Introduction to Graph

Notes by Yan Yan

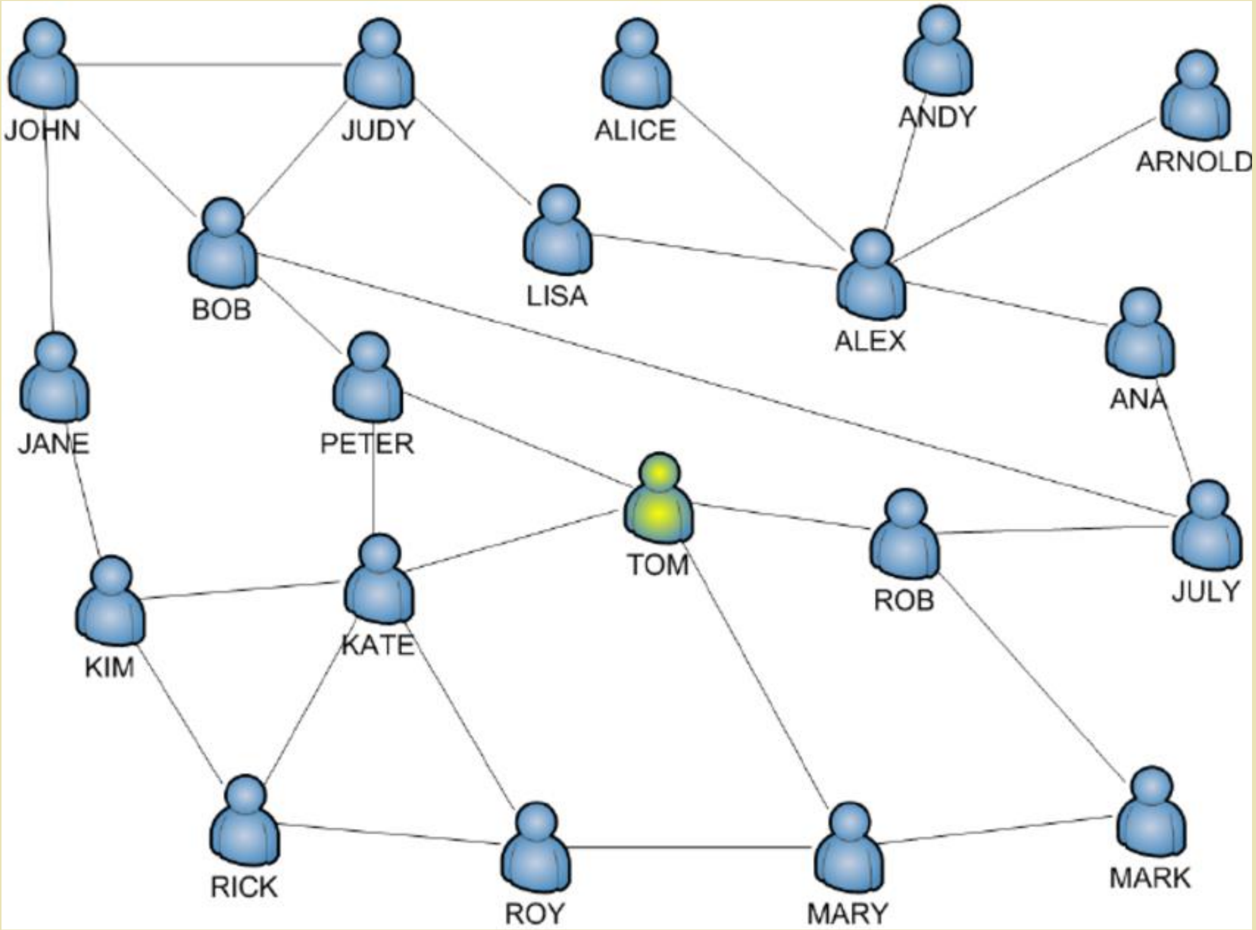
Contents

- Graph and Terminology
- Graph Representations
  - Adjacency list
  - Adjacency matrix
- Graph Traversal
  - Breadth-first search (BFS)
  - Depth-first search (DFS)
- Directed Graphs
- Weighted Graphs

Learning Objectives

1. Apply graph terminology to describe a graph
2. Given a graph, represent it using the adjacency list and adjacency matrix
3. Describe the advantages and disadvantages of adjacency list and adjacency matrix representation
4. Name different types of graphs introduced
5. Understand and implement the breadth-first search and depth-first search
6. Calculate the path lengths in a graph

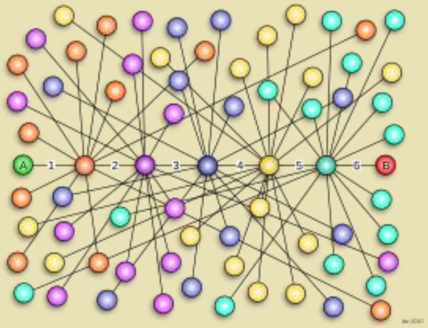
What is a Graph



https://www.researchgate.net/figure/An-example-social-network_fig2_220886840

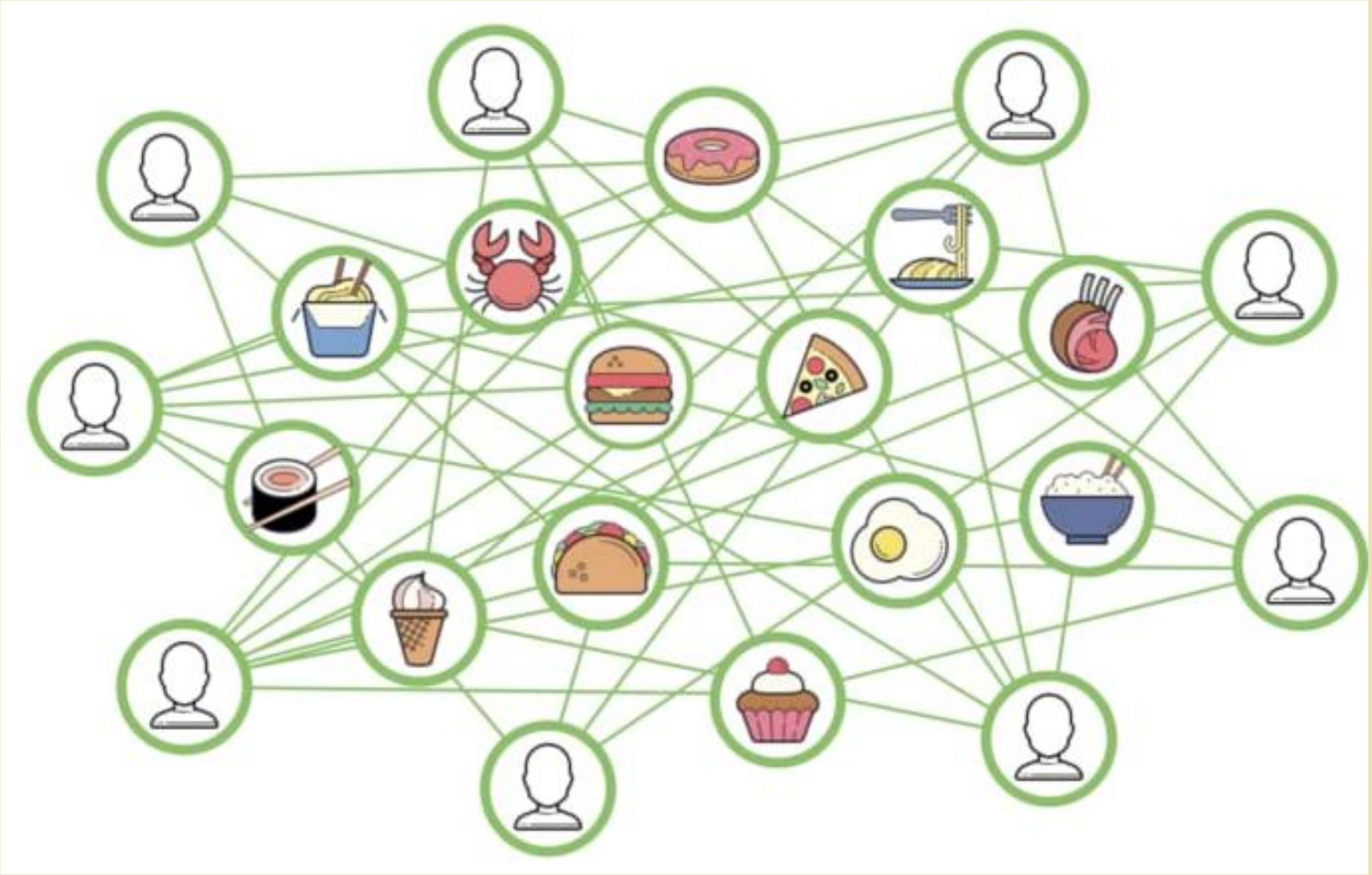
What is a Graph

- The small-world experiment comprised several experiments conducted by Stanley Milgram and other researchers examining the average path length for social networks of people in the United States.
- The research was groundbreaking in that it suggested that human society is a small-world-type network characterized by short path-lengths



