CS202 - Algorithm Analysis
Graph Algorithms Module-1

Aravind Mohan

Allegheny College

May 7, 2021

Discussion Based On ...

Sedgewick 4.1, 4.2

What is a Graph?

A **Graph** is a data structure that consists of a finite set of vertices(or nodes) and set of edges which connect a pair of nodes. More formally,

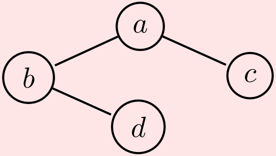
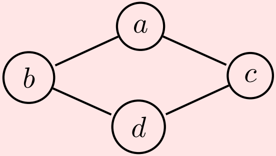
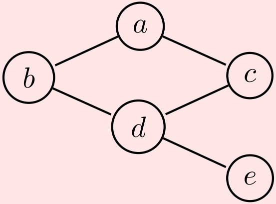
$$G = (V, E)$$

Application: Maps, Electrical Circuit, Facebook Friend List, Human Genetic data, Amazon Product data, and so on $\cdots$

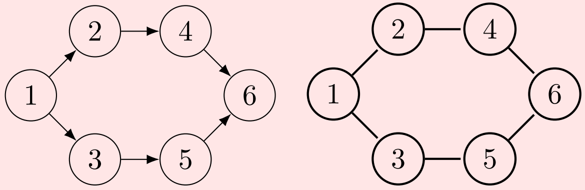
How is a Graph different from a Tree?

Tree	Graph
1. A **Tree** is a specialized case of a **Graph**. It is a connected graph with no circuits and self loops. 2. There should be only one path between any two vertices. 3. A tree does not contain any loops and is minimally connected.	1. A **Graph** consists of vertices, edges, and a set representing relationship between vertices and edges. 2. There can be any number of paths between any two vertices. 3. A Graph contains loops.

Tree Vs Graph

Tree	Graph
	 

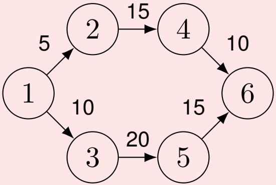
Type of Graphs (1)



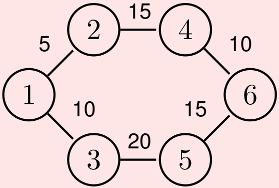
Directed (diGraph)

Undirected

Type of Graphs (2)

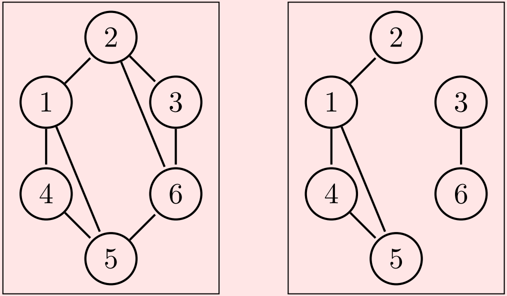


Weighted directed



Weighted undirected

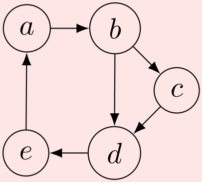
Type of Graphs (3)



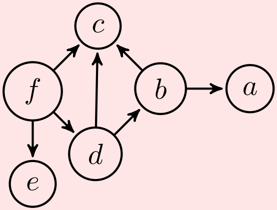
Connected **Disconnected**

A connected **Graph** consists of a path between every two vertices. It is disconnected otherwise.

Type of Graphs (4)



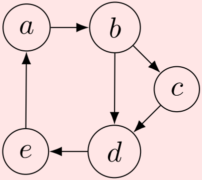
Strongly connected



Weakly connected

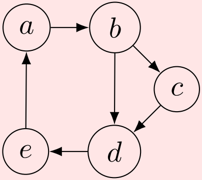
A **diGraph** is a strongly connected if every two vertices are reachable from each other. It is a weakly connected graph if the underlying undirected graph is connected.

Graph - Basic Terminology



- A **Source** vertex in a directed edge is its first endpoint. $Edge(a, b)$ has a source vertex a.
- A **Destination** vertex in a directed edge is its second endpoint (arrow pointed). $Edge(a, b)$ has a destination vertex b.
- A directed edge is said to be **Outgoing** on its source vertex. $Edge(a, b)$ is outgoing on a.
- A directed edge is said to be **Incoming** on its destination vertex. $Edge(a, b)$ is incoming on b.

