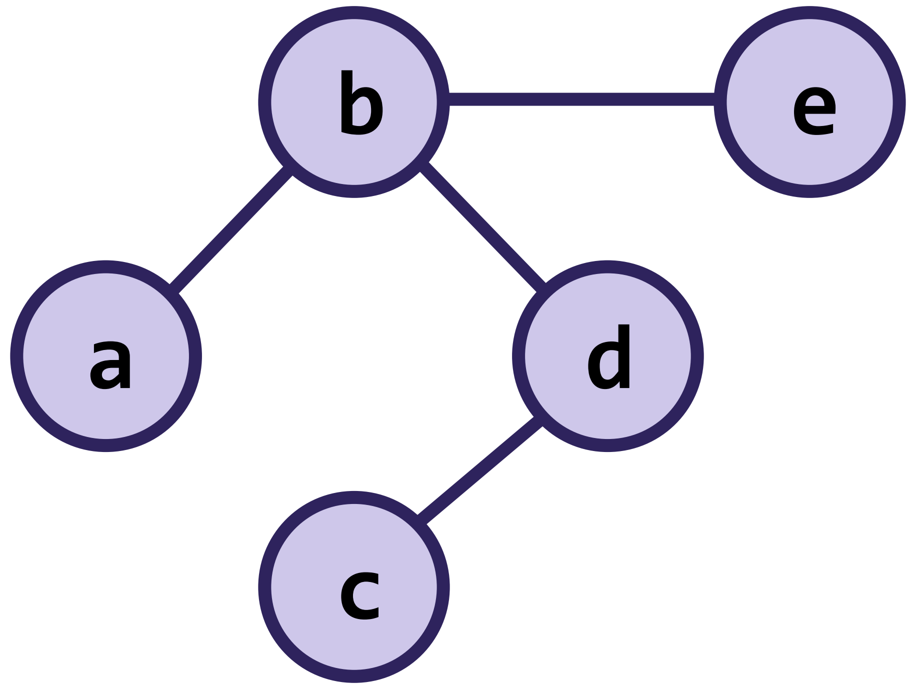
Complete Graph

Complete: If every vertex has a direct edge to every other

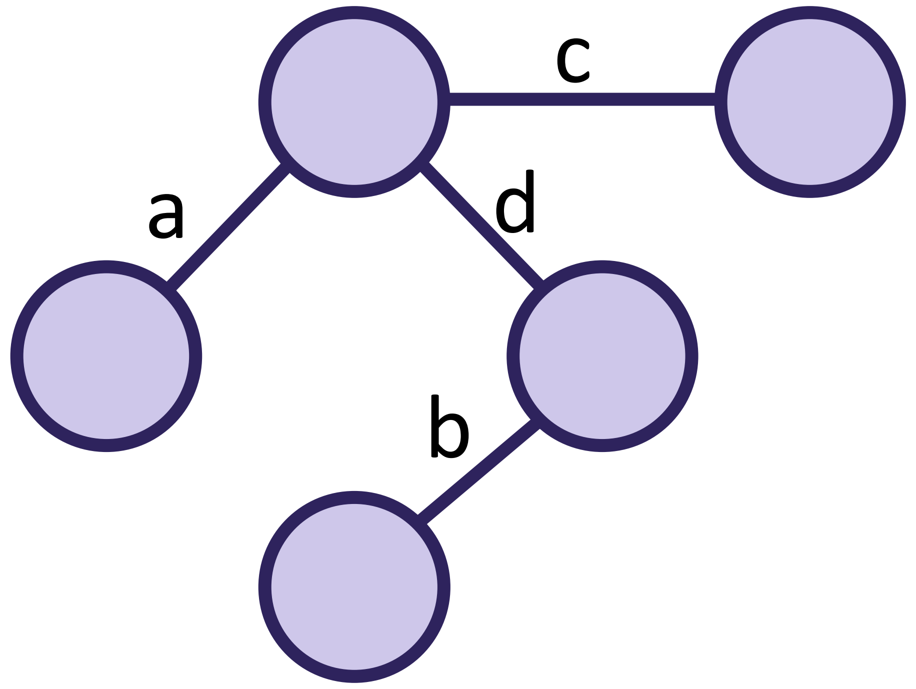


Labeled and Weighted Graphs

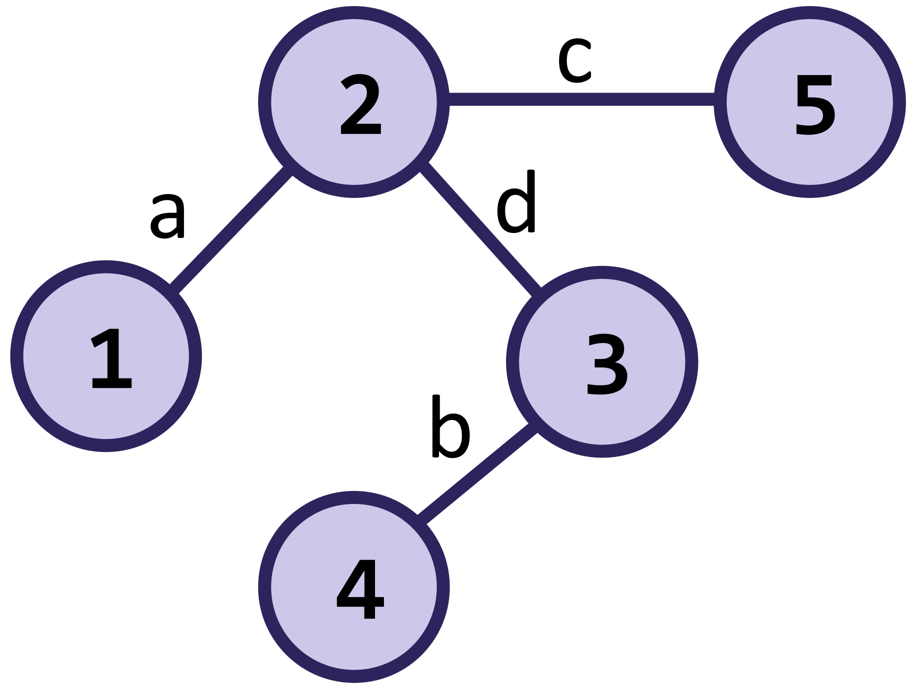
Vertex Labels



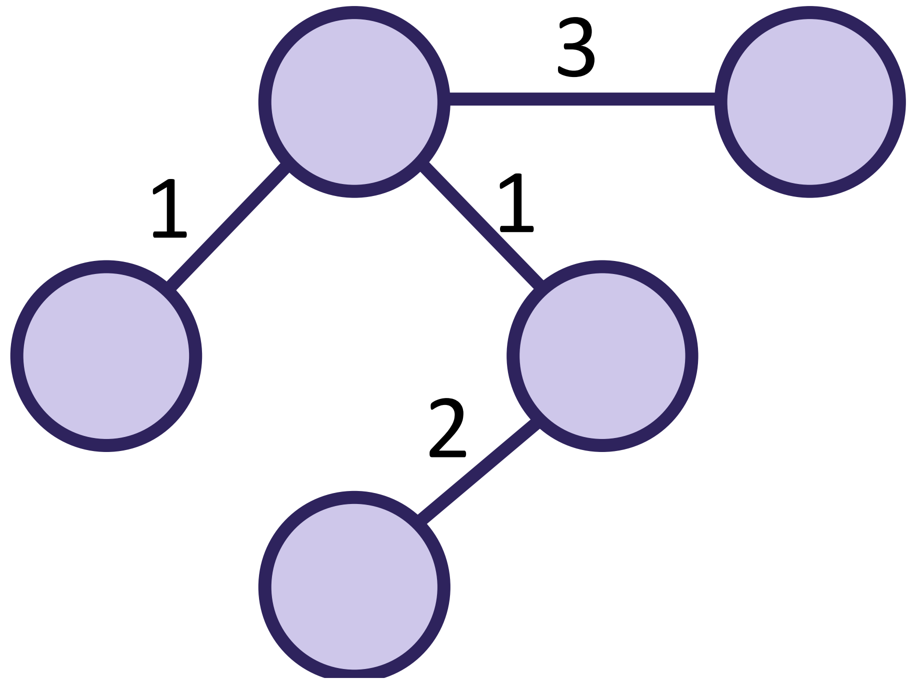
Edge Labels



Vertex & Edge Labels

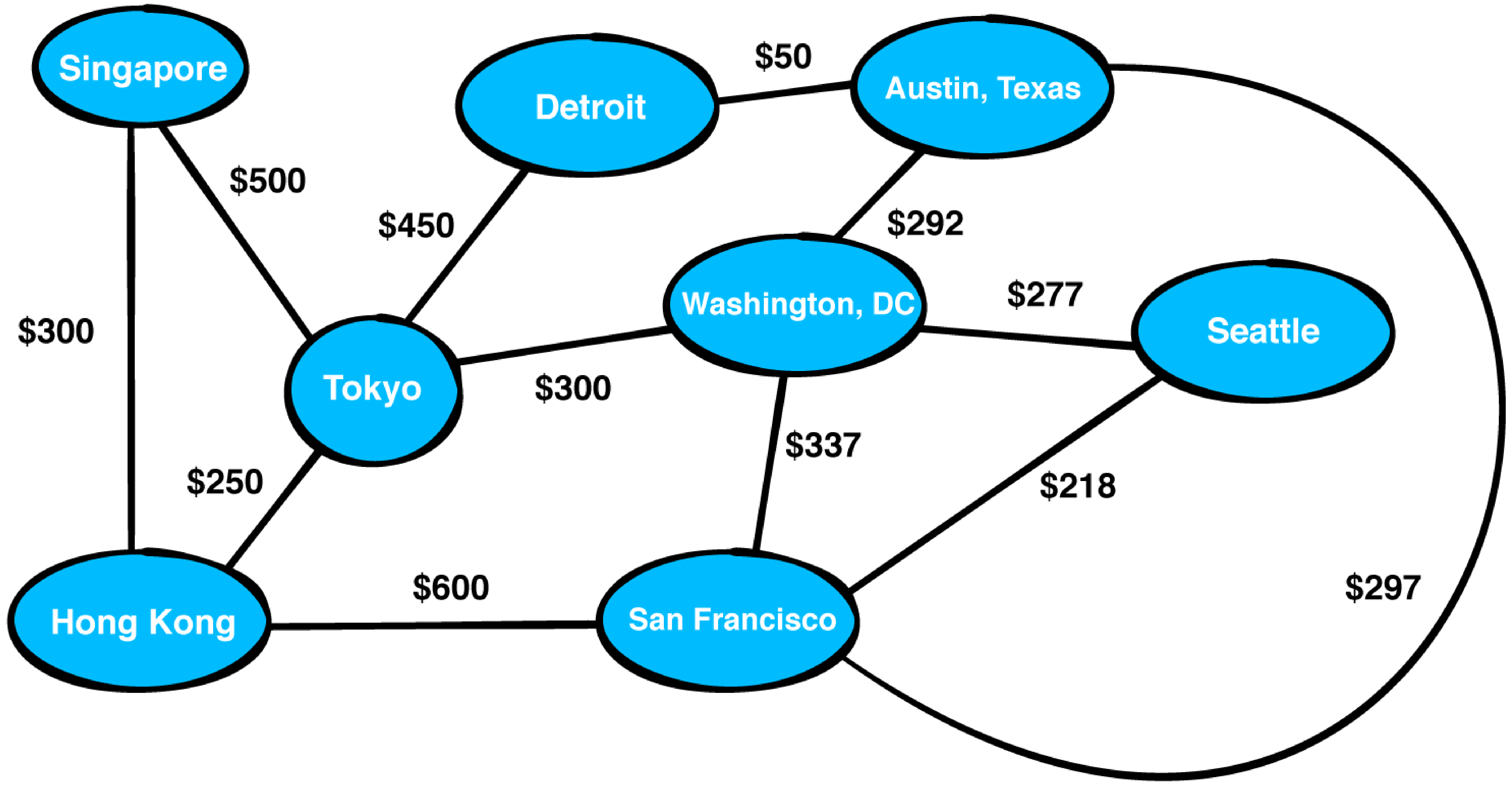


Numeric Edge Labels
(**Edge Weights**)



Weighted Graphs

- Weight represents the **cost** associated with a given edge
