Graph Data Structures

261217 Data Structures for Computer Engineers

Patiwet Wuttisarnwattana, Ph.D.

patiwet@eng.cmu.ac.th

Computer Engineering, Chiang Mai University

Graphs

- Tree is a data structure that consists of nodes that connects to other nodes but there is no loop/cycle.
- Graph is a data structure that consists of nodes that connects to other nodes but loop/cycle is allowed.
- Tree a subset of Graph.

Graphs

- Graphs are used to represent many real life applications.
- Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network. Graphs are also used in social networks like linkedIn, facebook.
- For example, in facebook, each person is represented with a vertex(or node). Each node is a structure and contains information like person id, name, gender and locale. See this for more applications of graph.

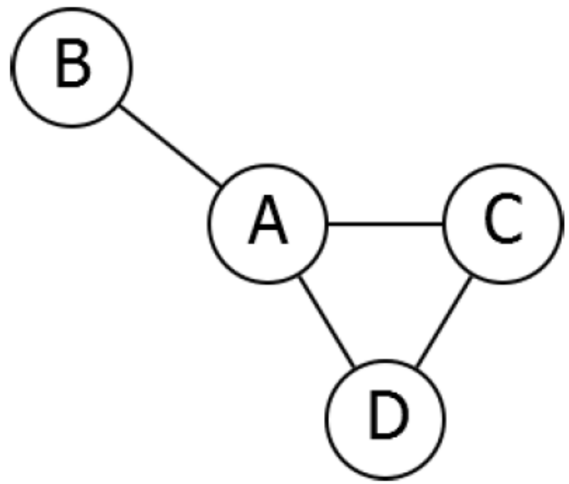
Undirected Graph

Definition

An (undirected) Graph is a collection V of vertices, and a collection E of edges each of which connects a pair of vertices.

Drawing Graphs

Vertices: Points. Edges: Lines.

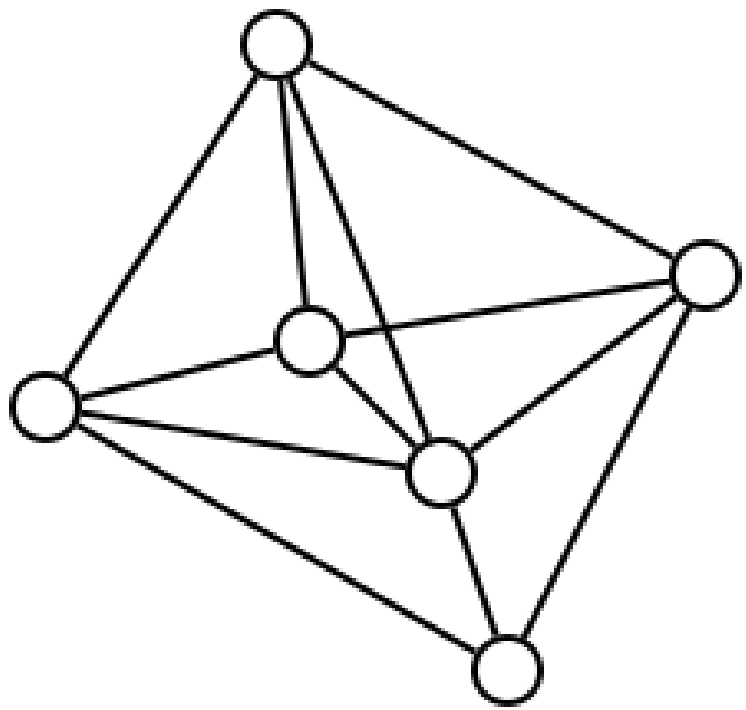


Vertices: A, B, C, D

Edges: (A, B), (A, C), (A, D), (C, D)