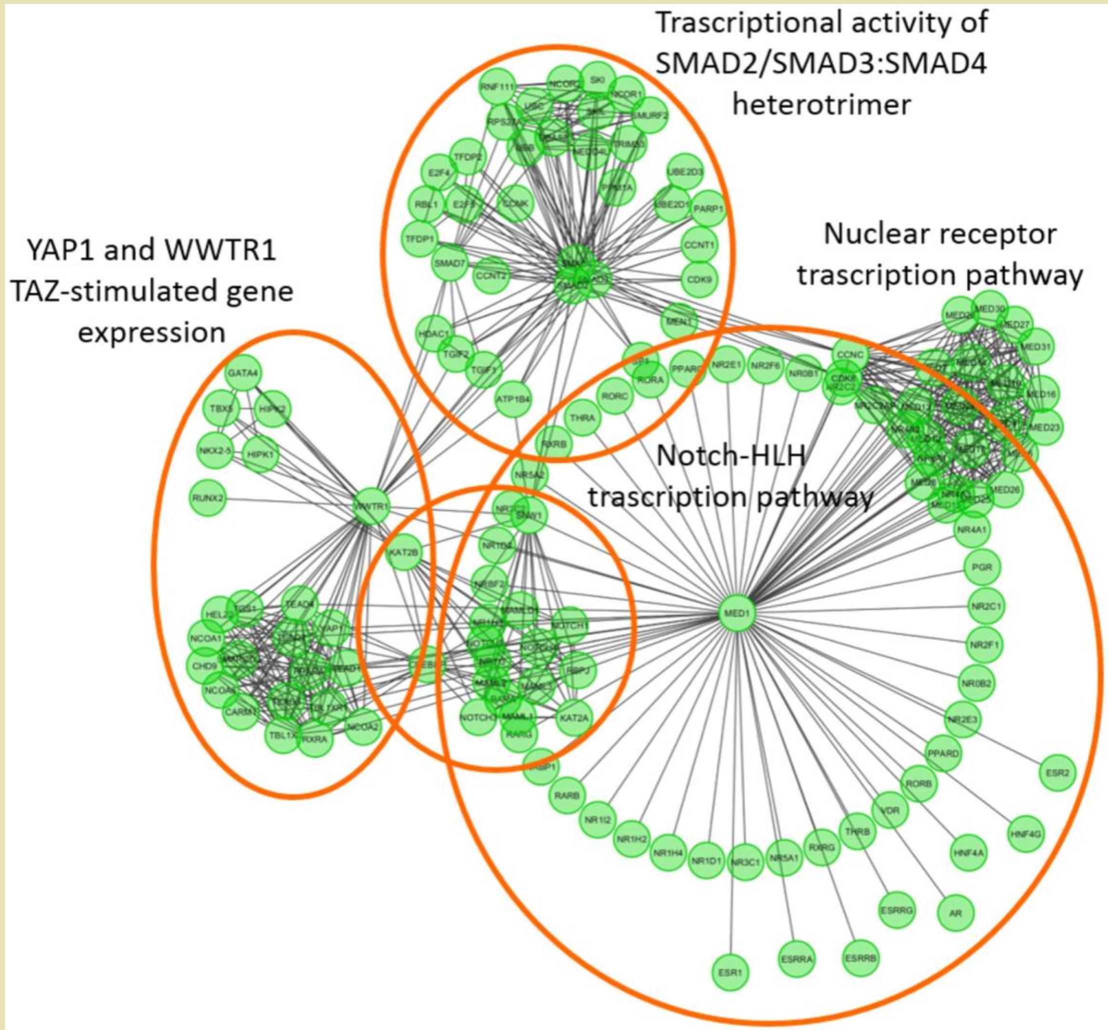
https://en.wikipedia.org/wiki/Small-world_experiment

What is a Graph



https://www.uber.com/en-CA/blog/uber-eats-graph-learning/

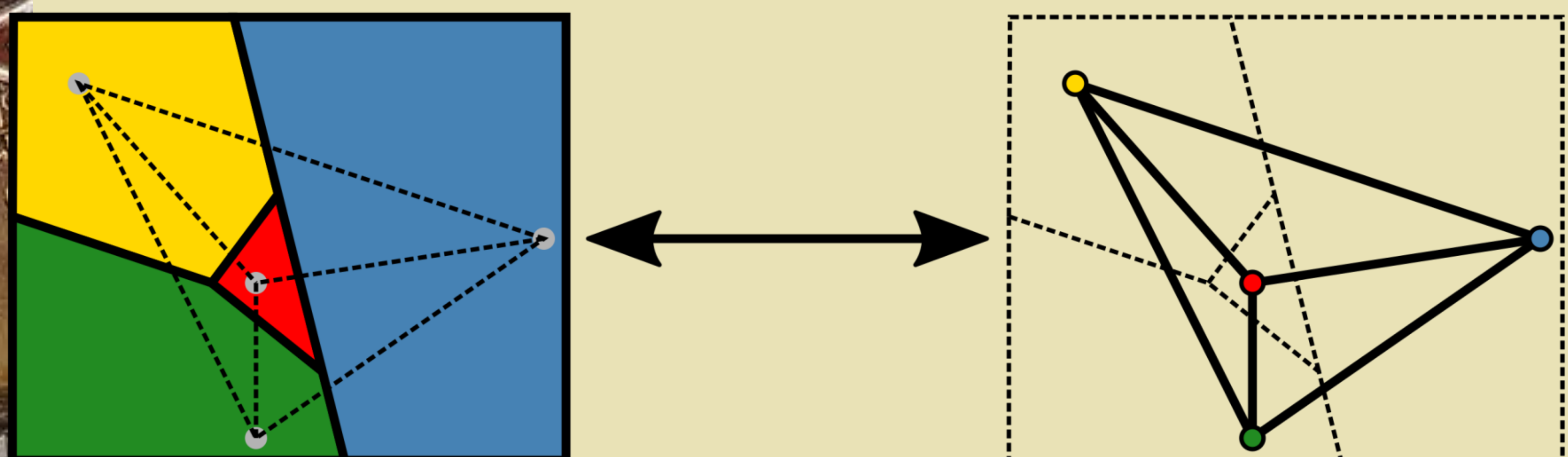
What is a Graph



https://www.nature.com/articles/s41598-018-23672-0/figures/1

What is a Graph

- Four color map theorem, states that no more than four colors are required to color the regions of any map so that no two adjacent regions have the same color.

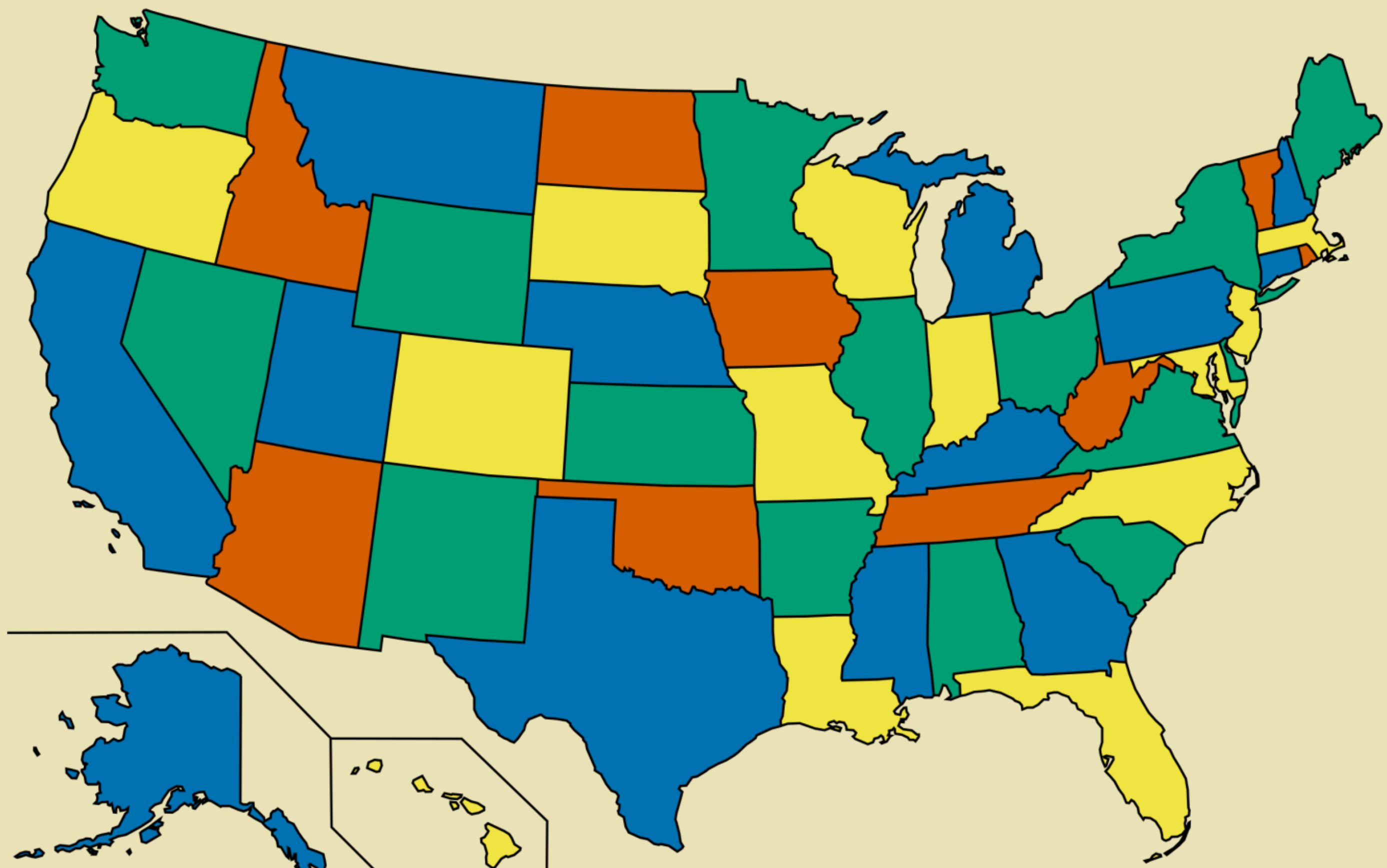


https://en.wikipedia.org/wiki/Four_color_theorem

a **coloring** of a graph almost always refers to a *proper vertex coloring*.