  - Edges in an unweighted graph have equal weights (e.g., all 0, or all 1)



Can the weights be negative?

Social Friendship Graph – Meta (G)

- Vertices?
- What defines an edge?

Directed or undirected?

Cyclic or Acyclic?

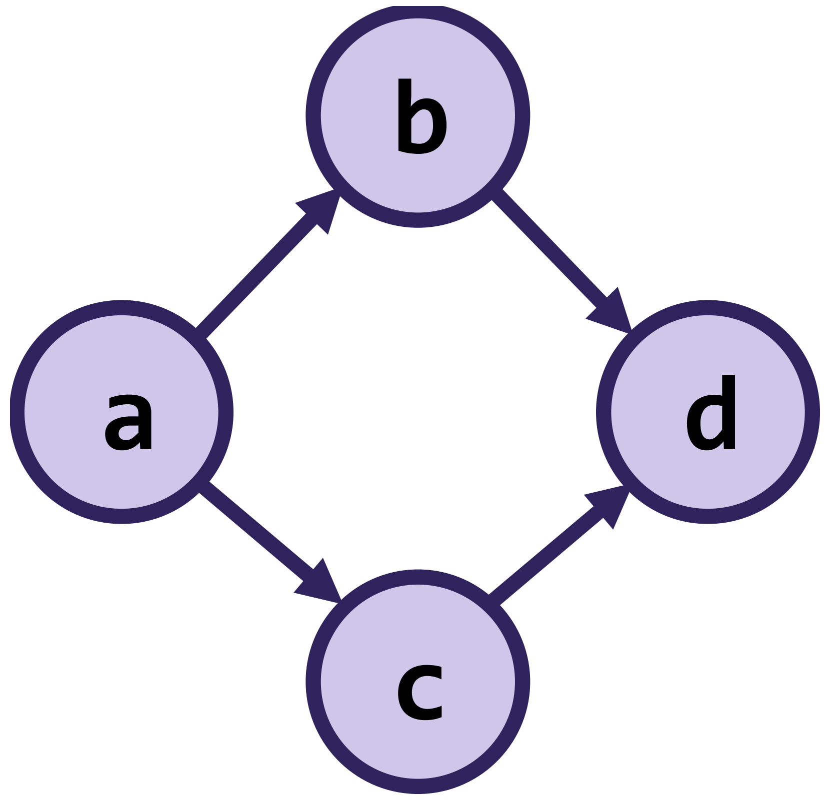
Self-Edge?

Connected Graph?

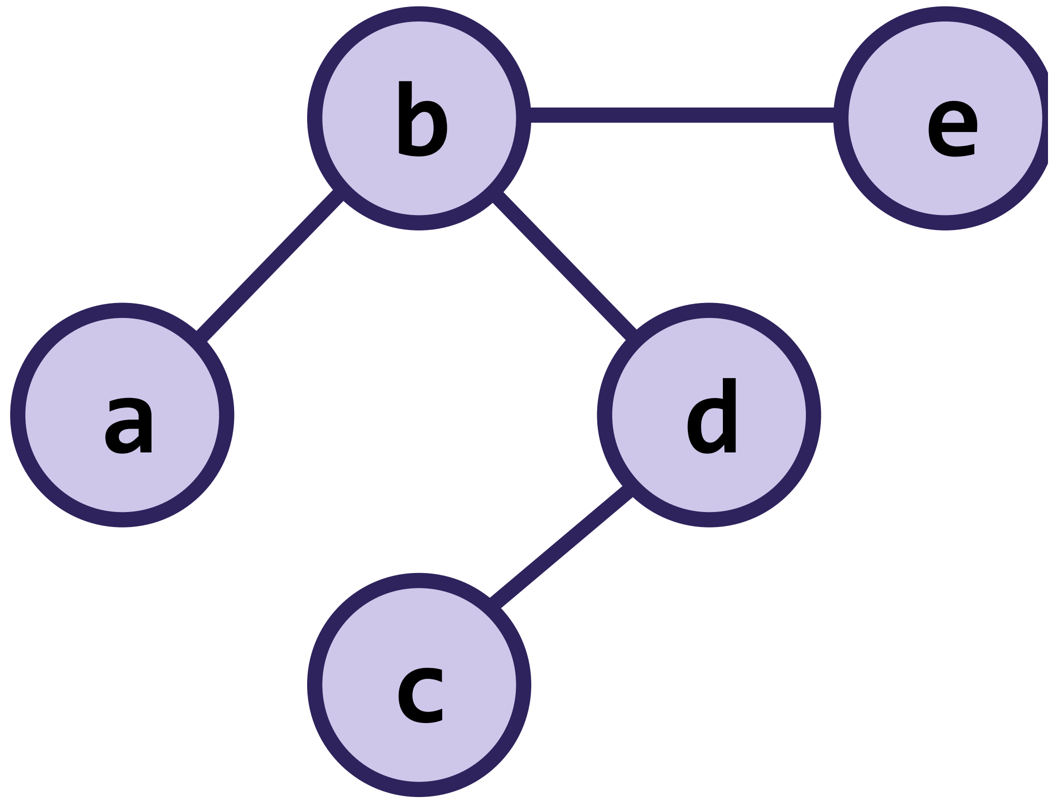
Weighted or unweighted?

Which of these are Acyclic Graphs?

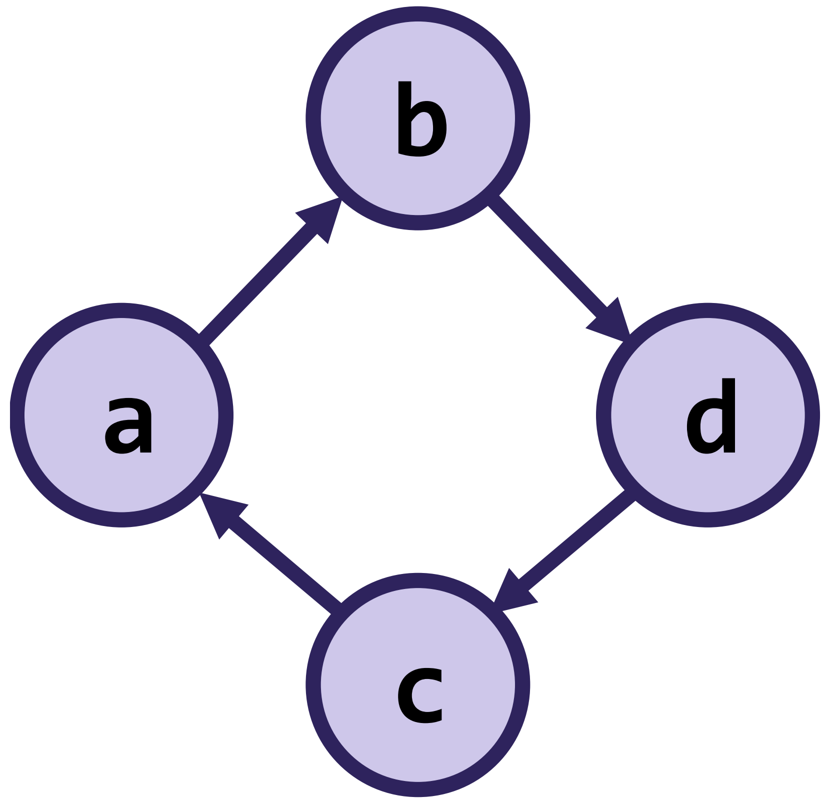
(A)