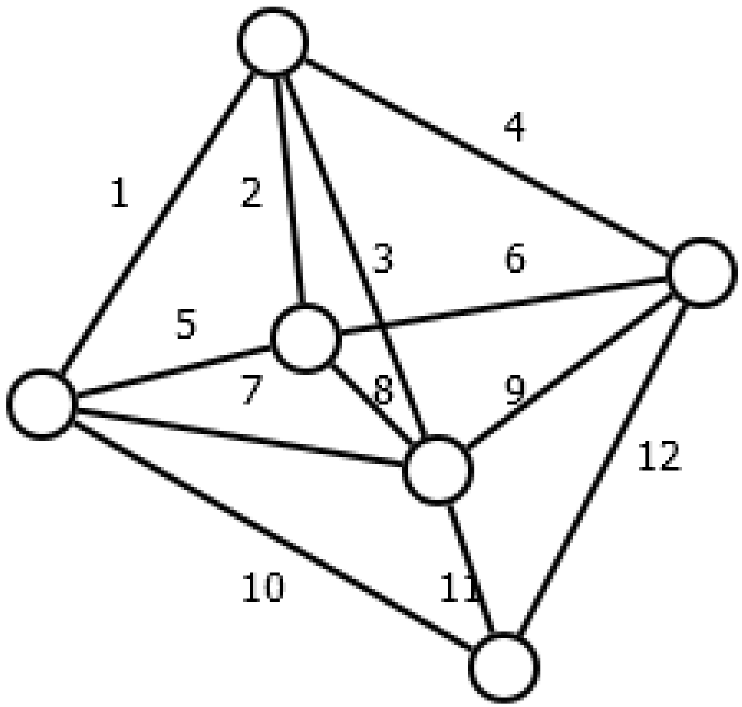
Problem

- How many edges are in the graph given below?



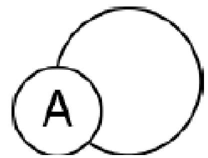
Answer

- 12

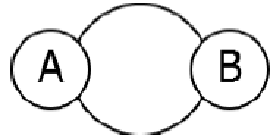


Loops and Multiple Edges

Loops connect a vertex to itself.



Multiple edges between same vertices.



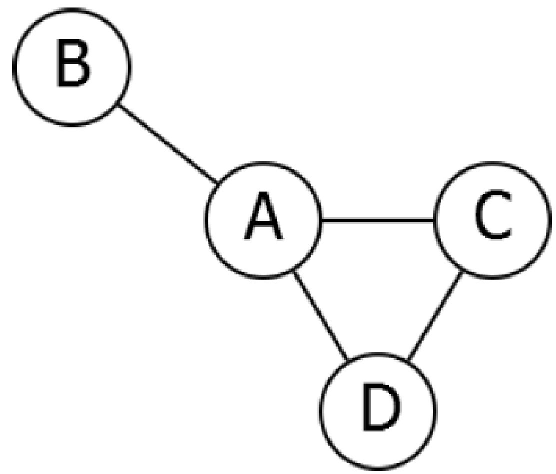
If a graph has neither, it is simple (Tree)

Graph Representation (Implementation)

1. Edge List
2. Adjacency Matrix
3. Adjacency List

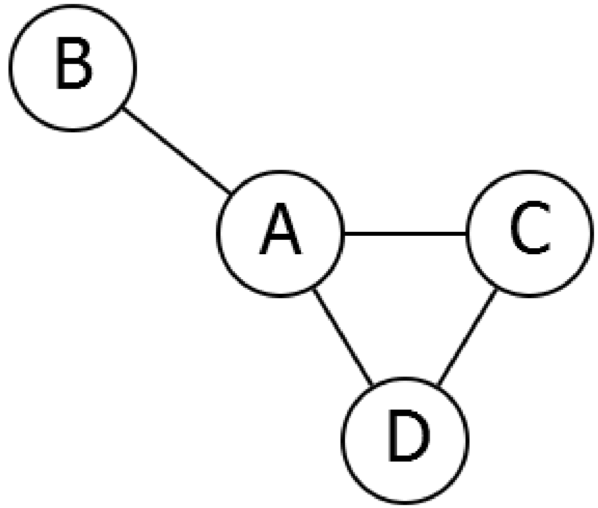
Edge List

List of all edges:

