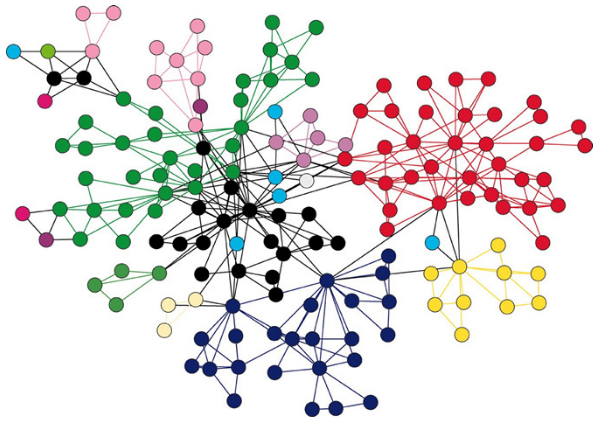
UNIVERSITY OF DIYALA

Data Structure

Lecture 6: The Graph

Asst. Instructor

Ali A. Al-Ani



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Graph

- *A Graph is a non-linear data structure consisting of nodes and **edges**. The nodes are sometimes also referred to as **vertices** and the edges are lines or arcs that connect any two nodes in the graph.*
- *Graphs provide the ultimate in data structure flexibility. Its can model both real-world systems, so they are used in hundreds of applications.*
- *Since they are powerful abstractions, graphs can be very important in modeling data. In fact, Graphs have applications in a host of different domains, including:*

1. ***Social network graphs:*** *to tweet or not to tweet. Graphs that represent who knows whom, who communicates with whom, who influences whom or other relationships in social structures.*

Department of Computer Science
College of Science

Graph

2. ***Transportation networks.*** *In road networks* ***vertices*** *are intersections and* ***edges*** *are the road segments between them, and for public transportation networks* ***vertices*** *are stops and* ***edges*** *are the links between them. Such networks are used by many map programs such as* ***Google maps*** *and* ***GPS*** *maps.*
3. ***Utility graphs.*** *The power grid, the Internet, and the water network are all examples of graphs where* ***vertices*** *represent connection points, and* ***edges*** *the wires or pipes between them.*
4. ***Document link graphs.*** *The best known example is the link graph of the web, where each web page is a vertex, and each hyperlink a directed edge.*

UNIVERSITY OF DIYALA

Graph: Formal Def.

- ***Graph****: A graph consists of a set of vertices* ***V*** *and a set of edges* ***E****, The set of edges describes relationships among the vertices such that each edge in* ***E*** *is a connection between a pair of vertices in* ***V****.*
- *A graph* ***G*** *is defined as follows:*

$$G = (V, E)$$

$V(G)$: *a finite, nonempty set of vertices.*

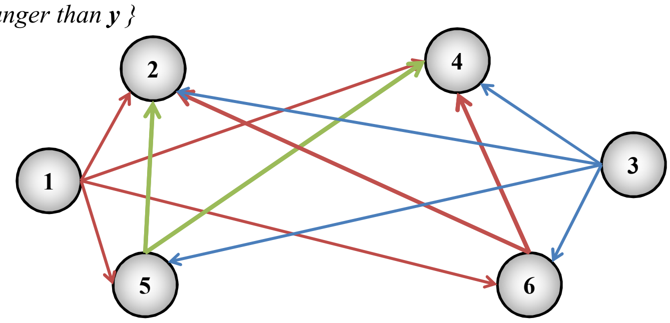
$E(G)$: *a set of edges (pairs of vertices).*

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Graph: Example_1

- ***A "Real-life" Example of a Graph:*** *V = set of 6 people: Ali(1), Ahmed(2), Tarq(3), Youisf(4), Suha(5), and Saja(6), of ages 12, 15, 12, 15, 13, and 13, respectively.*
- ***E*** *={* ***(x ,y)*** *| if x is younger than* ***y*** *}*