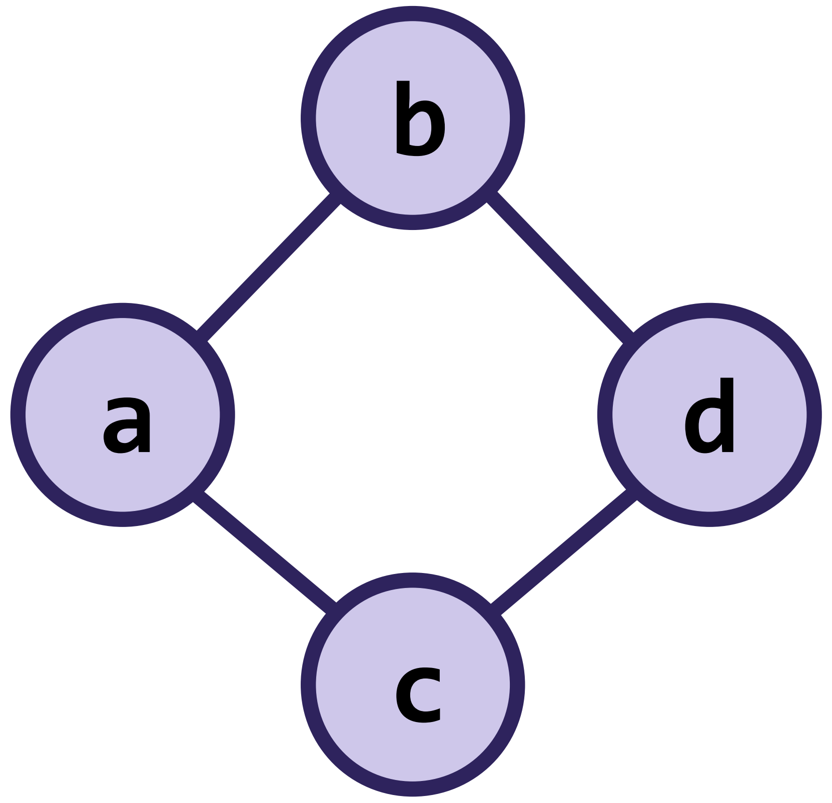
(B)



(C)

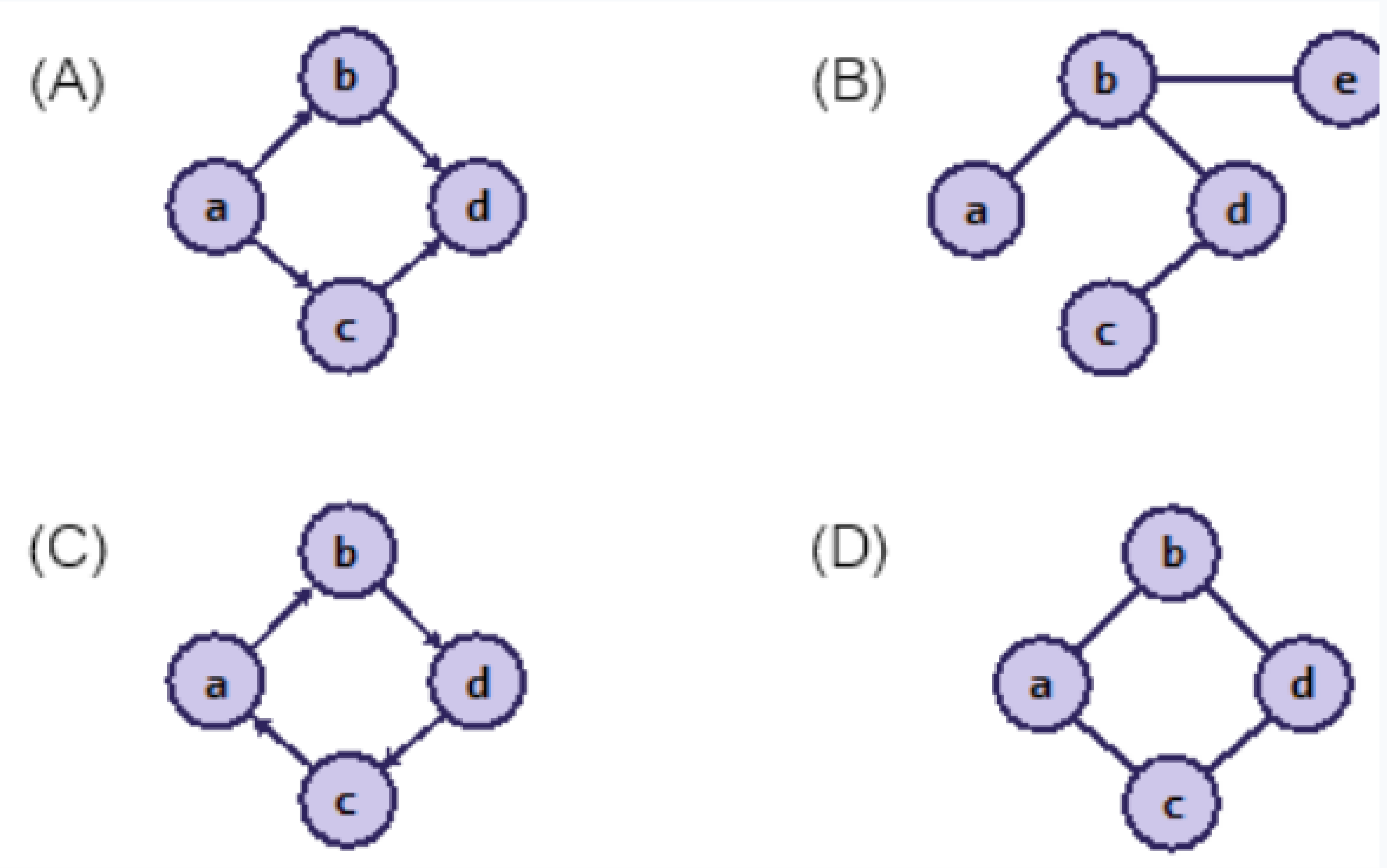


(D)



Which of the following are acyclic graphs?

✓✓ 0



A

0%

B

0%

C

0%

D

0%

What is the maximum number of edges in a diagraph with |V| vertices and |E| edges?

What is the maximum number of edges in a diagraph with |V| vertices and |E| edges?

|V|

0%

|V+E|

0%

|V^2|

0%

2V

0%

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

What is the minimum number of edges in undirected connected graph with |V| vertices and |E| edges?

What is the minimum number of edges in undirected connected graph with |V| vertices and |E| edges?

|V|

0%

|V-1|

0%

|V^2|

0%

2V

0%

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

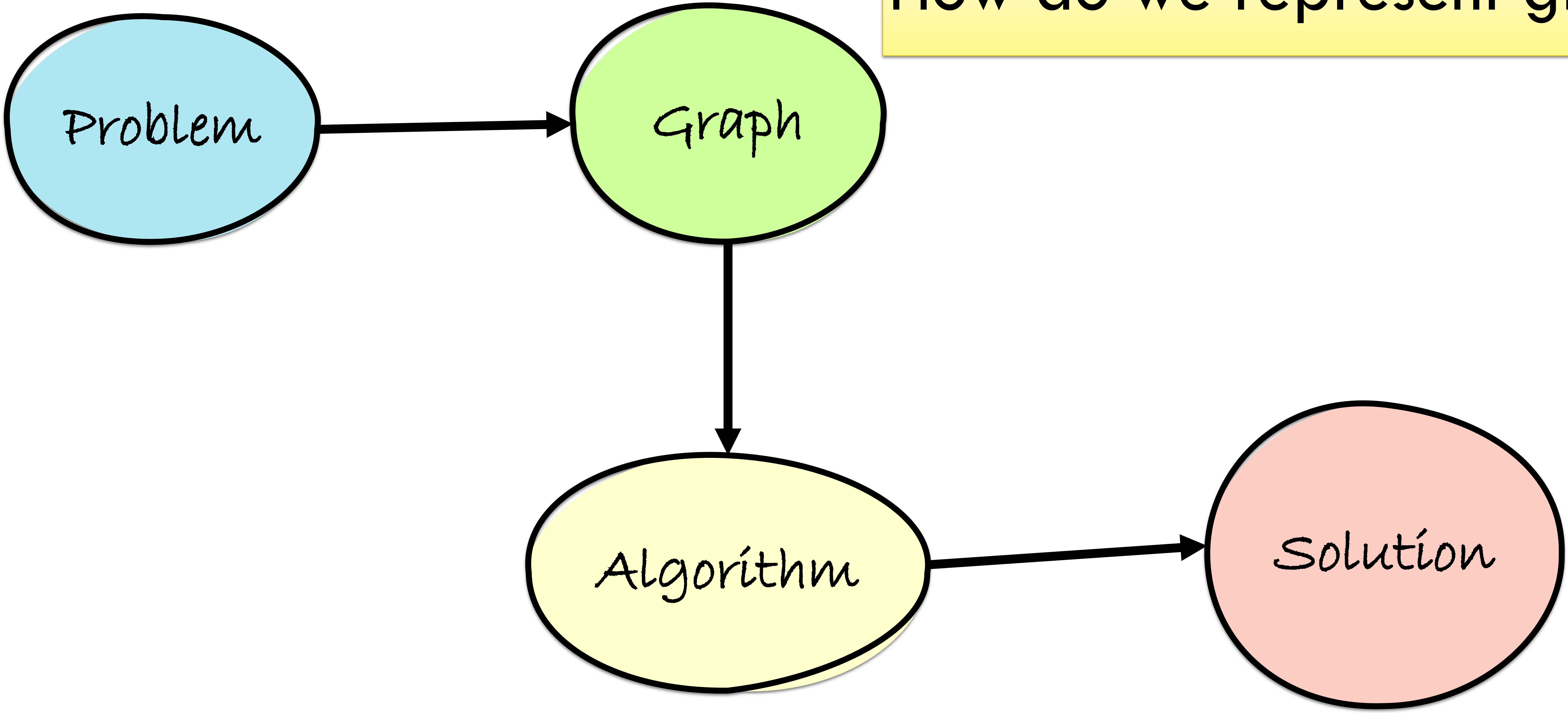
Edges in Graphs

- **Maximum** number of edges in any graph is in $O(|V|^2)$
  - Large storage needed if $|V|$ is large

