كلية علوم الحاسوب والرياضيات
College of Computer Science & Mathematics

Software Engineering Dept.- Second year

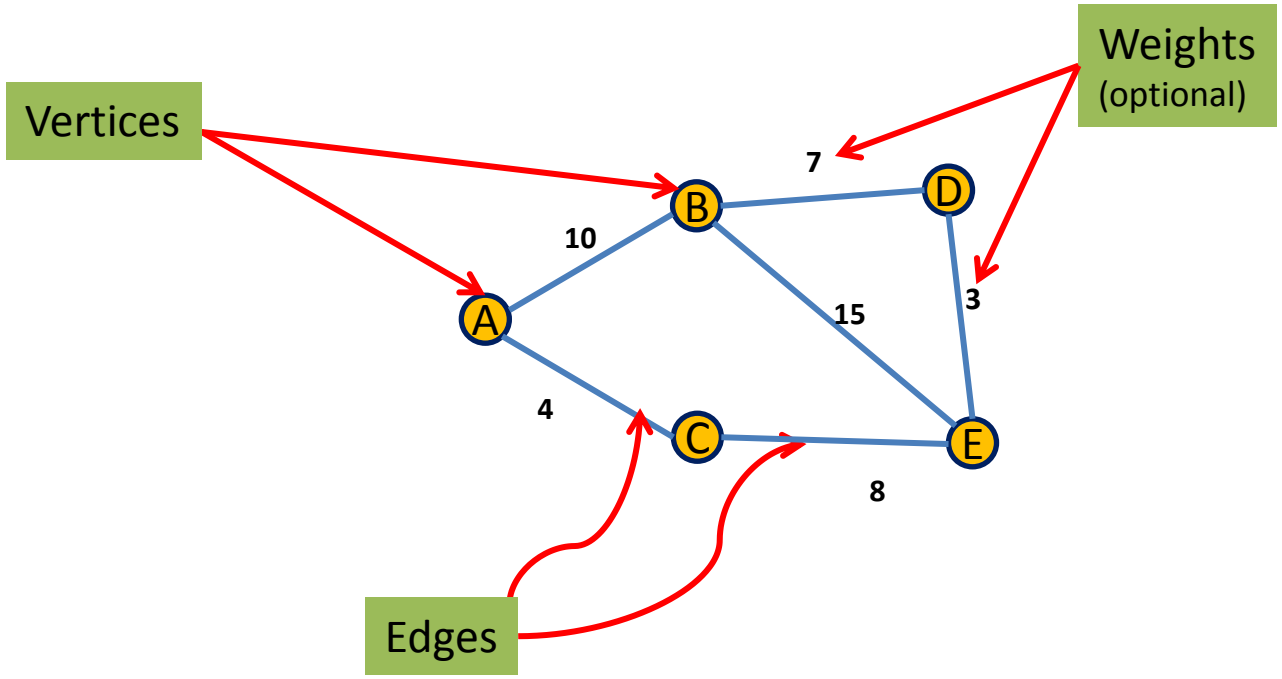
Graph Data Structure

by

Dr.laheeb M. Ibrahim

Graphs

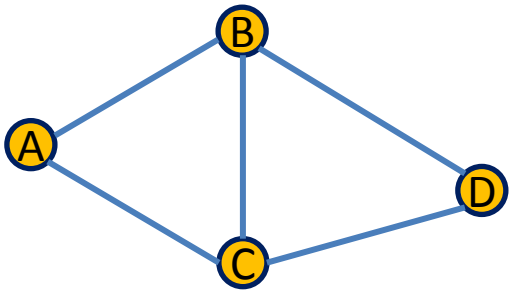
A ***graph*** *G* is defined as *G = (V, E).*
Here, *V* is a set of vertices and *E* is a set of edges. Edges may or may not have weights.



And edge connects two vertices and can be denoted by it's two endpoints, e.g., (A,C). We have, $E \subseteq V \times V$.