What is a Graph

- Example: A four-colored map of the states of the United States (ignoring lakes and oceans)



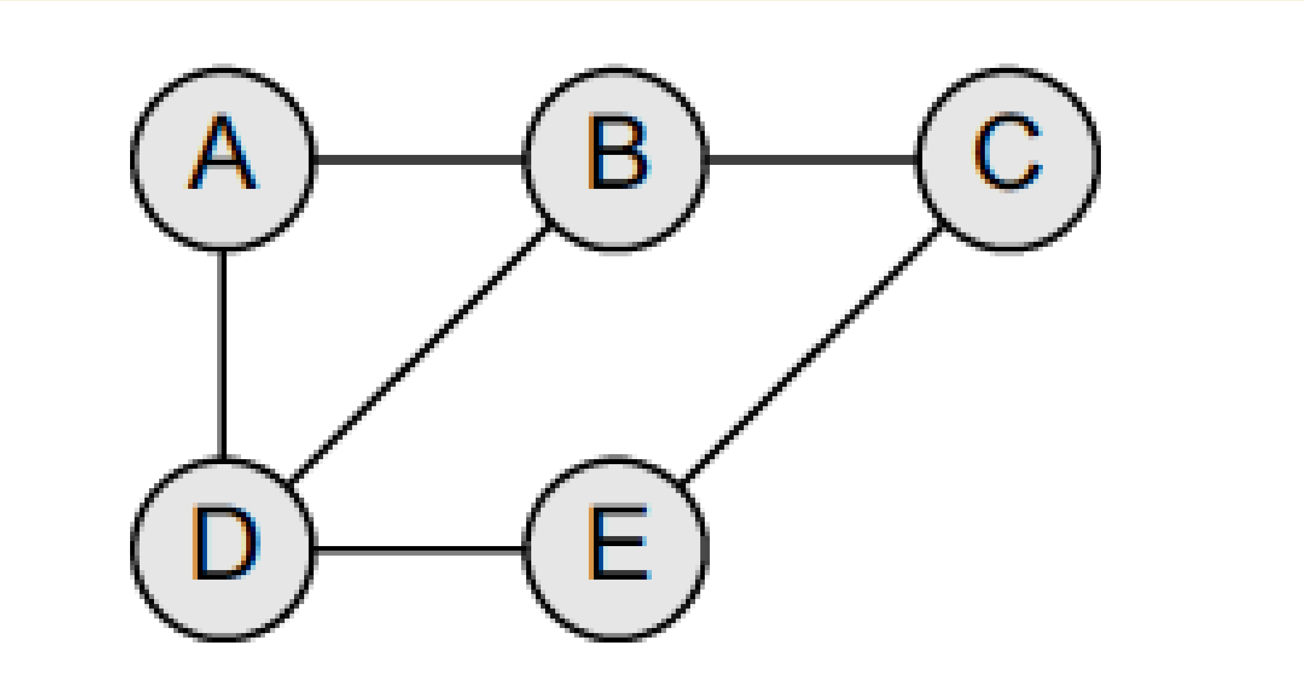
https://en.wikipedia.org/wiki/Four_color_theorem

Definition

- A graph is a non-linear data structure, a collection of nodes (vertices) connected to each other via edges.
- A graph can be defined as G = (V, E)
  - V is a set of vertices $\{v_1, v_2, \ldots v_n\}$
  - E is a set of edges, $e_1 = (v_1, v_2), e_2 = (v_1, v_3), \ldots$
  - Edges can be directed or undirected (more on this later)
- A graph is often viewed as a generalization of the tree structure

Definition -- Example

- V(G) = {A, B, C, D and E} and E(G) = {(A, B), (B, C), (A, D), (B, D), (D, E), (C, E)}.

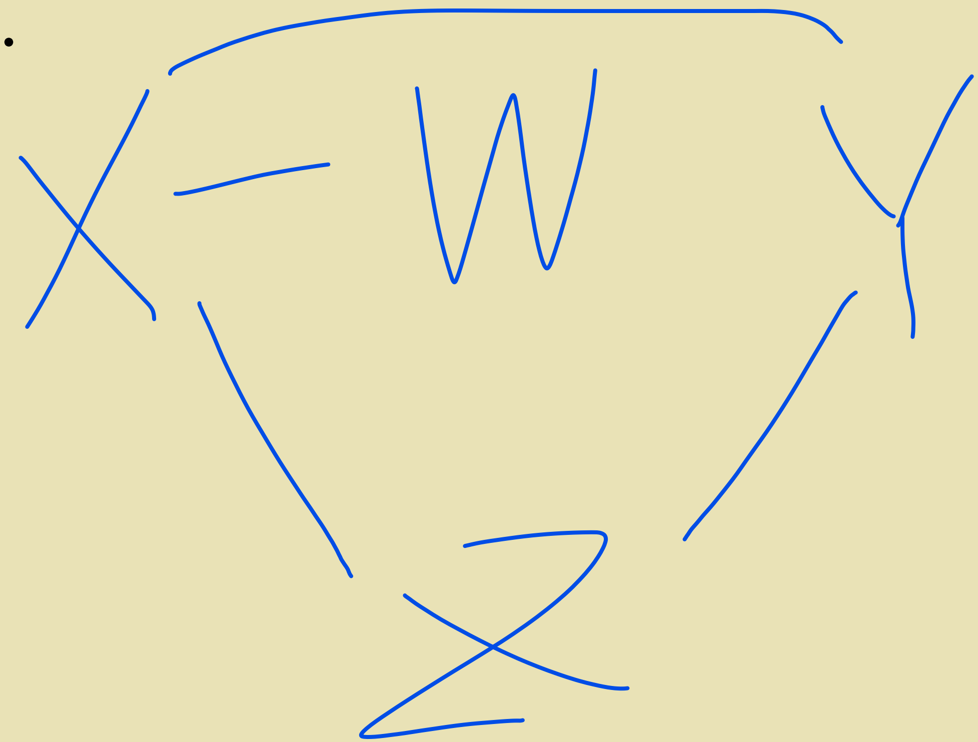


- Is it a directed graph or undirected graph?

Figure from book Data Structures Using C, 2nd edition (Chapter 13, Figure 13.1), by Reema Thareja

Definition -- Exercise

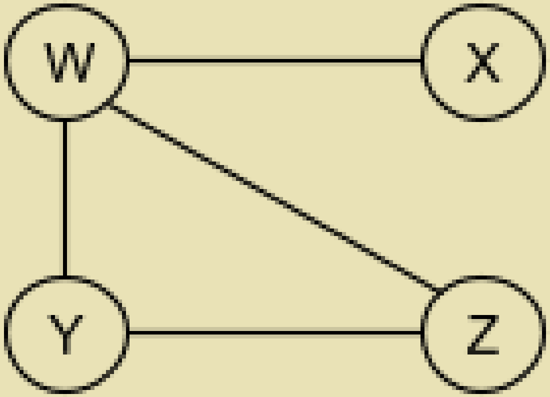
- Given V(G) = {W, X, Y, Z} and E(G) = {(W, X), (W, Y), (W, Z), (Y, Z)}. Draw the undirected graph.



Zybook 11.3 Graph representations: Adjacency lists

Terminology

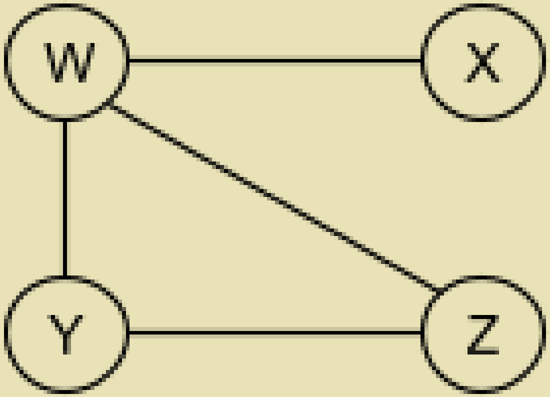
- Two nodes are **adjacent** if there is an edge connecting them.
- The **degree** of a node is the number of edges connected to it.
- A **path** is a sequence of edges leading from a source (starting) vertex to a destination (ending) vertex. The path length is the number of edges in the path.