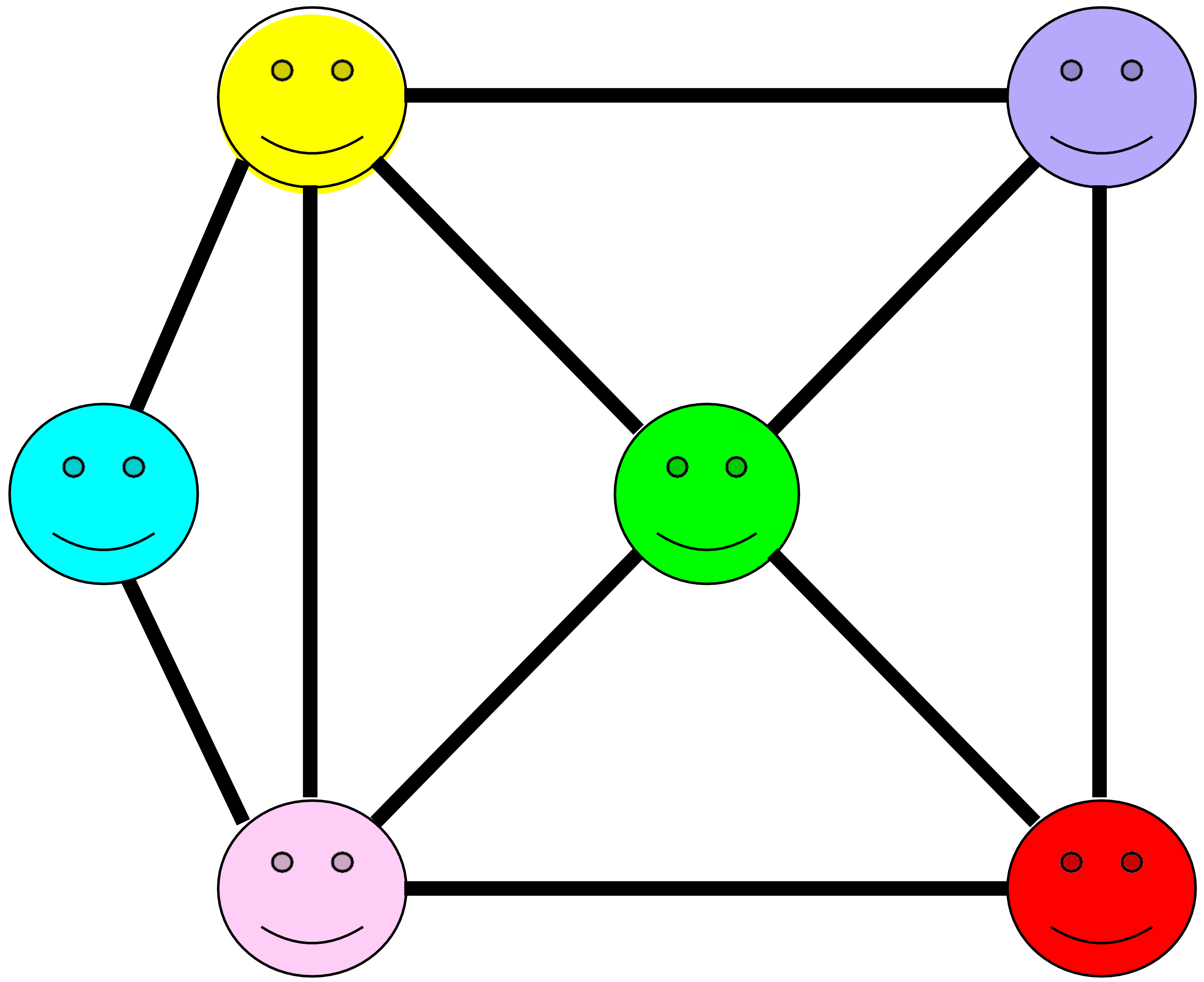
- However, in many graphs the number of edges are much smaller than the maximum number of edges
  - Such graphs are known as sparse graphs
    - Even though no one definition exists for sparsity, but usually graphs for which $|E|$ is in $O(|V|)$ are called **sparse graphs**

Graphs in Problem Solving

How do we represent graphs?



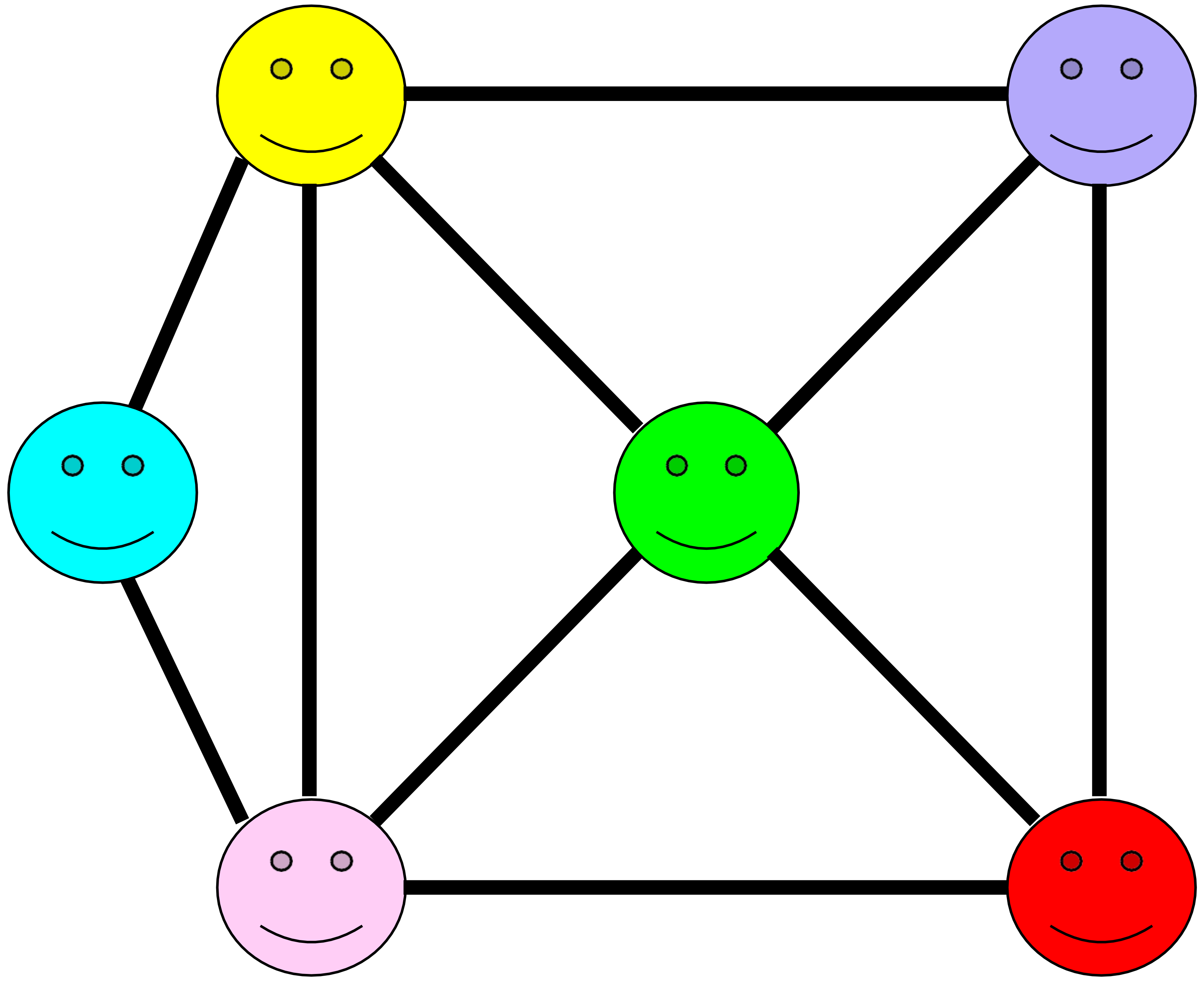
Representing Graphs: Adjacency Matrix (AM)



	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

$$M[i][j] = \begin{cases} 1, & \text{if } (i,j) \text{ is an edge in } G \\ 0 & \text{otherwise} \end{cases}$$

Representing Graphs: Adjacency Matrix (AM)

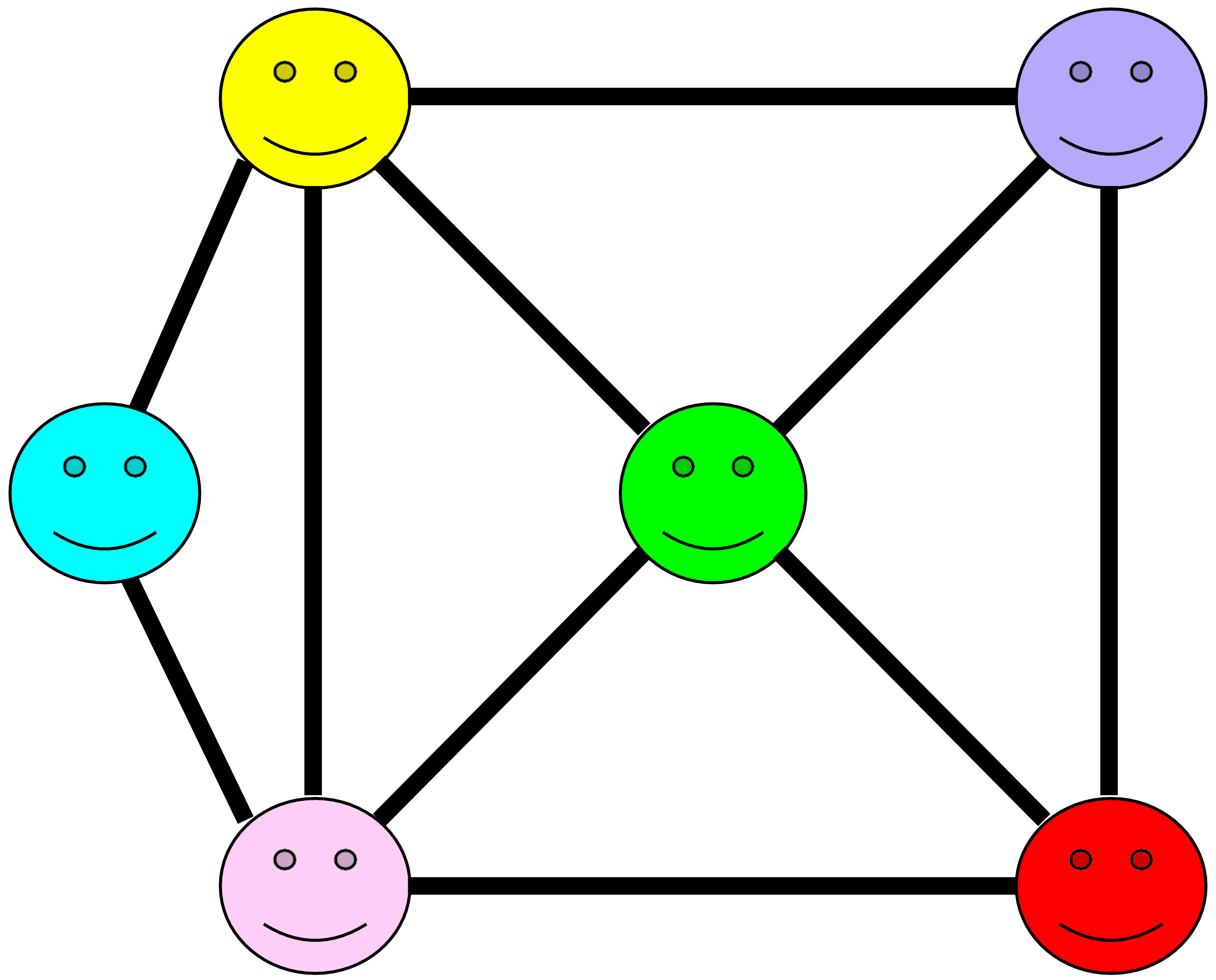


	0	1	2	3	4	5
0	0	1	1	0	0	0
1	1	0	1	1	1	0
2	1	1	0	1	0	1
3	0	1	1	0	1	1
4	0	1	0	1	0	1
5	0	0	1	1	1	0

What is the amount of memory used in AMs?