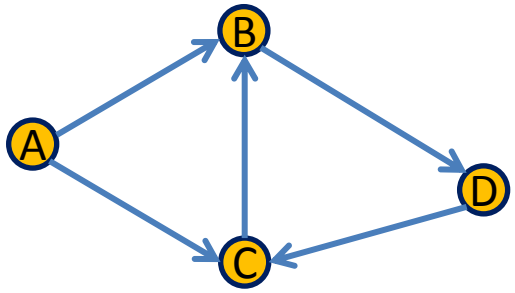
Classifications of graphs

Undirected Graph



V = {A, B, C, D}
E = { (A, B), (A,C),
(B,D), (B,C), (C, D) }

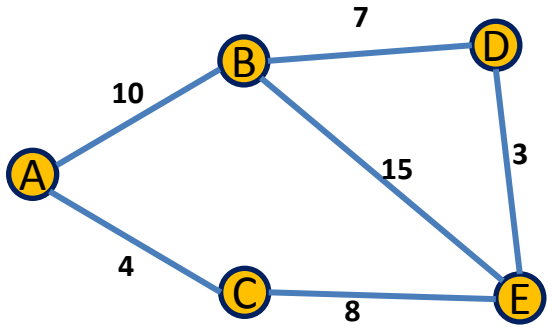
Directed Graph

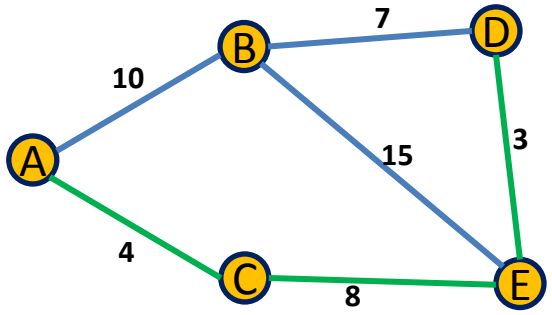


V = {A, B, C, D}
E = { (A, B), (A,C),
(B,D), (C,B), (D, C) }

Why Graphs?

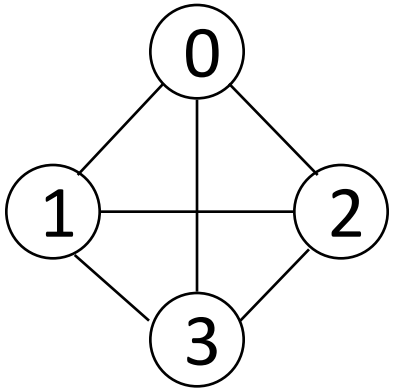
For example, given a network of cities and roads connecting them, what is the shortest path between two given cities?



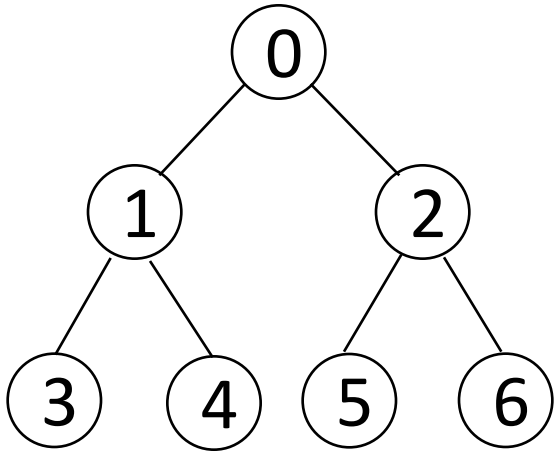


Shortest path between A and D?