Zybook 11.3 Graph representations: Adjacency lists

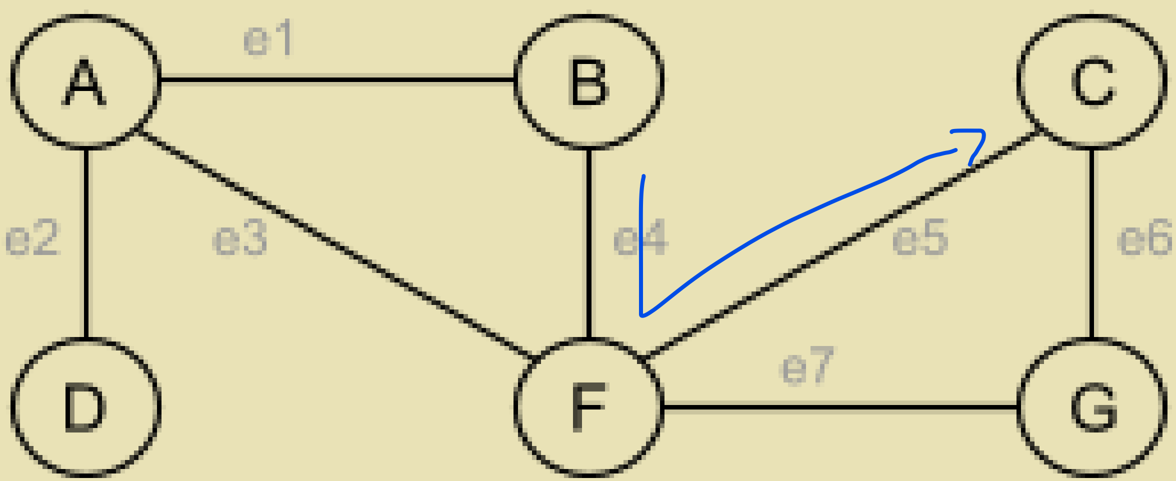
Terminology

- A **cycle** is a path that starts and ends at the same vertex, forming a closed loop.
- The **distance** between two vertices is the number of edges on the shortest path between those vertices.



Definition -- Exercise

1. Are A and B adjacent? Y
2. Is there a cycle in this graph? N
3. Name a path (vertices visited) from B to C. F
4. What is the distance between B and C? 2



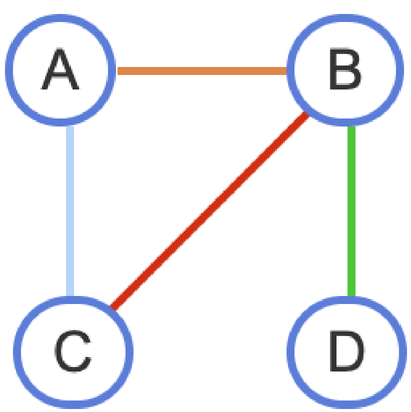
Zybook 11.3 Graph representations: Adjacency lists

Terminology

- Types of Graph
  - **Directed** Graph (Digraph): Edges have a direction, indicating a one-way relationship.
  - **Undirected** Graph: Edges have no direction, representing mutual relationships.
  - **Weighted** Graph: Each edge has a weight or cost associated with it, often used in optimization problems.
  - **Cyclic** Graph: Contains at least one cycle (a path that starts and ends at the same node).
  - **Acyclic** Graph: Does not contain any cycles.
  - **Connected** Graph: There is a path between every pair of nodes.

Graph representations: Adjacency lists

- In an **adjacency list** graph representation, each vertex has a list of adjacent vertices, each list item representing an edge.

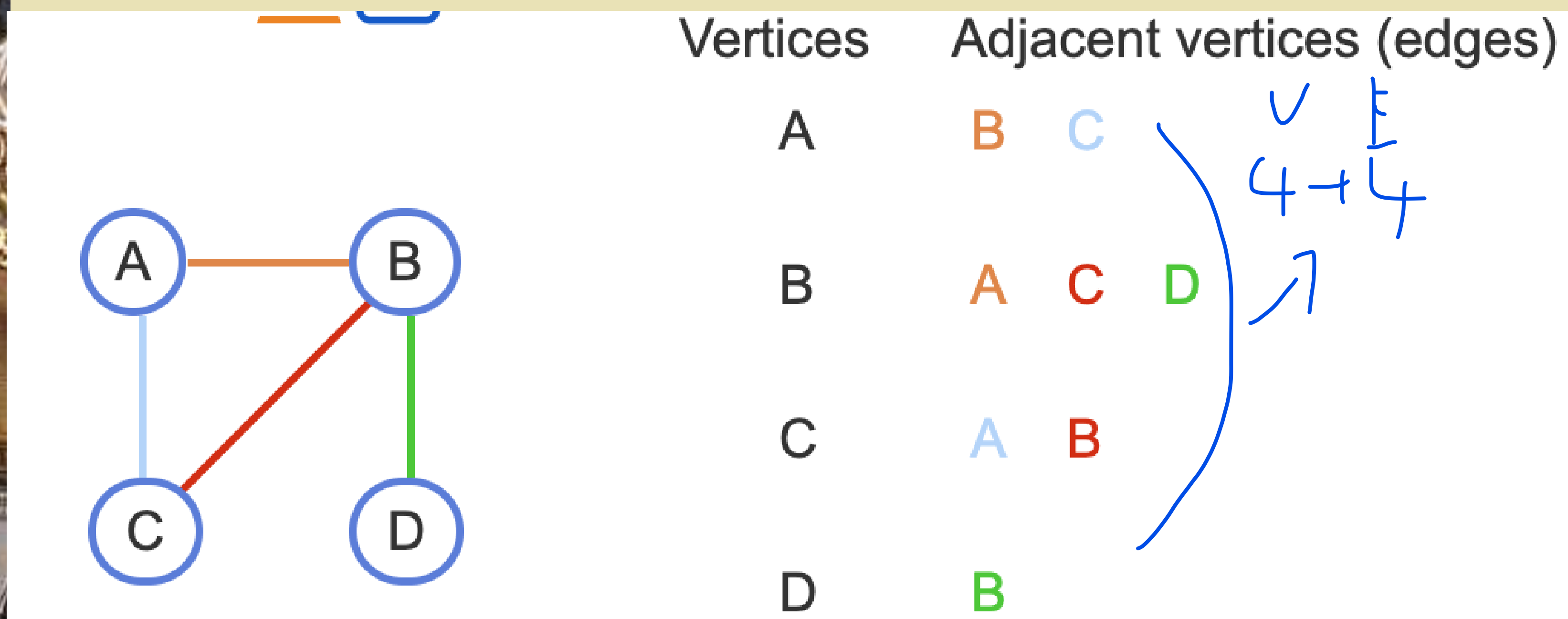


Vertices	Adjacent vertices (edges)
A	B C
B	A C D
C	A B
D	B

Zybook 11.3 Graph representations: Adjacency lists

Graph representations: Adjacency lists

- A key advantage of an adjacency list graph representation is a size of O(V + E)
  - each vertex appears once
  - each edge appears twice



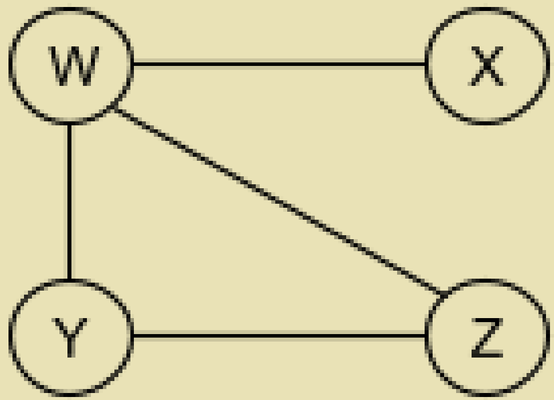
Zybook 11.3 Graph representations: Adjacency lists

Graph representations: Adjacency lists

- A disadvantage is that determining whether two vertices are adjacent is O(V)
  - must be traversed all other vertices to check if they are connected to this vertex, and that list could have V items.
  - However, in most applications, a vertex is only adjacent to a small fraction of the other vertices, yielding a sparse graph.
- A **sparse** graph has far fewer edges than the maximum possible.

Adjacency lists Exercise

- Write down the adjacency lists of the following graph

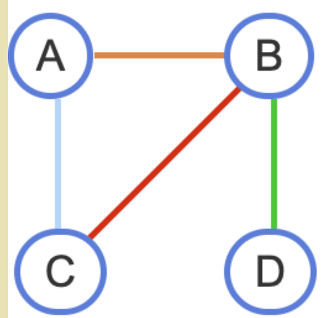


Vertices	Adjacent vertices		
W	X	Y	Z
X	W		
Y	W	Z	
Z	W	Y	

Zybook 11.3 Graph representations: Adjacency lists

Graph representations: Adjacency matrix

- In an **adjacency matrix** graph representation, each vertex is assigned to a matrix row and column, and a matrix element is 1 if the corresponding two vertices have an edge or is 0 otherwise.