- Space Complexity: $\boldsymbol{O}(|\boldsymbol{V}|^2)$
  - $O(|V|)$: # of slots in the array, one for each vertex
    - Each slot will store $O(|V|)$ edges

Representing Graphs: Adjacency Matrix (AM)



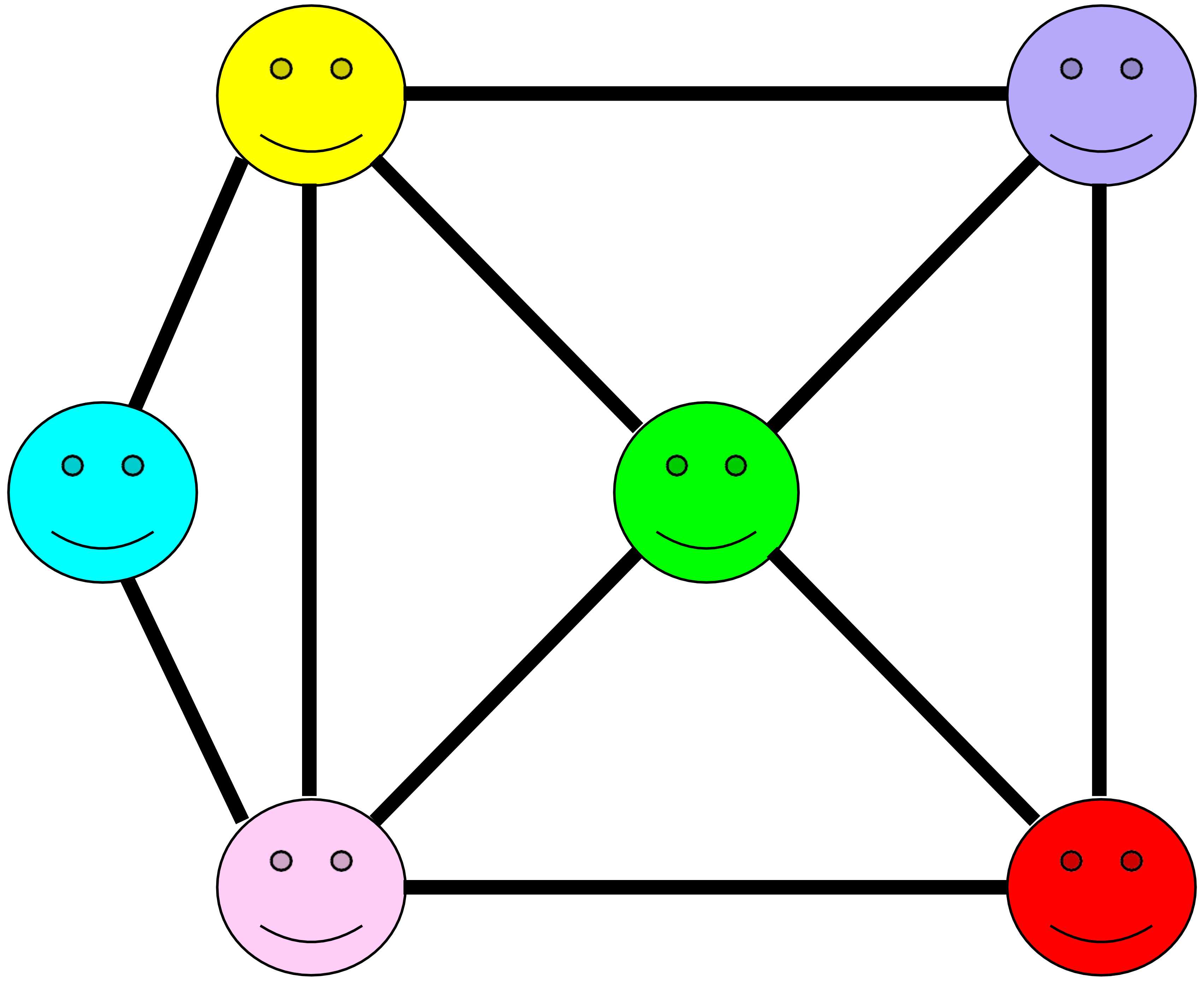
	0	1	2	3	4	5
0	0	1	1	0	0	0
1	1	0	1	1	1	0
2	1	1	0	1	0	1
3	0	1	1	0	1	1
4	0	1	0	1	0	1
5	0	0	1	1	1	0

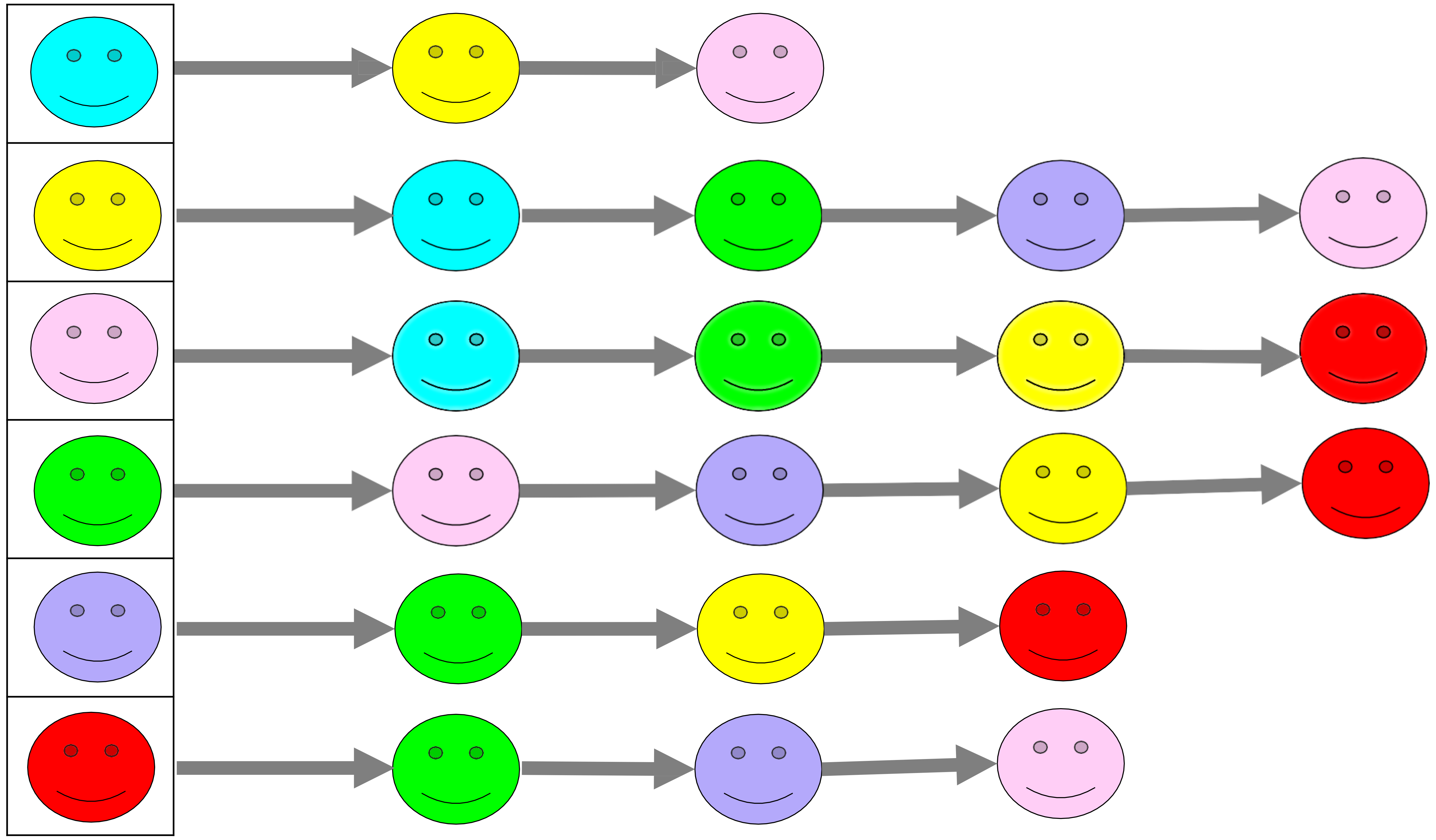
Properties:

- Symmetric
- Fast Lookup(Search)
- Use excessive space (good for dense graph)

How can we encode graphs more compactly?

Representing Graphs: Adjacency List (AL)





- A collection of linked lists, one for each vertex
- Each list stores neighboring vertices that are adjacent to the vertex

What is the amount of memory used in ALs?

- $\boldsymbol{O(|V| + |E|)}$
  - $O(|V|)$: # of slots in the array, one for each vertex
  - $O(|E|)$: # of linked list nodes in the entire AL

Adjacency Matrix Implementation

```
class Graph {
 private:
    int numVertices;
    vector<vector<int>> adjMatrix;
 public:
      Graph(int nVertices) {
        numVertices = nVertices;
        adjMatrix.resize(numVertices, vector<int>(numVertices, 0));
    }
      void addEdge(int i, int j) {
        adjMatrix[i][j] = 1;
        adjMatrix[j][i] = 1;
    }
};
```

Adjacency List – Implementation

```
class Graph {
 private:
    int numVertices;
    vector<vector<int>> adjMatrix;
 public:
      Graph(int nVertices) {
        numVertices = nVertices;
        adjMatrix.resize(numVertices);
    }
      void addEdge(int i, int j) {
        adjMatrix[i].push_back(j);
        adjMatrix[j].push_back(i);
    }
};
```

Exercise

Assume that vertices are numbered from 1 to 7 in a binary heap. Please draw it. Give an equivalent adjacency-matrix representation.