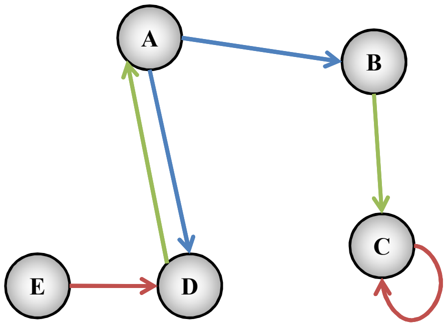


Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Graph: Example_2

- ***V*** *= { A, B, C, D, E}*
- ***E*** *= { (A,B), (B,C), (A,D), (D,A), (C,C), (E,D) }*
- *When (* ***x, y****) is an edge,*
- *we say that* ***x*** *is adjacent to* ***y****, and* ***y*** *is adjacent from* ***x****.*
- ***A*** *is adjacent to* ***B****.*
- ***B*** *is not adjacent to* ***A****.*
- ***C*** *is adjacent from* ***B****.*

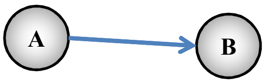


Department of Computer Science
College of Science

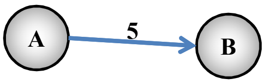
The Graph Terminology

1. ***Adjacent nodes**: two nodes are adjacent if they are connected by an edge*

Ex:** **A** is adjacent to **B.



2. ***Path:** a sequence of vertices that connect two or more nodes in a graph.*
3. ***Weighted graph:** a graph in which each edge carries a value*



4. ***Complete graph:** a graph in which every vertex is directly connected to every other vertex.*

UNIVERSITY OF DIYALA

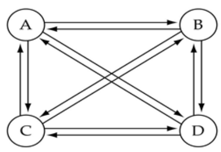
Graph Terminology

1. *We can compute the number of edges in a complete directed graph with **N** vertices as follow:*

No. of E = N * (N-1)

No. of E = 4 * (4-1)

No. of E = 12

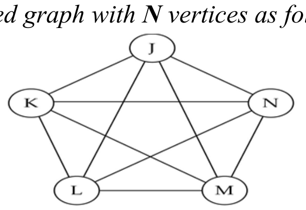


2. *Also we can compute the number of edges in a complete undirected graph with **N** vertices as follow:*

No. of E = N * (N-1) / 2

No. of E = 5* (5-1) / 2

No. of E = 10



Department of Computer Science
College of Science

Directed vs. Undirected Graphs

1. ***Directed graph (digraph)***

- *When the edges in a graph have a direction, the graph is called directed (or digraph).*
- *Let **G** be a directed graph*
  1. *The **Indegree** of a node **x** in **G** is the number of edges coming to **x***
  2. *The **Outdegree** of **x** is the number of edges leaving **x**.*

2. ***Undirected graph***

- *When the edges in a graph have no direction, the graph is called undirected*

UNIVERSITY OF DIYALA

Graph Representation

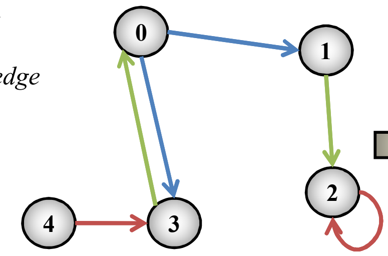
- *For graphs to be computationally useful, they have to be conveniently represented in programs. There are two computer representations of graphs:*
  1. ***Adjacency matrix representation.***
  2. ***Adjacency lists representation.***
- *Basic Operations: Following are basic primary operations of a Graph:*
  1. ***Add Vertex: Adds a vertex to the graph.***
  2. ***Add Edge: Adds an edge between the two vertices of the graph.***
  3. ***Display Vertex: Displays a vertex of the graph.***

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Adjacency Matrix Representation

- *In this representation, each graph of n nodes is represented by an n x n matrix A, that is, a two-dimensional array A:*
- *The nodes are (re)-labeled* ***0,1,2,…,n***
- ***A [i] [j] = 1*** *if (i,j) is an edge*
- ***A [i] [j] = 0*** *if (i,j) is not an edge*



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

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Adjacency Matrix Representation