	A	B	C	D
A		1	1	
B	1		1	1
C	1	1		
D		1		

Zybook 11.4 Graph representations: Adjacency matrices

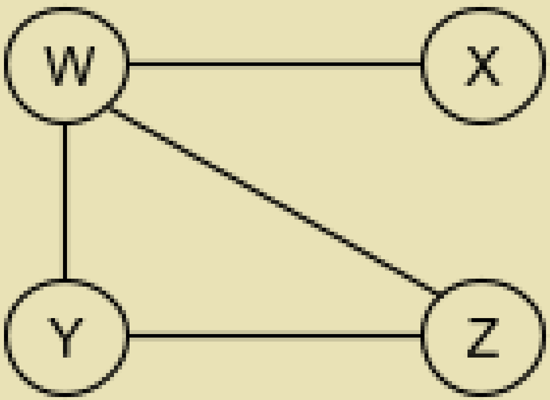
Graph representations: Adjacency matrix

- Assuming the common implementation as a two-dimensional array whose elements are accessible in O(1), then an adjacency matrix's key benefit is O(1) determination of whether two vertices are adjacent: The corresponding element is just checked for 0 or 1.
- A key drawback is $O(V^2)$ size.
  - Ex: A graph with 1000 vertices would require a 1000 x 1000 matrix, meaning 1,000,000 elements. An adjacency matrix's large size is inefficient for a sparse graph, in which most elements would be 0's.

Zybook 11.4 Graph representations: Adjacency matrices

Adjacency Matrix Exercise

- Write down the adjacency matrix of the following graph



	W	X	Y	Z
W	0	1	1	1
X	1	0	0	0
Y	1	0	0	1
Z	1	0	1	0

Zybook 11.4 Graph representations: Adjacency matrices

Adjacency Matrix Exercise

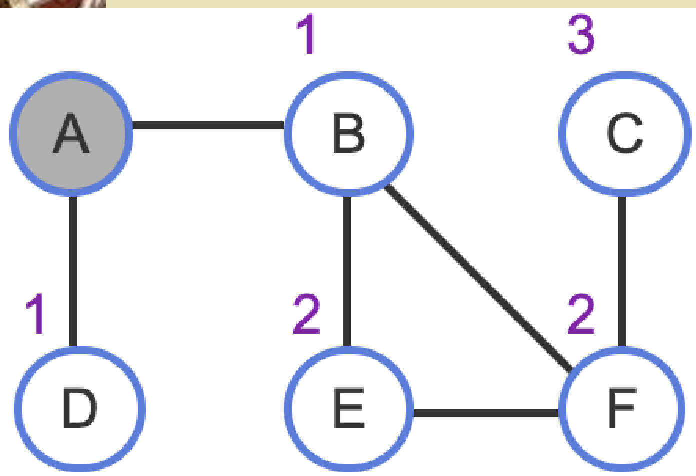
- Given the adjacency matrix,
  1. How many edges does D have? 3
  2. Is there a path from A to E? N

	A	B	C	D	E
A	0	1	1	1	0
B	1	0	1	1	1
C	1	1	0	1	1
D	1	1	1	0	0
E	0	1	1	0	0

Zybook 11.4 Graph representations: Adjacency matrices

Graphs: Breadth-first search

- A **breadth-first search** (BFS) is a traversal that visits a starting vertex, then all vertices of distance 1 from that vertex, then of distance 2, and so on, without revisiting a vertex.



Starting at A: A B D E F C

0 1 1 2 2 3

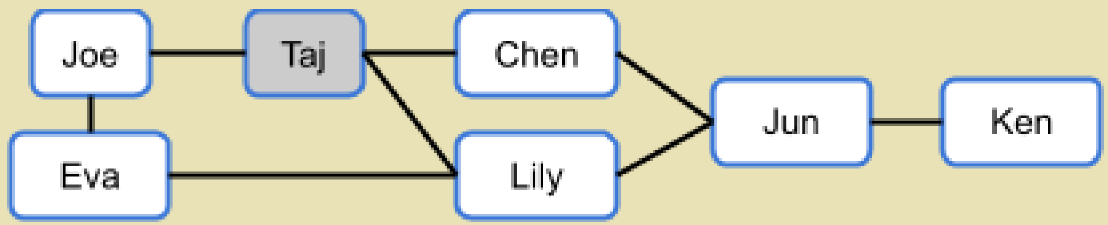
Zybook 11.5 Graphs: Breadth-first search

Graphs: Breadth-first search

- Example: Social networking connection recommender
  - Social networking sites like Facebook and LinkedIn use graphs to represent connections among people. For a particular user, a site may wish to recommend new connections. One approach does a breadth-first search starting from the user, recommending new connections starting at distance 2 (distance 1 people are already connected with the user).

Graphs: Breadth-first search

- Example: Social networking connection recommender



Breadth-first traversal:	Taj	Joe	Chen	Lily	Eva	Jun	Ken
	0	1	1	1	2	2	3

Hello Taj. People you may know:

Eva

Jun

Ken

Zybook 11.5 Graphs: Breadth-first search

Graphs: Breadth-first search algorithm

```
BFS(startVertex) {
  initialize queue with startVertex
  initialize discoveredSet with startVertex

  while (queue is not empty) {
    currentVertex = dequeue from queue
    // Visit or process the current vertex
    process(currentVertex) // can be print or others
    // Explore each adjacent vertex of the current vertex
    for each neighbor of currentVertex {
      if (neighbor is not in discoveredSet) {
        // Enqueue undiscovered neighbor and mark it as
discovered
        enqueue neighbor in queue
        add neighbor to discoveredSet
      }
    }
  }
}
```

Graphs: Breadth-first search

- When the BFS algorithm first encounters a vertex, that vertex is said to have been **discovered**.
- In the BFS algorithm, the vertices in the queue are called the **frontier**, being vertices thus far discovered but not yet visited.
- Because each vertex is visited at most once, an already-discovered vertex is not enqueued again.
- A "visit" may mean to print the vertex, append the vertex to a list, compare vertex data to a value and return the vertex if found, etc.

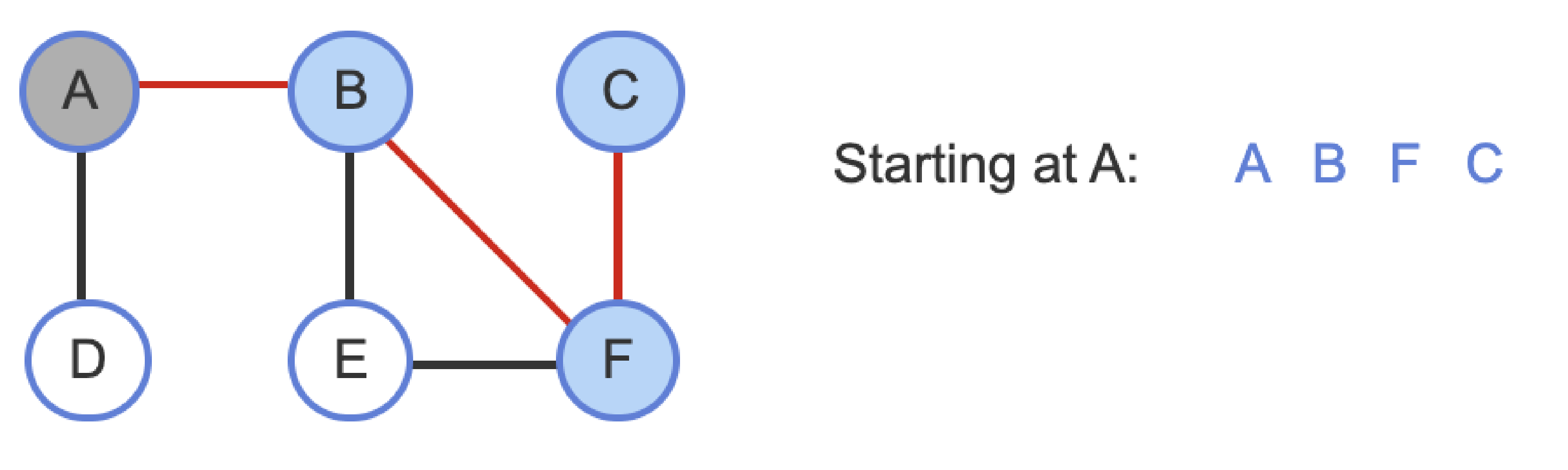
Zybook 11.5 Graphs: Breadth-first search

Graphs: Depth-first search

- A **depth-first search** (DFS) is a traversal that visits a starting vertex, then visits every vertex along each path starting from that vertex to the path's end before backtracking.

Graphs: Depth-first search

- Example

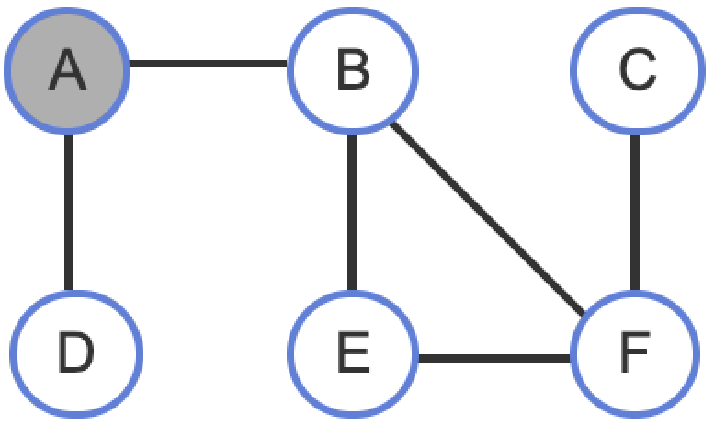


Starting at A descends along a path to the path's end before backtracking.