Concept Check!

What is the out-degree of vertex 2 and in-degree of vertex 1?

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

What is the out-degree of the node 2 and in-degree of node 1?

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

2, 0 — 0%

1, 3 — 0%

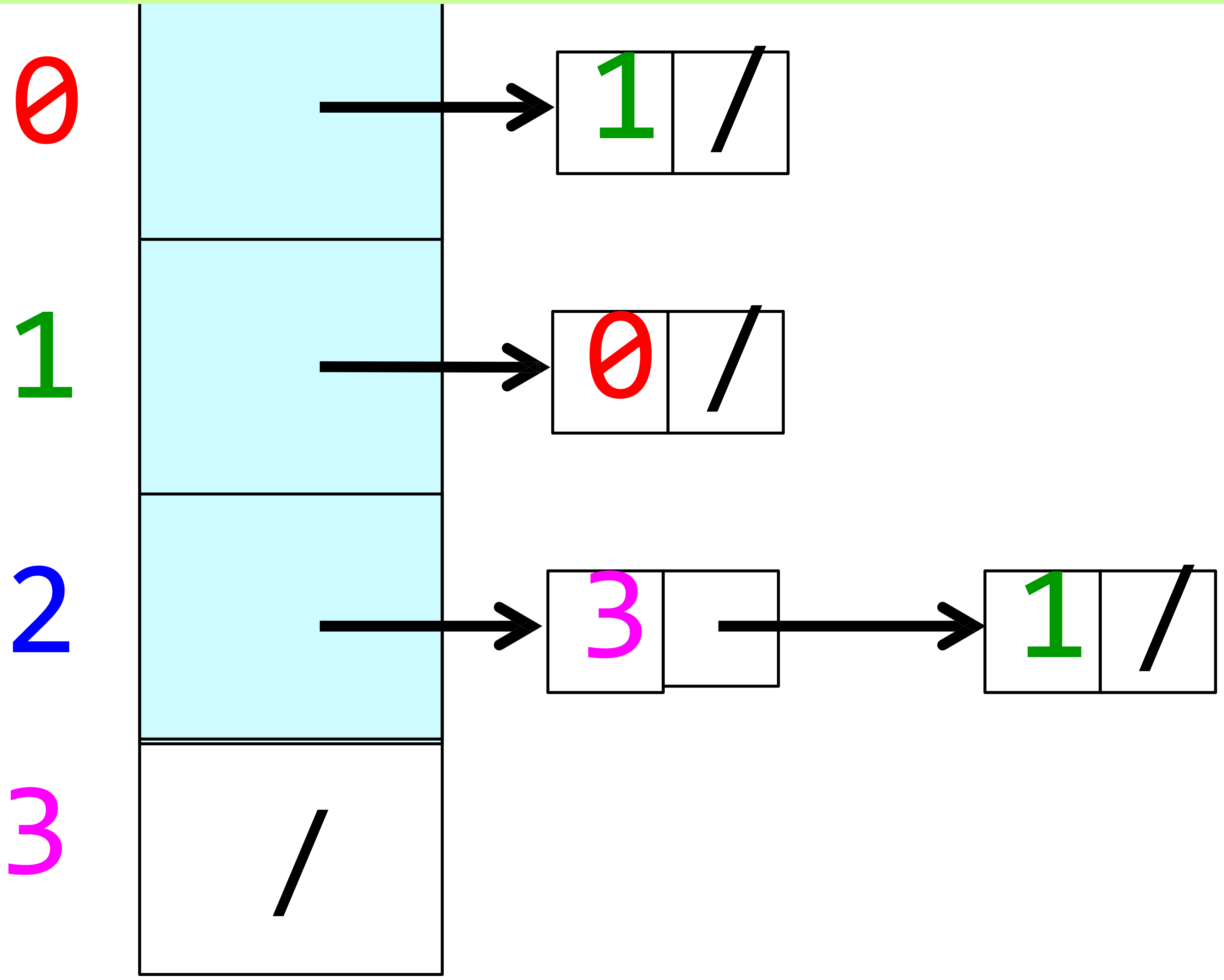
2, 3 — 0%

3, 2 — 0%

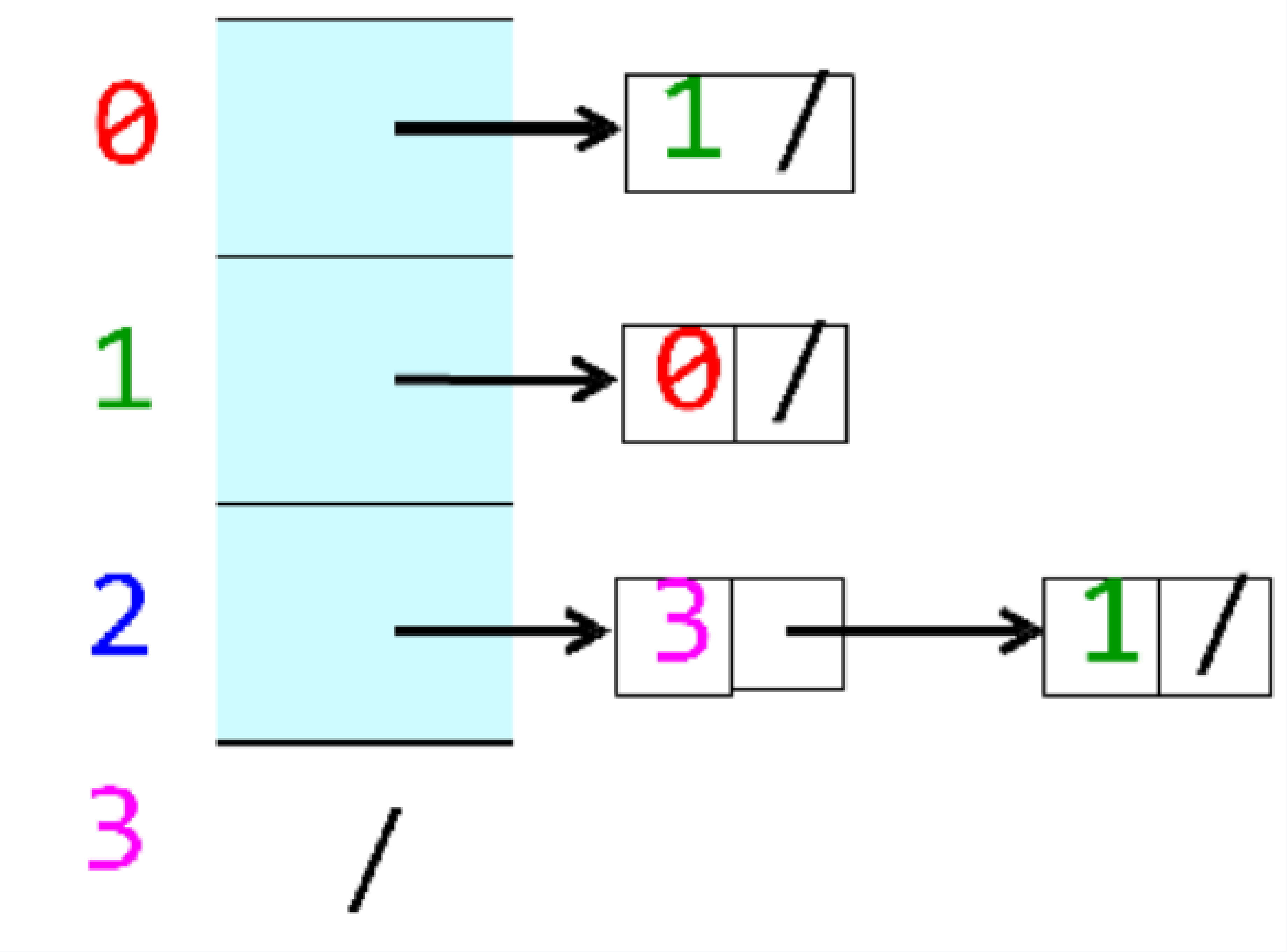
Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Concept Check!

Is this a directed or undirected graph?



For the given list representation of the graph, is the graph undirected?



Yes

0%

No

0%

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

AM versus AL

	Question	Winner	Reason
1	Faster to remove an edge?		
2	Faster to find the outdegree of a vertex?		
3	Faster to add a new vertex? (incl. edges)		
4	Less memory on sparse graphs ($\lvert E\rvert \in O(\lvert V\rvert)$)?		
5	Less memory on dense graphs ($\lvert E\rvert \in O(\lvert V\rvert^2)$)?		