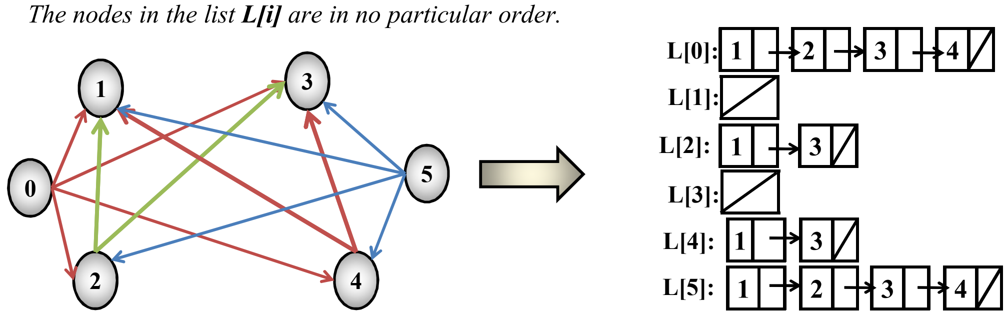
- ***Advantage****:*
  - *Simple to implement.*
  - *Easy and fast to tell if a pair (i, j) is an edge: simply check if A[i][j] is 1 or 0*
- ***Disadvantage:***
  - *No matter how few edges the graph has, the matrix takes* $\boldsymbol{O(n^2)}$ *in memory.*

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Adjacency Lists Representation

- *A graph of **n** nodes is represented by a one-dimensional array **L** of linked lists, where:*
  - ***L [i]** is the linked list containing all the nodes adjacent from node **i**.*
  - *The nodes in the list **L[i]** are in no particular order.*



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Adjacency Lists Representation

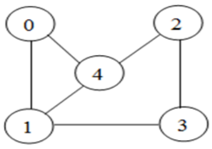
- ***Advantage:***
  - *Saves on space (memory): the representation takes as many memory words as there are nodes and edge.*
- ***Disadvantage:***
  - *It can take up to **O(n)** time to determine if a pair of nodes **(i, j)** is an edge: one would have to search the linked list **L[i]**, which takes time proportional to the length of **L[i]**.*

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Undirected Representation

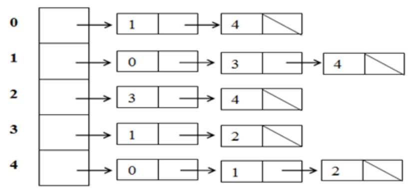
- ***Representations of the Undirected Graphs:*** *The same two representations for directed graphs can be used for undirected graphs*
- ***Adjacency matrix A:*** *A [i] [j]=1 if (i,j) is an edge; 0 otherwise*
- ***Adjacency Lists:*** *L [i] is the linked list containing all the neighbors of i*



(a)

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

(b)



(c)

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Graph Traversal Techniques

- ***There are two standard graph traversal techniques:***
  - ***Depth-First Search (DFS)***
  - ***Breadth-First Search (BFS)***
- *In both **DFS** and **BFS**, the nodes of the undirected graph are visited in a systematic manner so that every node is visited exactly one.*
- *Both **BFS** and **DFS** give rise to a tree:*
  - *When a node **x** is visited, it is labeled as visited, and it is added to the tree.*
  - *If the traversal got to node **x** from node **y**, **y** is viewed as the parent of **x**, and **x** a child of **y.***