Zybook 11.6 Graphs: depth-first search

Graphs: Depth-first search

- Example



Starting at A: A B F C E

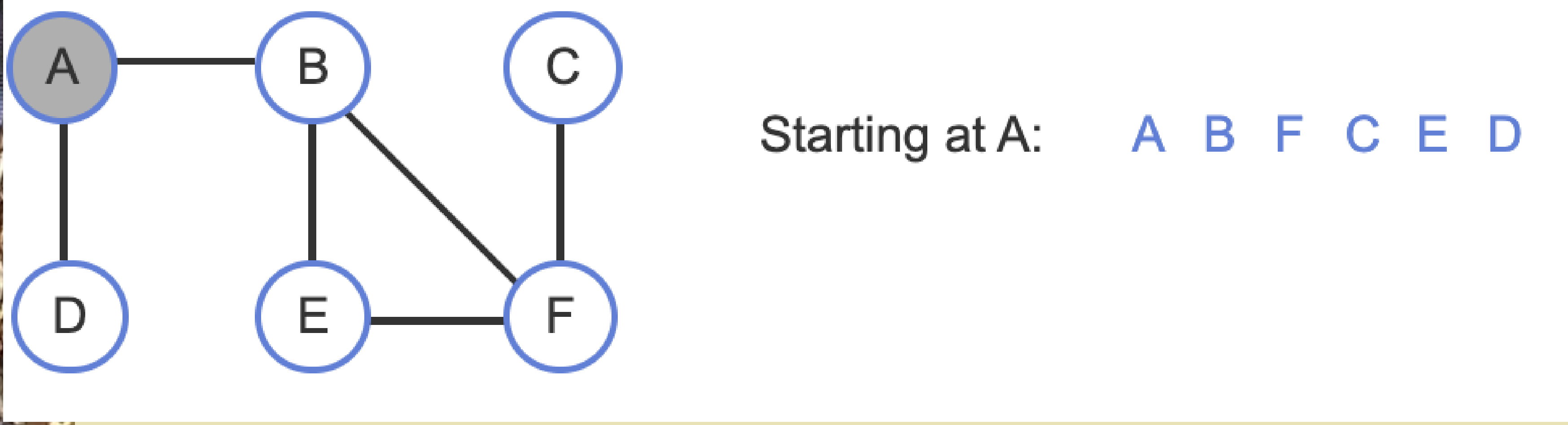
A B F C F E

Reach path's end: Backtrack to F, visit F's other adjacent vertex (E). B already visited, backtrack again.

Zybook 11.6 Graphs: depth-first search

Graphs: Depth-first search

- Example

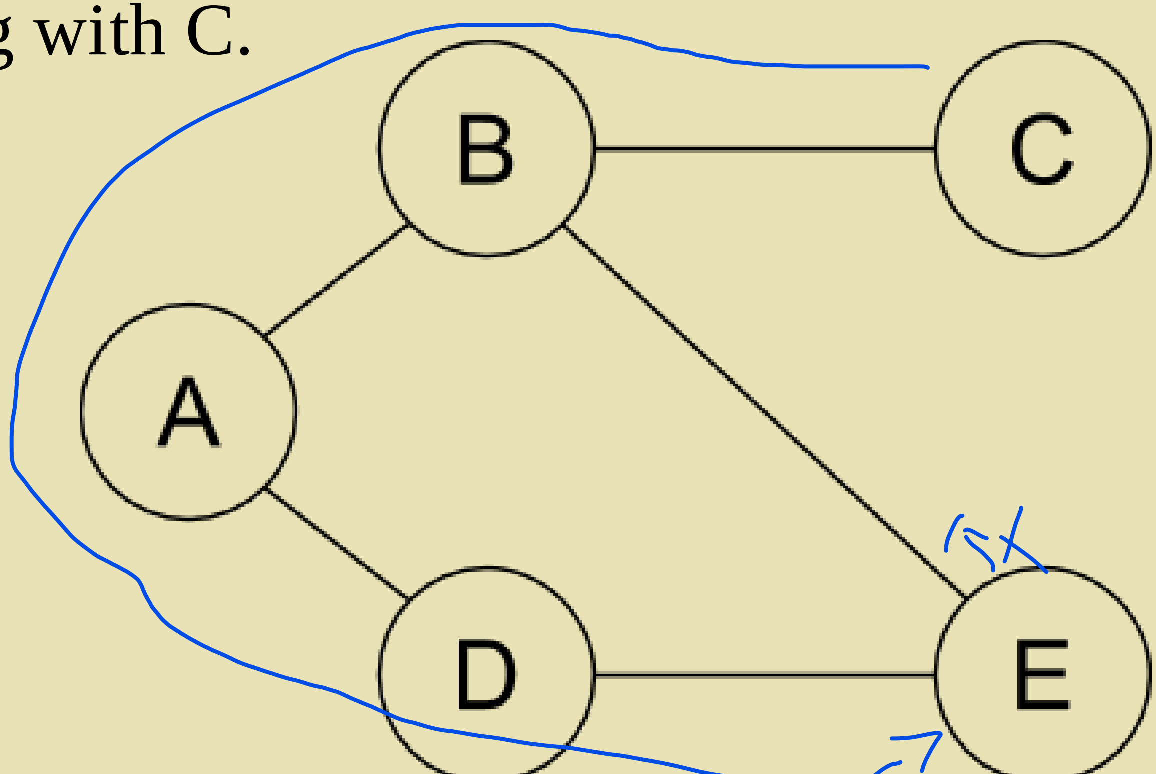


Backtracked all the way to A. Visit A's other adjacent vertex (D). No other adjacent vertices: Done.

Zybook 11.6 Graphs: depth-first search

Graphs: Depth-first search

- Exercise: Perform a valid depth-first search of the graph below. Assume the starting vertex is C. Write down the order of the vertices visited, starting with C.



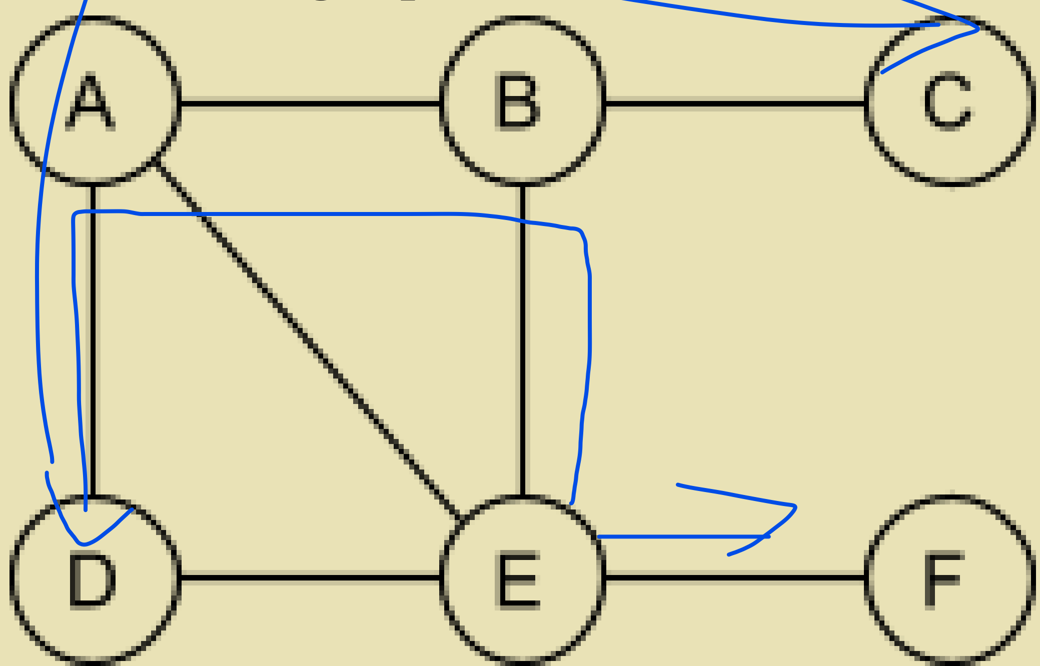
Zybook 11.6 Graphs: depth-first search

Graphs: Depth-first search

- Exercise: Perform a depth-first search of the graph below. Assume the starting vertex is E. Write down the order of vertices visited (start with E).
  - When a vertex has multiple connected vertices, the algorithm visit them using alphabet order.