Graph - Basic Terminology (2)



- **Degree** of a vertex is the total number of edges connected to a vertex. For example, $Deg(b) = 3$ and $Deg(c) = 2$

  $$\forall \text{ vertex, } v \in \text{graph G, } Deg(v) = In(v) + Out(v)$$

- **Indegree** of a vertex is the total number of incoming edges connected to a vertex. $In(b) = 1$ and $In(d) = 2$
- **Outdegree** of a vertex is the total number of outgoing edges connected to a vertex. $Out(b) = 2$ and $Out(d) = 1$

Graph Properties

- Let us suppose a Graph G has V vertices and E edges, then $E < V^2$.
- If E is close to $V \times log(V)$ then the graph is called a **Dense** graph. G is too strongly connected and is in a complete form.
- If $E < V \times log(V)$ then the graph is called a **Sparse** graph.

Graph Representation

So how do we represent a Graph in a Program?

- **Adjacency Matrix**
- **Adjacency List**

- Adjacency Matrix is used primarily to represent a Dense graph.
- Adjacency List is used primarily to represent a Sparse graph.

Graph Representation

Adjacency Matrix

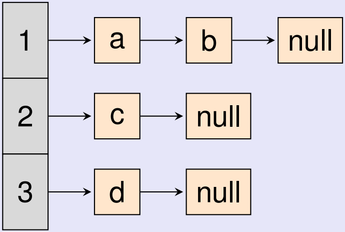
$$\begin{array}{cc} \begin{array}{ccccc} a & b & c & d & e \end{array} & \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} \end{array}$$

Space Complexity = $O(V^2)$

$$\begin{pmatrix} a[i][j] = 1 & \text{if i and j are adjacent} \\ = 0 & \text{otherwise} \end{pmatrix}$$

Graph Representation

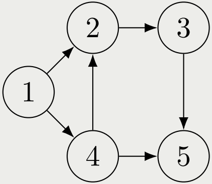
Adjacency List



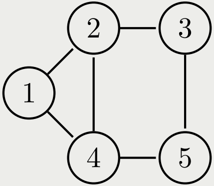
Space Complexity = $O(V + 2E)$

An array of lists of vertices. The list item **(i)** contains vertex **(j)** if there exists an $Edge(i, j)$ in Graph G.

Graph Representation - Example 1

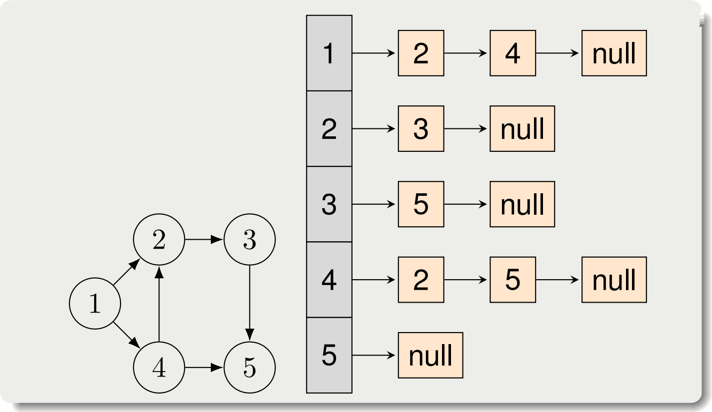


$$\begin{array}{cc} \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} & \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \end{array}$$

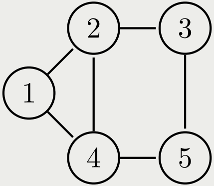


$$\begin{array}{cc} \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} & \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \end{array}$$

Graph Representation - Example 2



Graph Representation - Example 3



1 → 2 → 4 → null

2 → 1 → 3 → 4 → null

3 → 2 → 5 → null

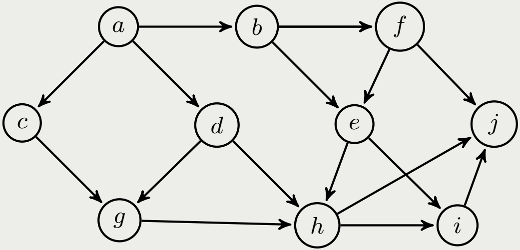
4 → 1 → 2 → 5 → null

5 → 3 → 4 → null

Graph - Exercise

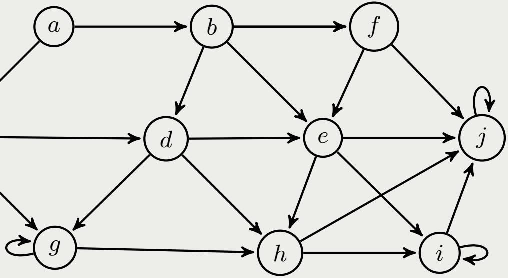
Try out 1

- **Identify** the in-degree and out-degree of all vertices (a to j) in the Graph provided below:



Try out 2

- **Draw** the adjacency matrix and adjacency list representation of the Graph provided below:



Next:

- **Graph Traversal Algorithms:**
  BFS and DFS
- **Graph Shortest Path Algorithms:**
  Dijkstras algorithm.

Reading Assignment

Sedgewick 4.1, 4.2

Questions?

Please ask if there are any Questions!