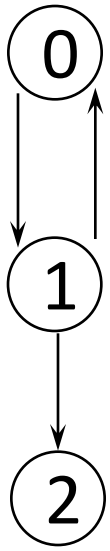
Examples for Graph



G_1



G_2



G_3

$V(G_1)=\{0,1,2,3\}$ $E(G_1)=\{(0,1),(0,2),(0,3),(1,2),(1,3),(2,3)\}$

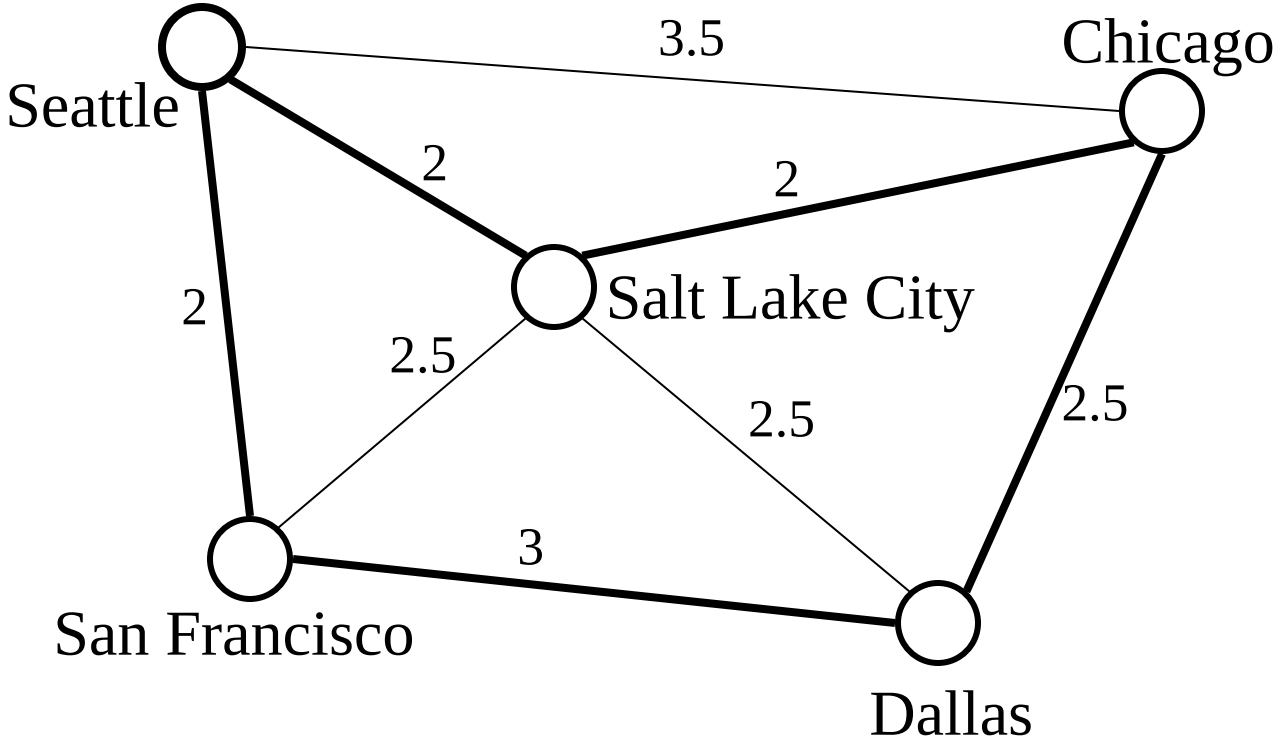
$V(G_2)=\{0,1,2,3,4,5,6\}$ $E(G_2)=\{(0,1),(0,2),(1,3),(1,4),(2,5),(2,6)\}$

$V(G_3)=\{0,1,2\}$ $E(G_3)=\{<0,1>,<1,0>,<1,2>\}$

Path Length and Cost

Path length: the number of edges in the path

Path cost: the sum of the costs of each edge

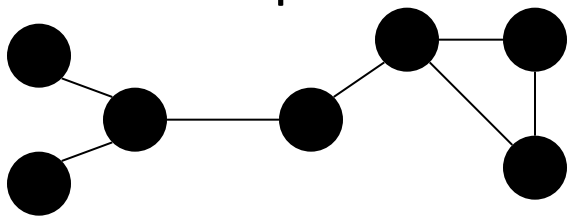


length(p) = 5

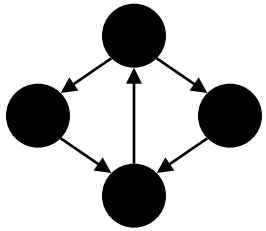
cost(p) = 11.5

Connectivity

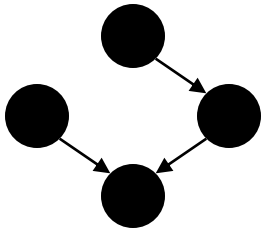
Undirected graphs are *connected* if there is a path between any two vertices



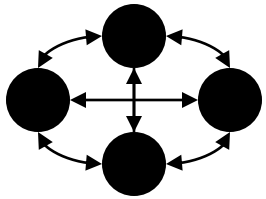
Directed graphs are *strongly connected* if there is a path from any one vertex to any other



Directed graphs are *weakly connected* if there is a path between any two vertices, *ignoring direction*



A *complete* graph has an edge between every pair of vertices



Some Graph Operations

- Traversal

  Given G=(V,E) and vertex v, find all $w \in V$, such that w connects v.