E B A D C F

→ Stack



Zybook 11.6 Graphs: depth-first search

Graphs: Depth-first search algorithm

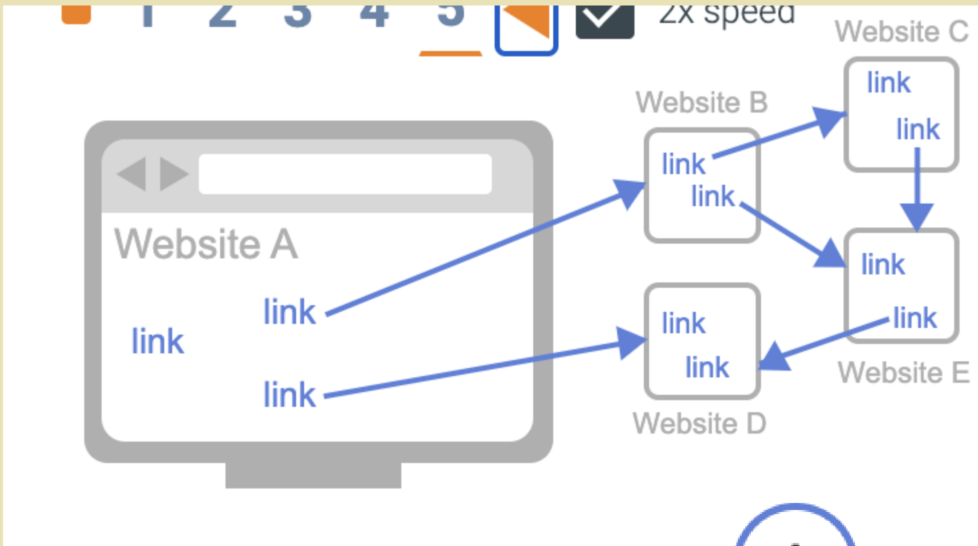
```
DFS(startVertex) {
   initialize stack with startVertex

   while (stack is not empty) {
      currentVertex = pop from stack
      if (currentVertex is not in visitedSet) {
         // Visit or process the current vertex
         process(currentVertex)
         // Mark the current vertex as visited
         add currentVertex to visitedSet

         // Push each unvisited adjacent vertex to the stack
         for each neighbor of currentVertex {
            if (neighbor is not in visitedSet) {
               push neighbor to stack
            }
         }
      }
   }
}
```

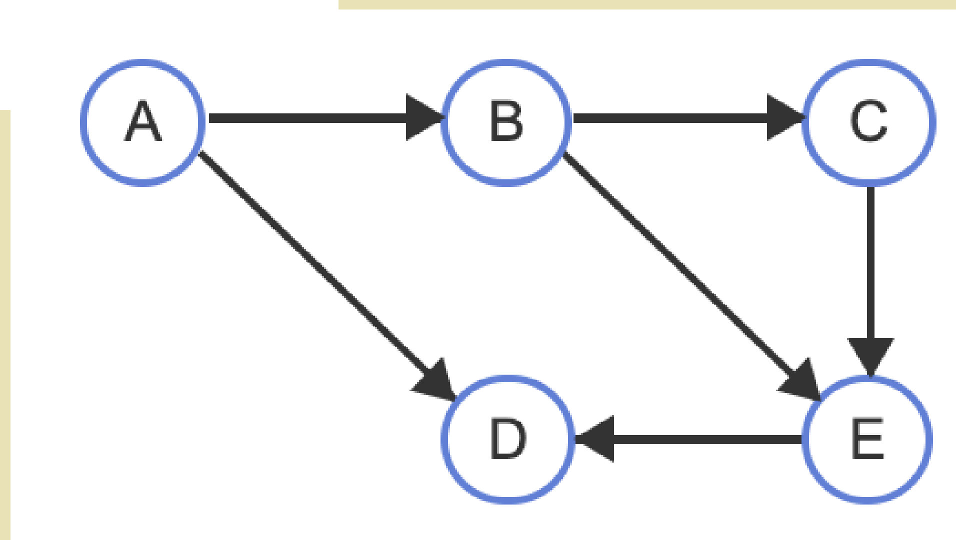
Graphs: Depth-first search recuersive algorithm

```
RecursiveDFS(vertex) {
  // If the vertex has not been visited yet
  if (vertex is not in visitedSet) {
    // Mark the vertex as visited
    add vertex to visitedSet

    // Visit or process the current vertex
    process(vertex)

    // Recursively visit each adjacent vertex
    for each neighbor of vertex {
      RecursiveDFS(neighbor)
    }
  }
}
```

Directed Graphs

- Example

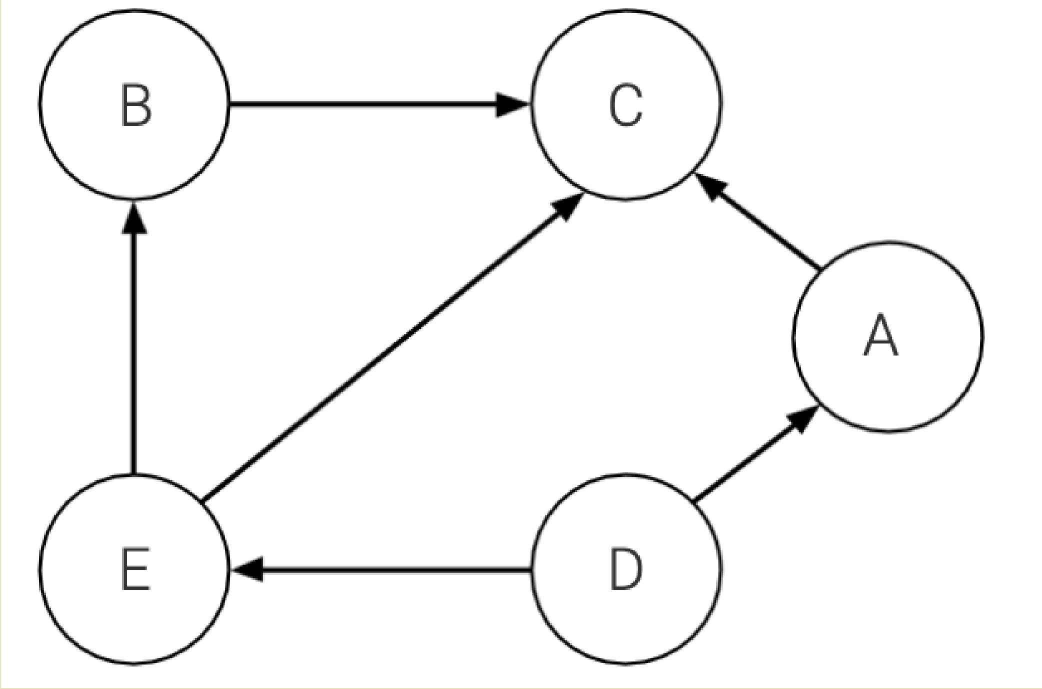




Zybook 11.7 Directed Graphs

Directed Graphs

- Exercise
  1. Which vertices are adjacent to D? E A
  2. B is adjacent to C.
  3. Are there circles in this graph? NO

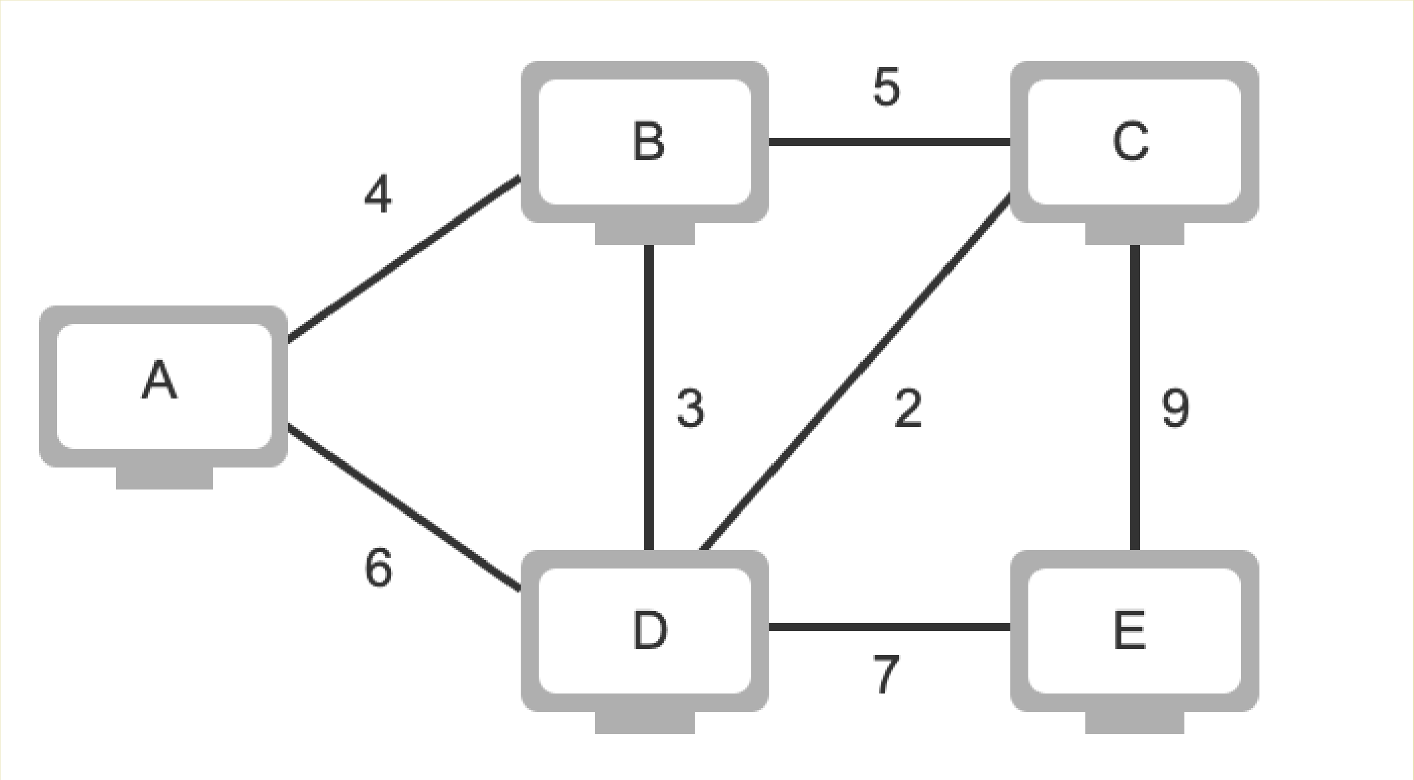


Weighted Graphs

- A **weighted graph** associates a weight (w) with each edge.
  - E.g. e=(A,B), w_e=5
- A graph edge's weight, or cost, represents some numerical value between vertex items, such as flight cost between airports, connection speed between computers, or travel time between cities.
- A weighted graph may be directed or undirected.
- In a weighted graph, the **path length** is the sum of the edge weights in the path.
  - Find shortest path problem
- Unweighted graph can be viewed as w=1 for all edges.