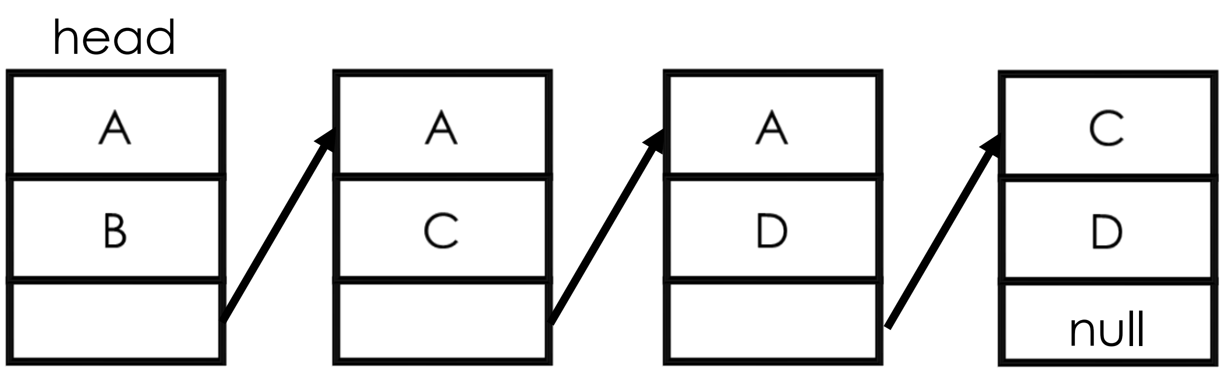


Edges: $(A, B), (A, C), (A, D), (C, D)$

Edge List

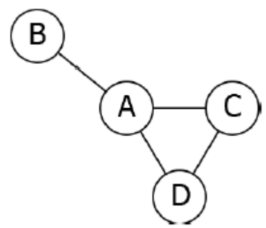


Each node represents edge



Adjacency Matrix

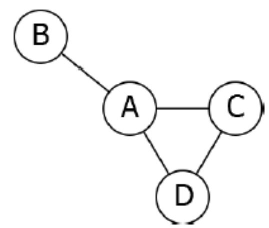
Matrix. Entries 1 if there is an edge, 0 if there is not.



$$\begin{array}{ccccc} & A & B & C & D \\ A & 0 & 1 & 1 & 1 \\ B & 1 & 0 & 0 & 0 \\ C & 1 & 0 & 0 & 1 \\ D & 1 & 0 & 1 & 0 \end{array}$$

Adjacency Matrix

Matrix. Entries 1 if there is an edge, 0 if there is not.



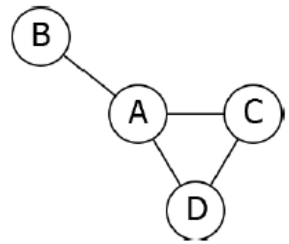
Map index to Object (Vertex)

0	*A*
1	*B*
2	*C*
3	*D*

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

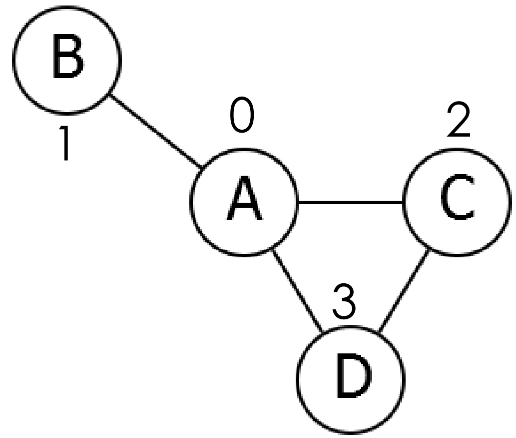
Adjacency List

For each vertex, a list of adjacent vertices.



A adjacent to B, C, D
B adjacent to A
C adjacent to A, D
D adjacent to A, C

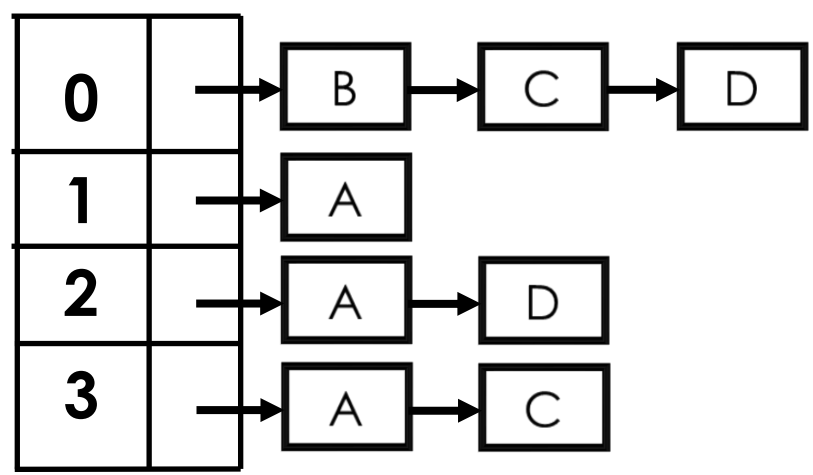
Adjacency List (Array of Linked List)



Map index to Object (Vertex)