  – Depth First Search (DFS)
  preorder tree traversal

  – Breadth First Search (BFS)
  level order tree traversal

Depth-First Search

- In depth-first search
  - Nodes are visited deeply on the left-most branches before any nodes are visited on the right-most branches
- In DFS the nodes “being worked on” are kept on a stack

Iterative Version DFS

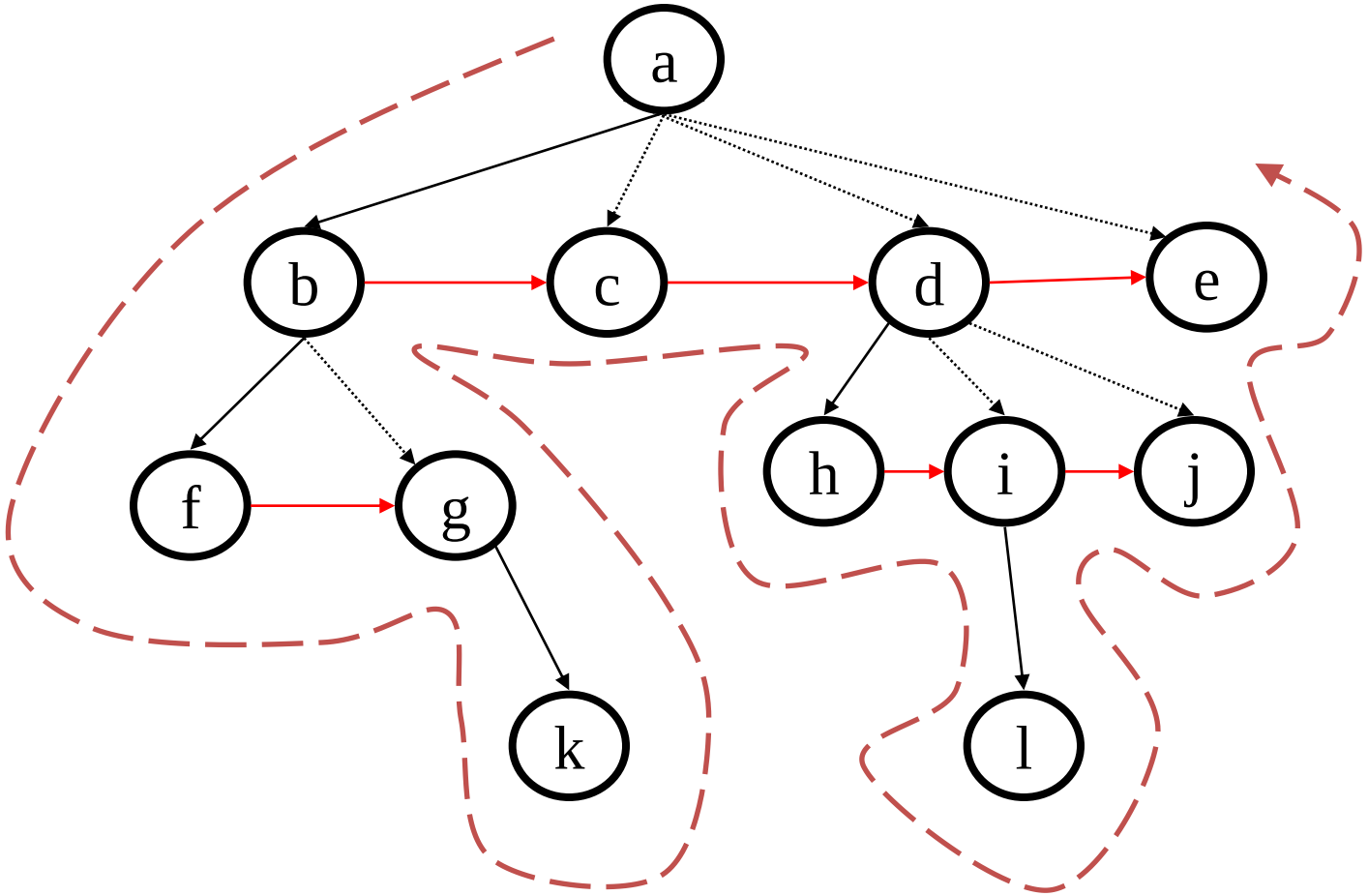
Pre-order Traversal

Push root on a Stack

Repeat until Stack is empty:

- Pop a node
- Process it
- Push it’s children on the Stack

Recall: Tree Traversals



a b f g k c d h i l j e

Breadth-First Search

- Level-order traversal is an example of Breadth-First Search
- BFS characteristics
  - Nodes being worked on maintained in a FIFO Queue, not a stack
  - Iterative style procedures often easier to design than recursive procedures