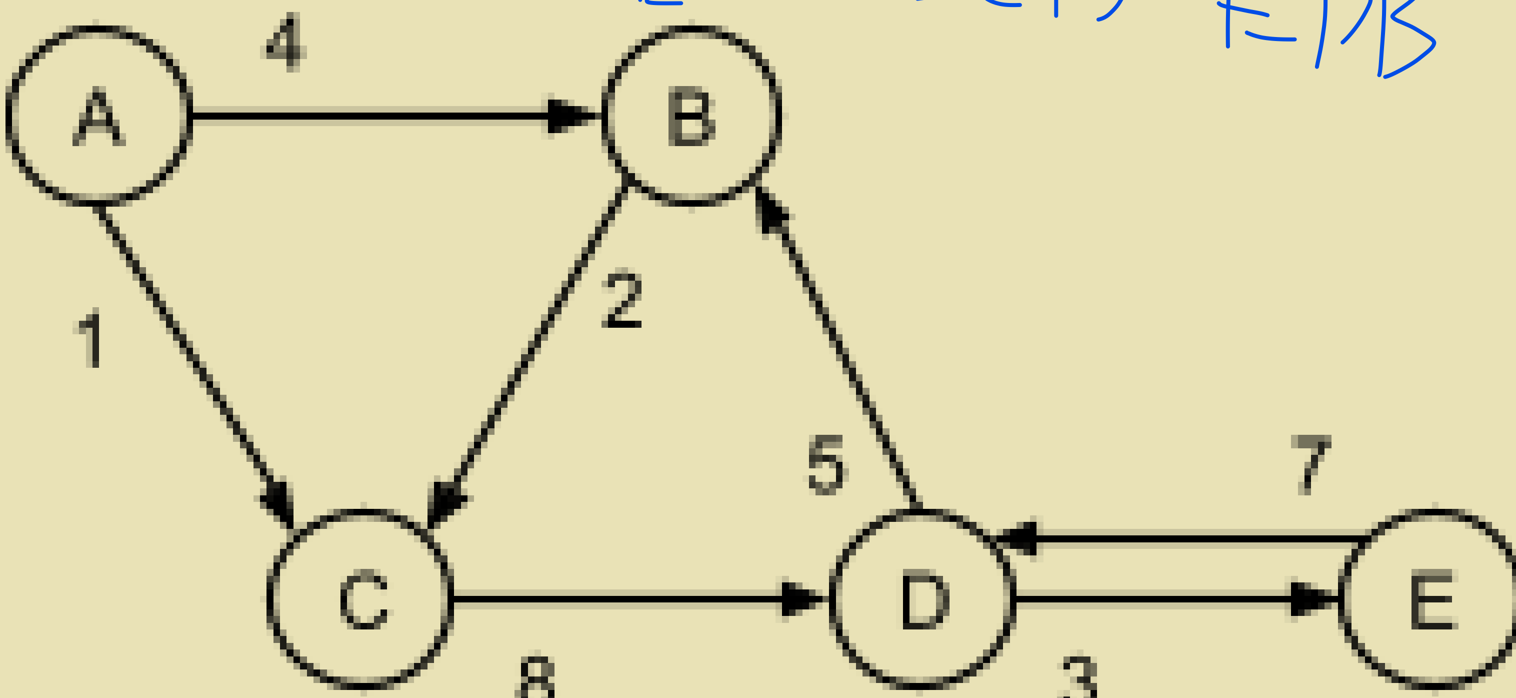
Weighted Graphs

- Examples
  - Path A-B-C, the path length is 9



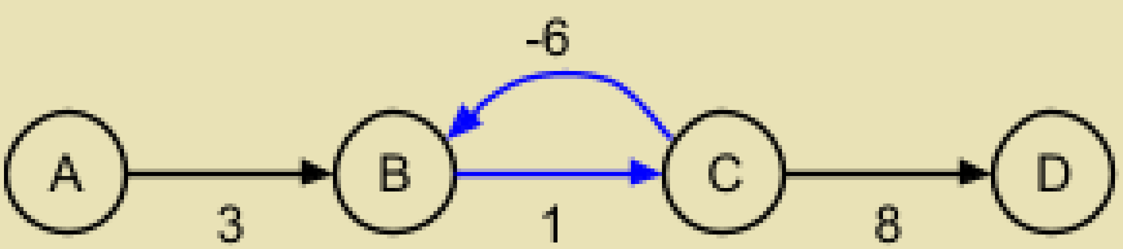
Weighted Graphs

- Directed weighted graph example
  - Cycle in this graph B→C →D →B, cycle length is 15.
  - Any other cycles? DE BCD EDB



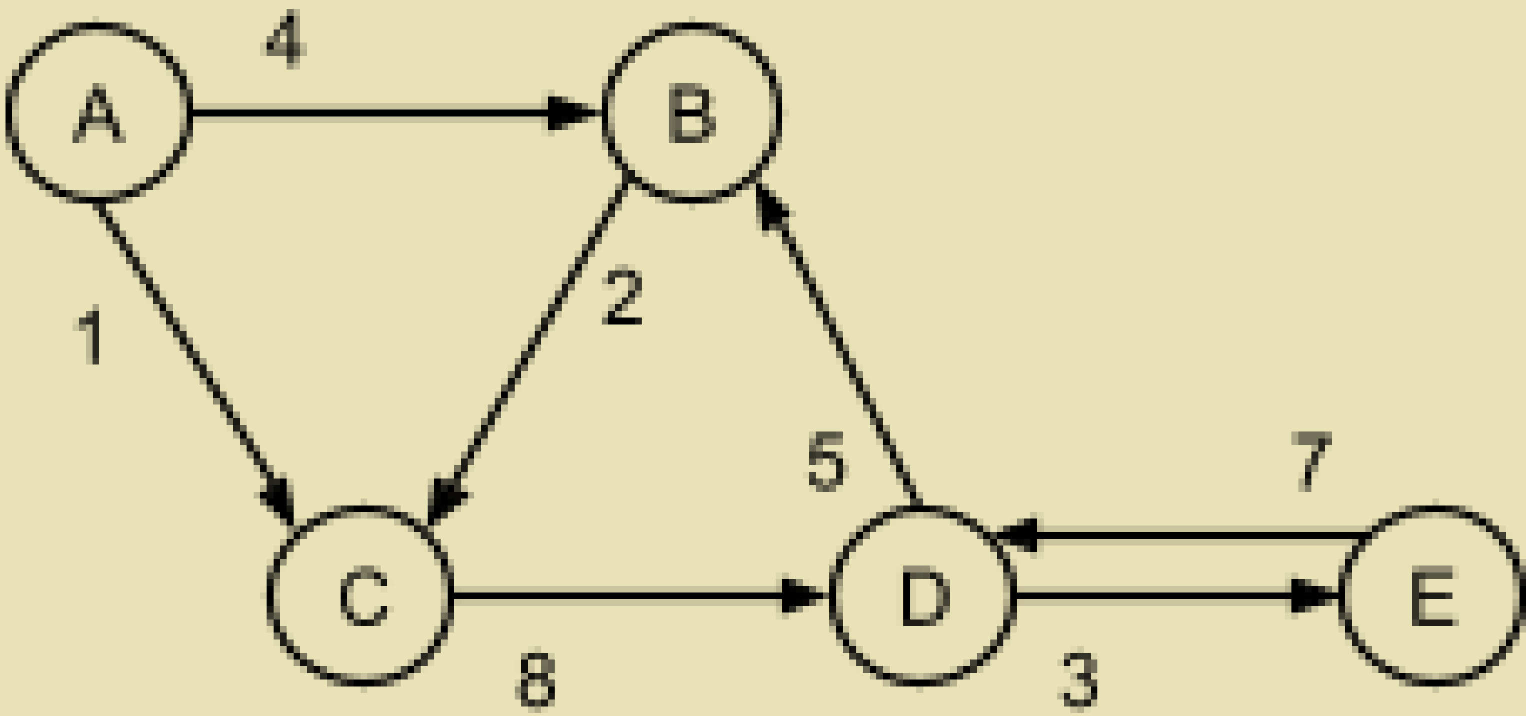
Weighted Graphs

- The **cycle length** is the sum of the edge weights in a cycle.
- A negative edge weight cycle has a cycle length less than 0.
- A shortest path does not exist in a graph with a negative edge weight cycle
  - each loop around the negative edge weight cycle further decreases the cycle length, so no minimum exists.



Weighted Graphs -- Exercise

- Find a path from A to D ACD=9
- Fina the shortest path from A to D.



References and Useful Resources

- Applications of Graph Data Structure https://www.masaischool.com/blog/applications-of-graph-data-structure/

That’s about this lecture!