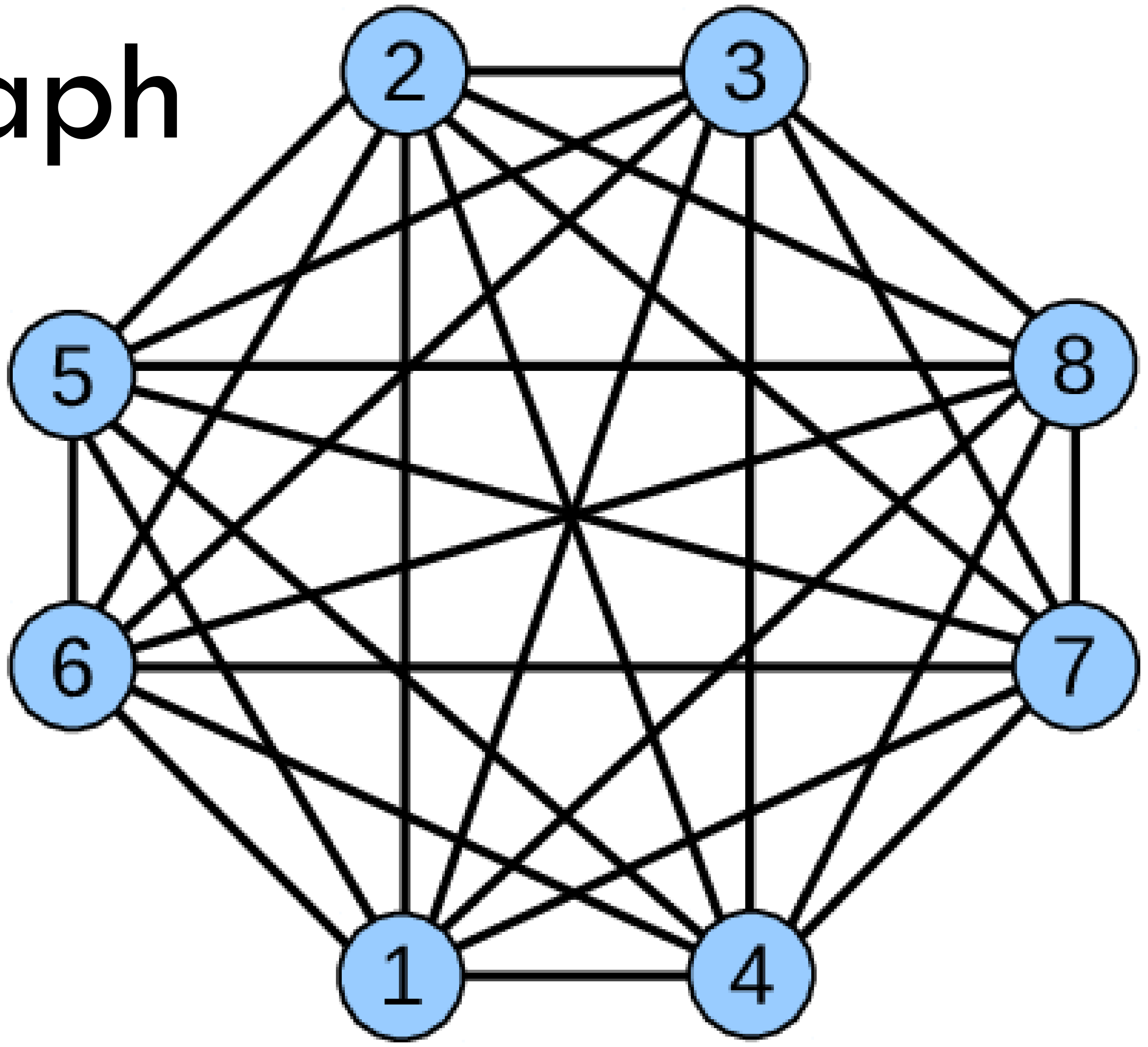
AM versus AL

	Question	Winner	Reason
1	Faster to remove an edge?	AM	O(1) to set the cell to false
2	Faster to find the outdegree of a vertex?	AL	O(deg(v)) entries to visit in AL compared to O(|V|) in AM
3	Faster to add a new vertex? (incl. edges)	AL	O(|V|) with AL but O(|V|²) with AM in the worst-case
4	Less memory on sparse graphs ($\|E\| \in O(\|V\|)$)?	AL	Many empty slots
5	Less memory on dense graphs ($\|E\| \in O(\|V\|^2)$)?	AM	Small win for AM due to extra space for pointers in case of AL

Memory Efficiency – AM or AL

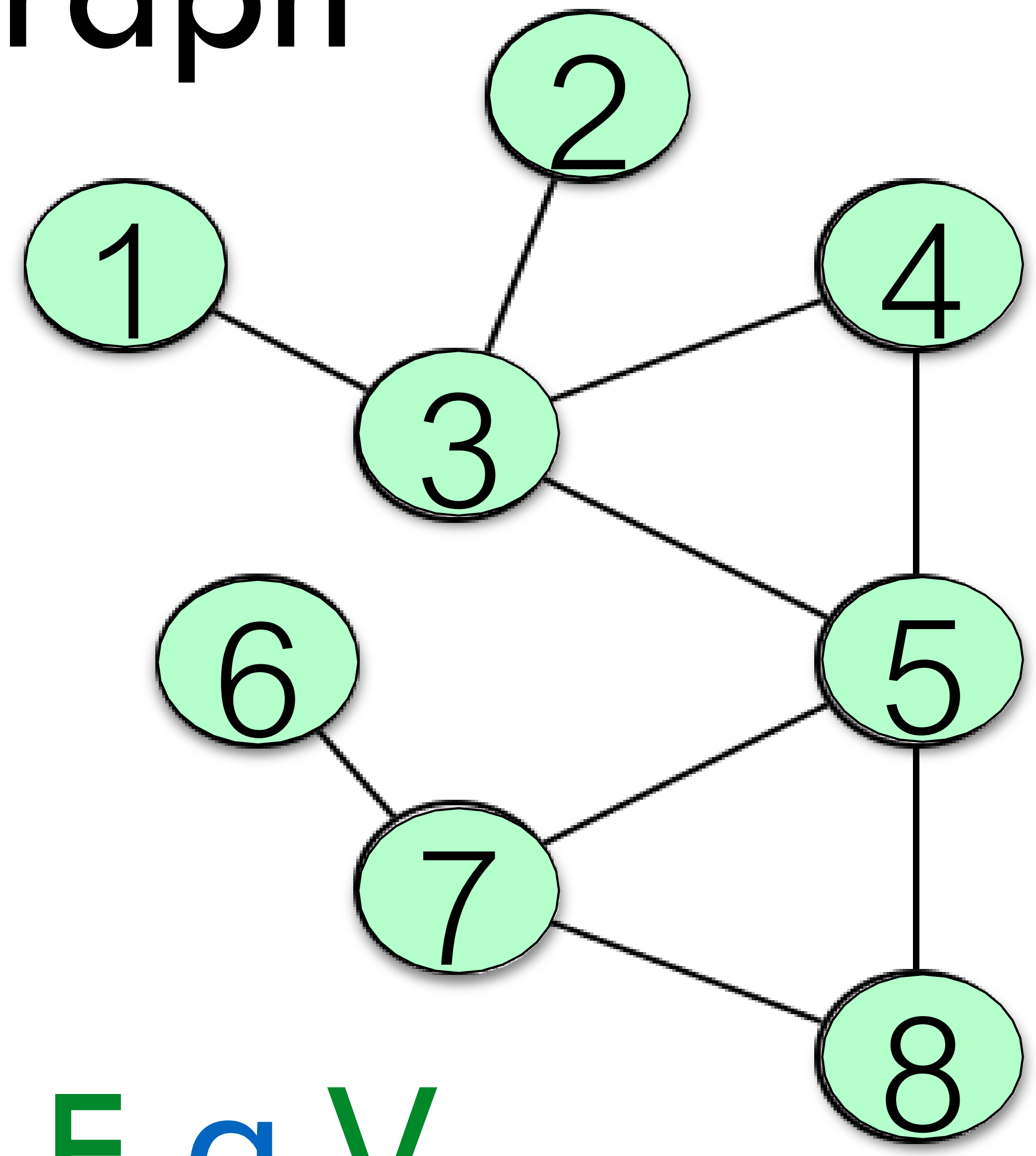
Which implementation is **more efficient** in terms of **memory usage?**