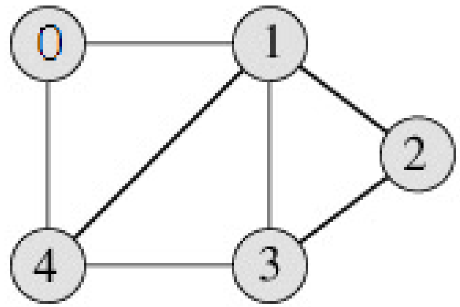
0	**A**
1	**B**
2	**C**
3	**D**

Array of Linked List



Example

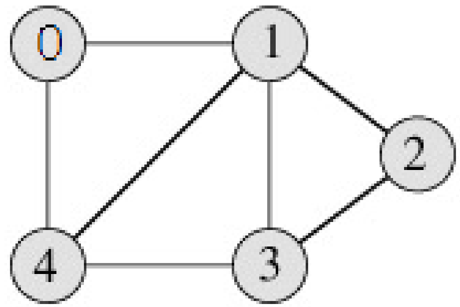
Graph of numbers



What are Adjacency Matrix Representation of the graph?
What are Adjacency List Representation of the graph?

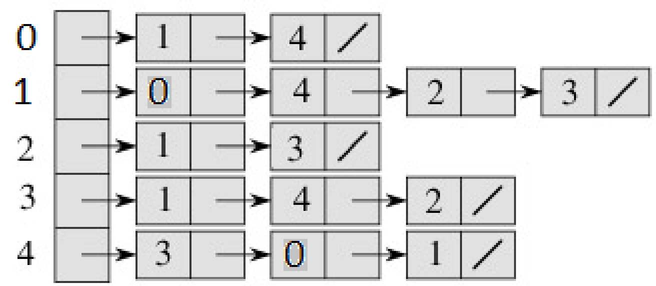
Example

Graph of numbers



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

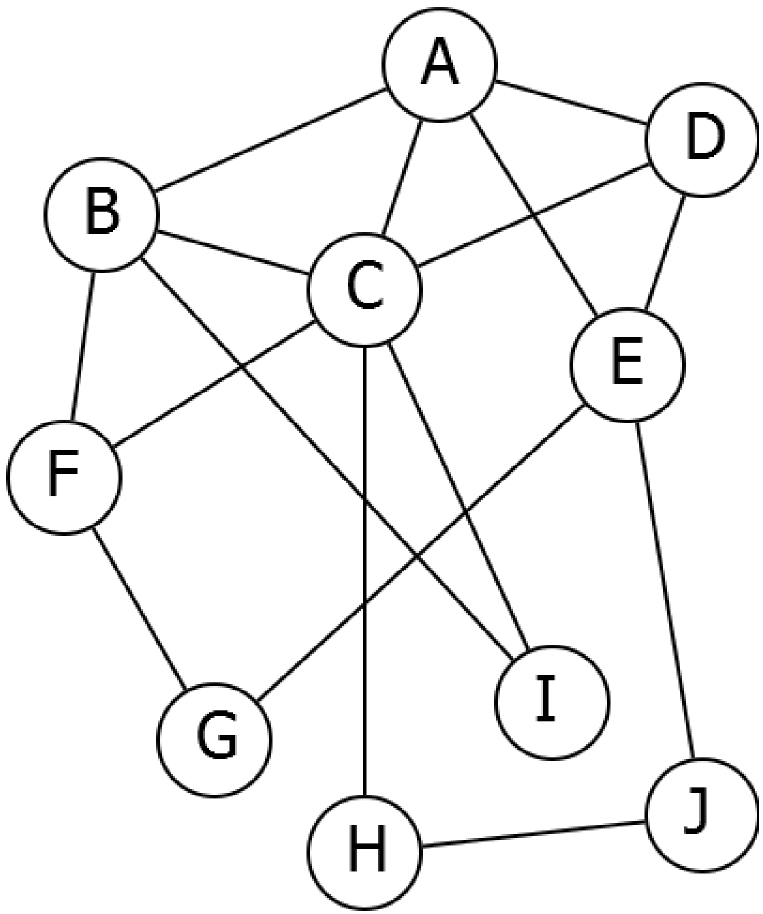
Adjacency Matrix



Adjacency List

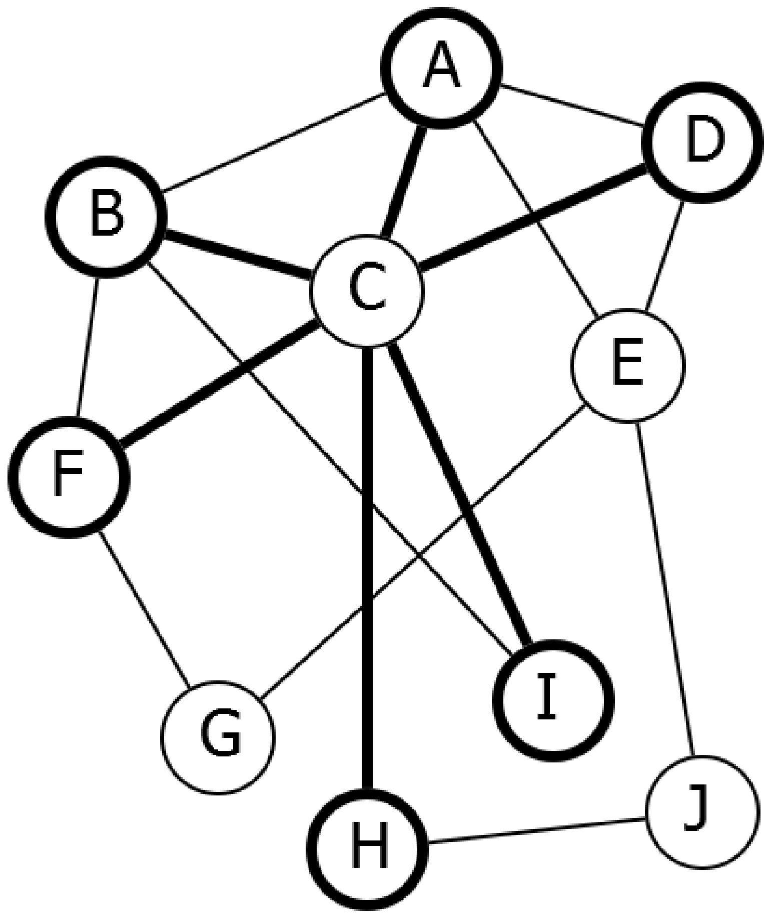
Problem

- What are the neighbors of C?



Problem

- A, B, D, F, H, I



Summary

Different operations are faster in different representations.

Op	hasEdge(V1, V2)	find all Neighbors(V)	Memory
Edge List	$O(\|E\|)$	$O(\|E\|)$	$O(\|E\|)$
Adj Matrix	$O(1)$	$O(\|V\|)$	$O(\|V\|^2)$