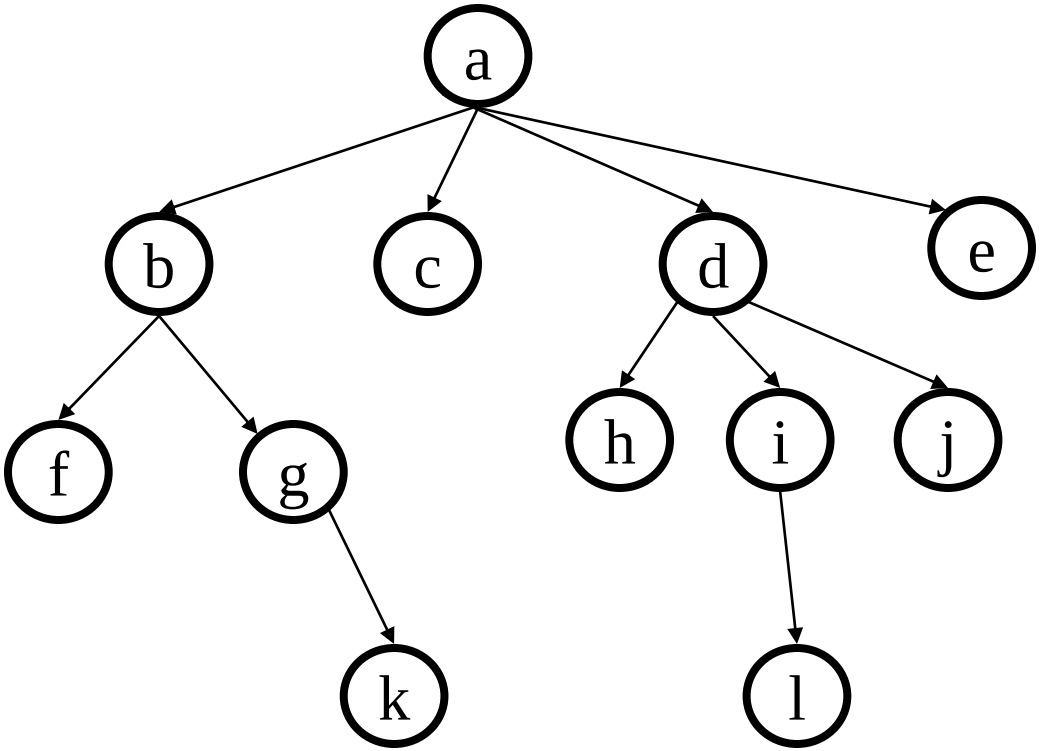
    Put root in a Queue

    Repeat until Queue is empty:

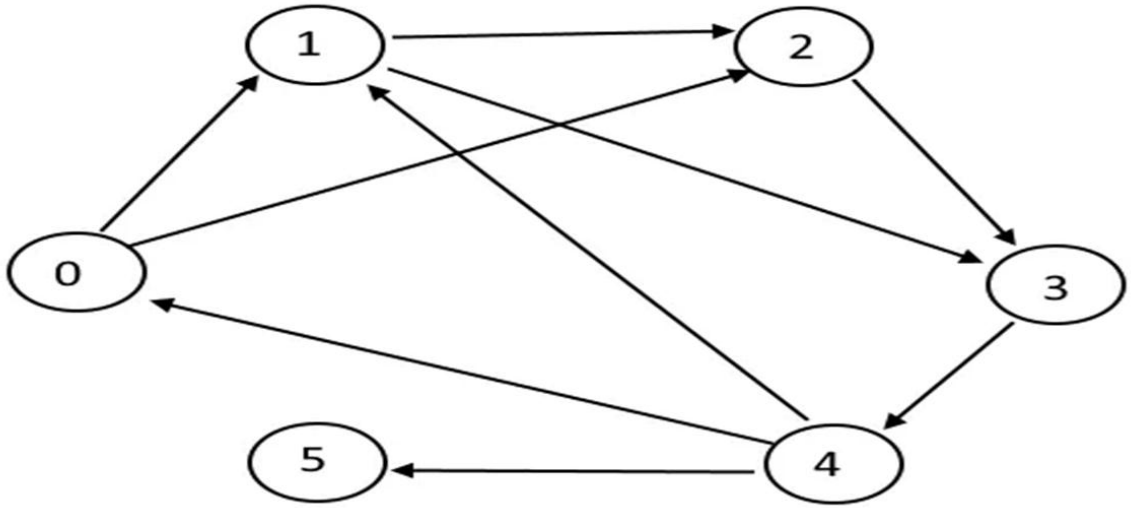
        Dequeue a node

        Process it

        Add it's children to queue


a
b
c
d
e
f
g
h
i
j
k
l

Graph



BFS starting from Node 0

0 2 1 3 4 5

Is BFS the Hands Down Winner?

- Depth-first search
  - Simple to implement (implicit or explict stack)
  - Does not always find shortest paths
- Breadth-first search
  - Simple to implement (queue)
  - Always finds shortest paths