Dense Graph



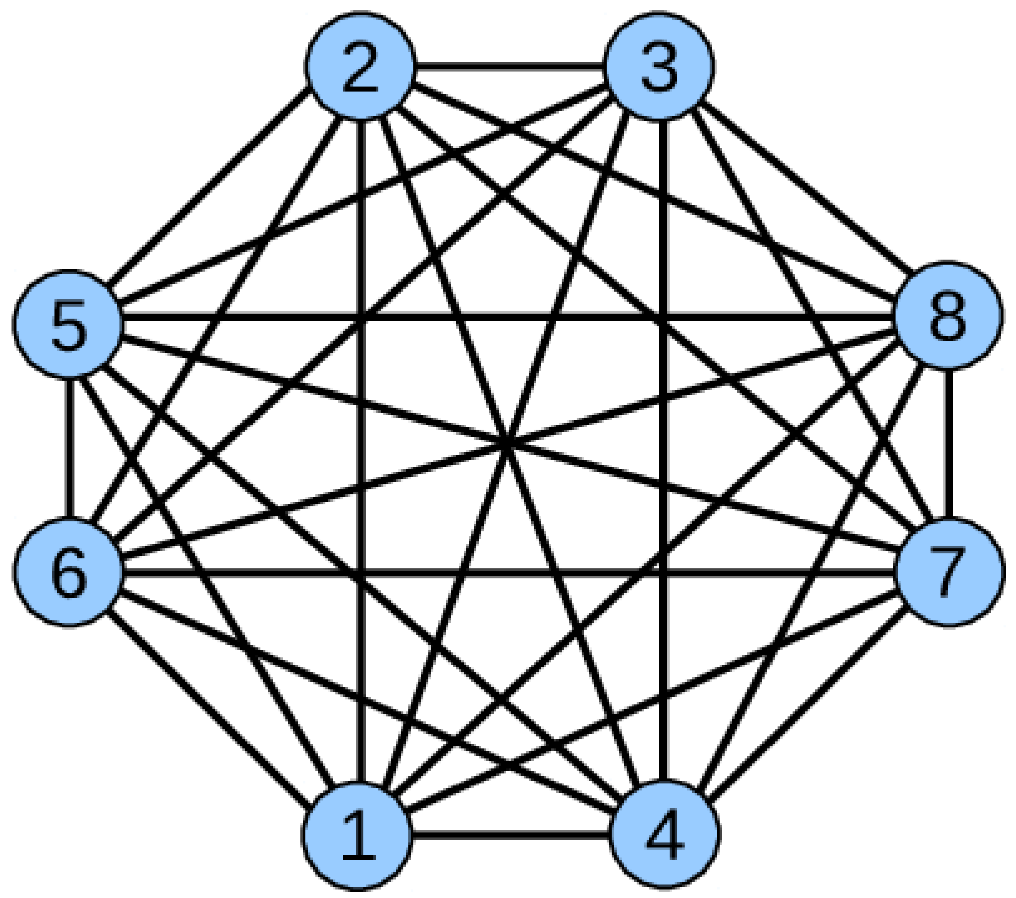
$E\ \alpha\ V^2$

Sparse Graph

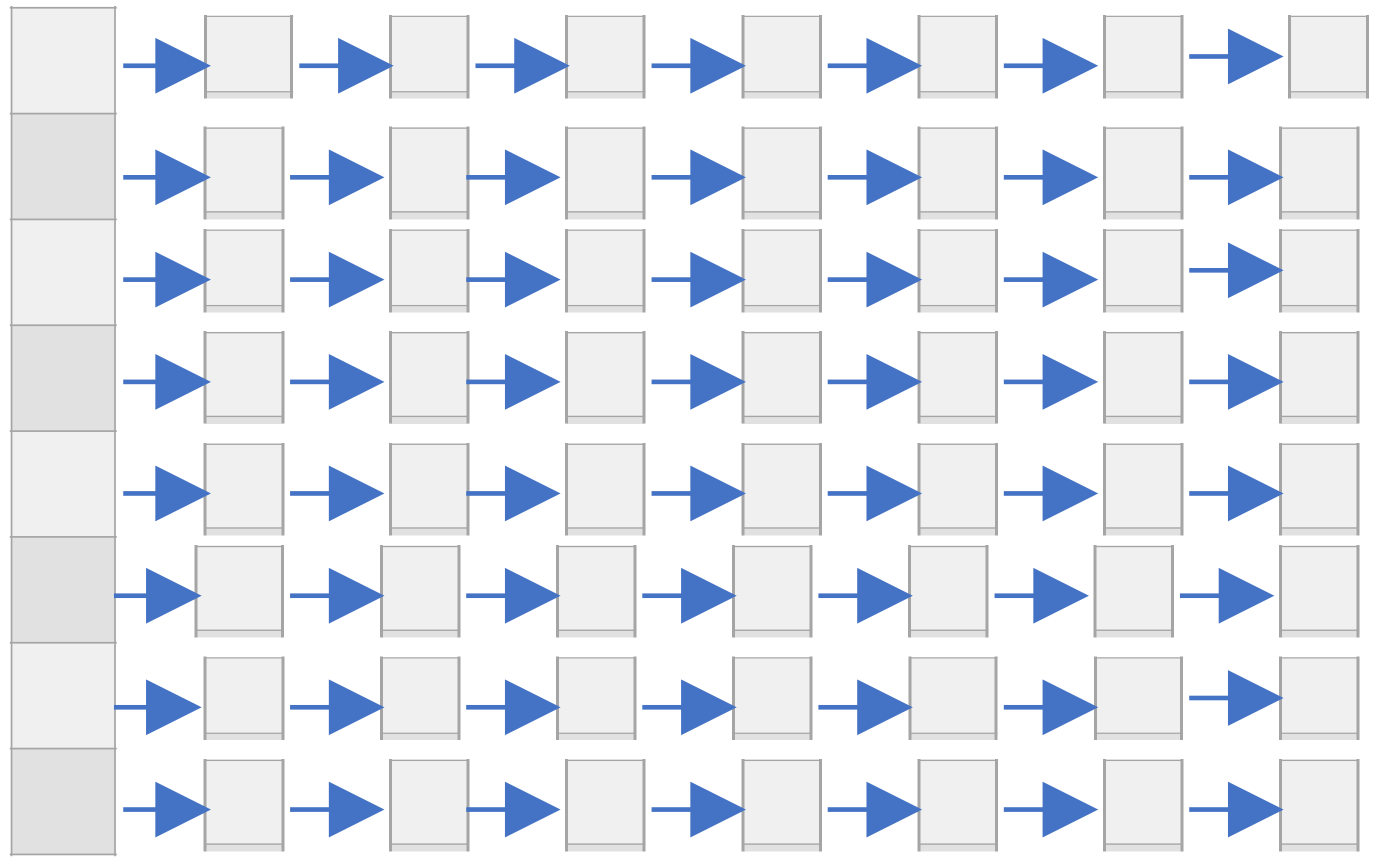


$E\ \alpha\ V$

Dense Versus Sparse Graphs



$E \alpha V^2$

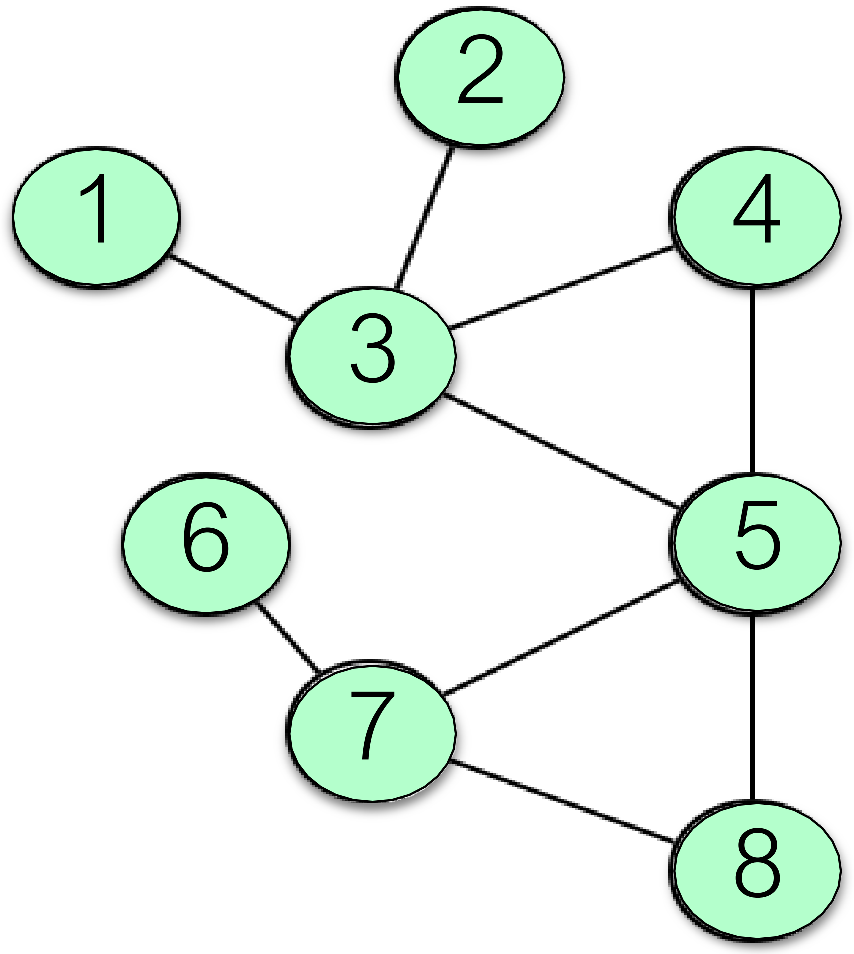


$(8 \times 8) \times (4 + 4) = 512$ bytes

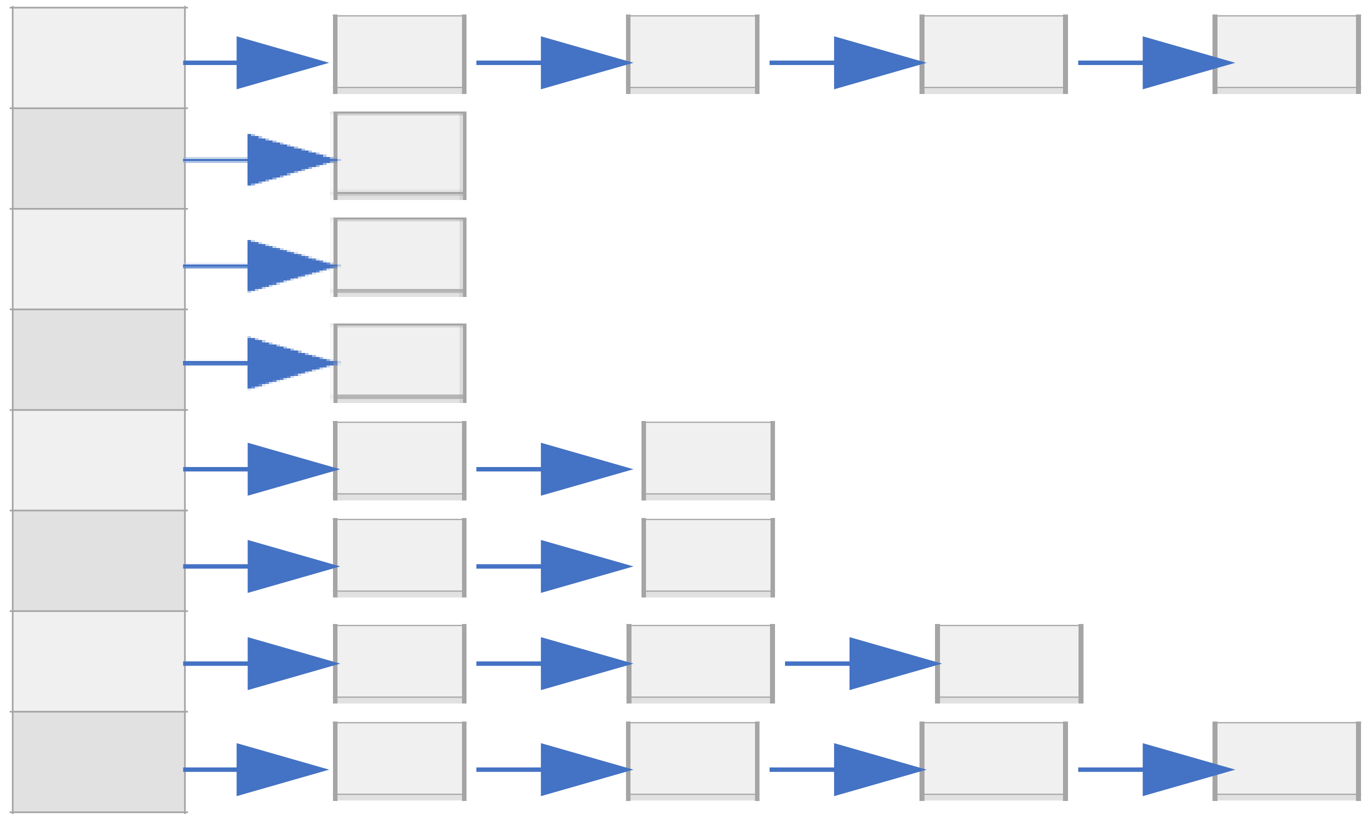
```
struct node {
int vertex;
struct node* next;
};
```

```
int cell[8][8];
```


$(8 \times 8) \times 4 = 256$ bytes



$E \alpha V$



$26 \times (4 + 4) = 208$ bytes

Memory Efficiency – AM or AL

Which implementation is **more efficient** in terms of **memory usage?**

- AL takes $O(V + E)$ + the extra space required for pointers for linked lists

- Adjacency Matrices (AMs) take more space on sparse graph, and the difference grows with the size of the graph
  - The number of empty (wasted) slots grow with the size of the graph. After a certain point, they outweigh the overhead of pointers in ALs

Questions