Adj. List	$O(\|V\|)$	$O(\|V\|)$	$O(\|V\|+\|E\|)$
Adj. List (if deg << $\|V\|$)	O(deg)	O(deg)	$O(\|V\|+\|E\|)$

For many problems, want adjacency list.

Algorithm Runtimes

Graph algorithm runtimes depend on $|V|$ and $|E|$.

For example, $O(|V| + |E|)$ (linear time), $O(|V||E|)$, $O(|V|^{3/2})$, $O(|V|\log(|V|) + |E|)$.

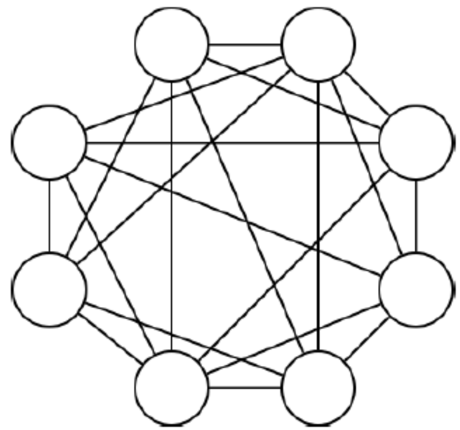
Density

Which is faster, $O(|V|^{3/2})$ or $O(|E|)$?

Depends on graph! Depends on the density, namely how many edges you have in terms of the number of vertices.

Dense Graphs

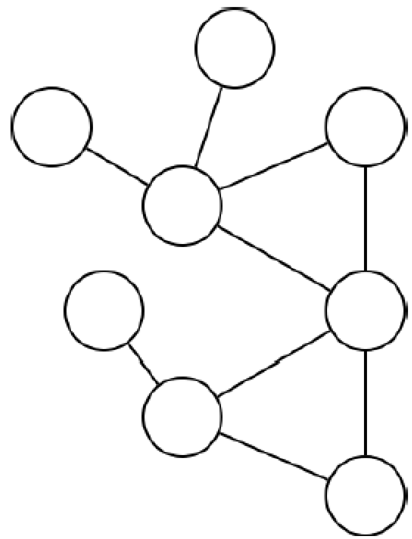
In dense graphs, $|E| \approx |V|^2$.



A large fraction of pairs of vertices are connected by edges.

Sparse Graphs

In sparse graphs, $|E| \approx |V|$.



Each vertex has only a few edges.