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

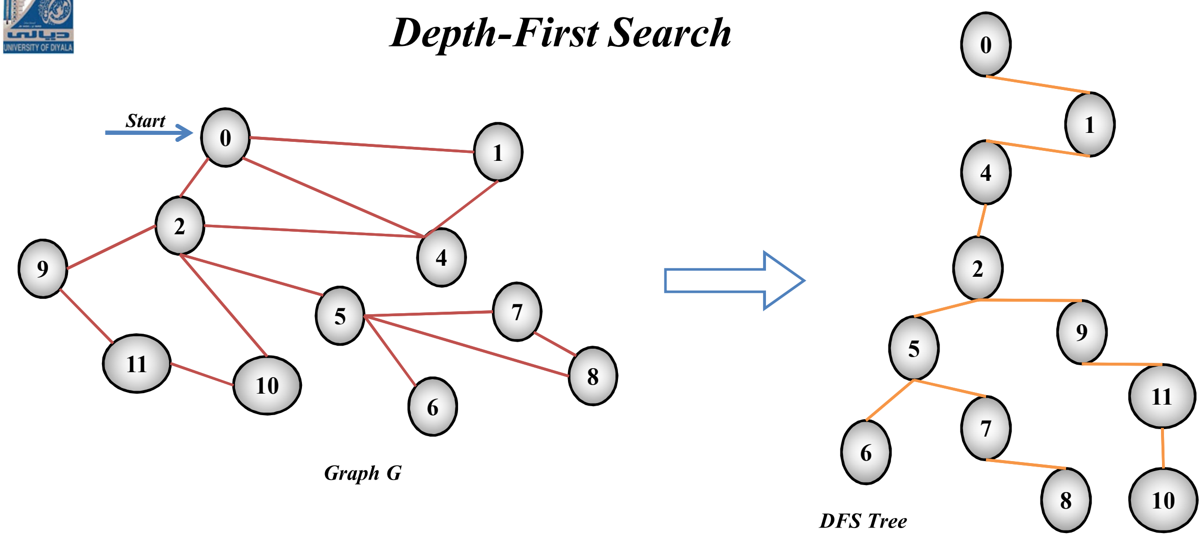
Depth-First Search

- ***DFS follows the following rules:***
  1. *Select an unvisited node **x**, visit it, and treat as the current node .*
  2. *Find an unvisited neighbor of the current node, visit it, and make it the new current node;*
  3. *If the current node has no unvisited neighbors, backtrack to the its parent, and make that parent the new current node;*
  4. *Repeat steps 3 and 4 until no more nodes can be visited.*
  5. *If there are still unvisited nodes, repeat from step 1.*

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Depth-First Search



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

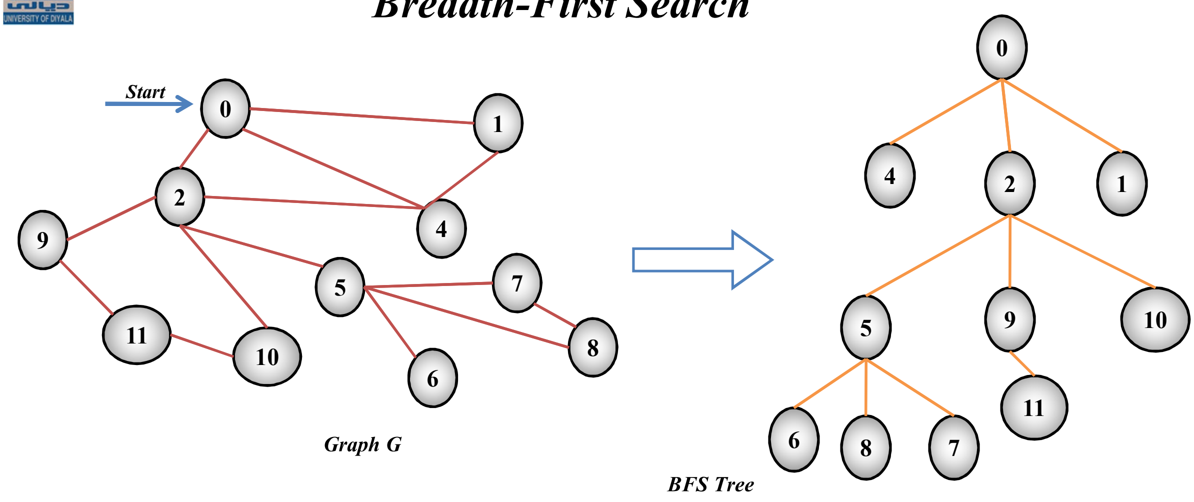
Breadth-First Search

- ***BFS follows the following rules:***
  1. *Select an unvisited node **x**, visit it, have it be the root in a BFS tree being formed. Its level is called the current level.*
  2. *From each node **z** in the current level, in the order in which the level nodes were visited, visit all the unvisited neighbors of **z**. The newly visited nodes from this level form a new level that becomes the next current level.*
  3. *Repeat step 2 until no more nodes can be visited.*
  4. *If there are still unvisited nodes, repeat from Step 1.*

Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

Breadth-First Search



Department of Computer Science
College of Science

UNIVERSITY OF DIYALA

The End

Department of Computer Science
College of Science