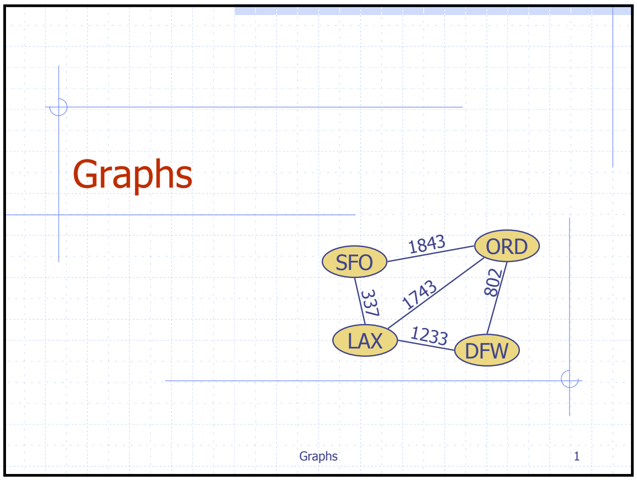
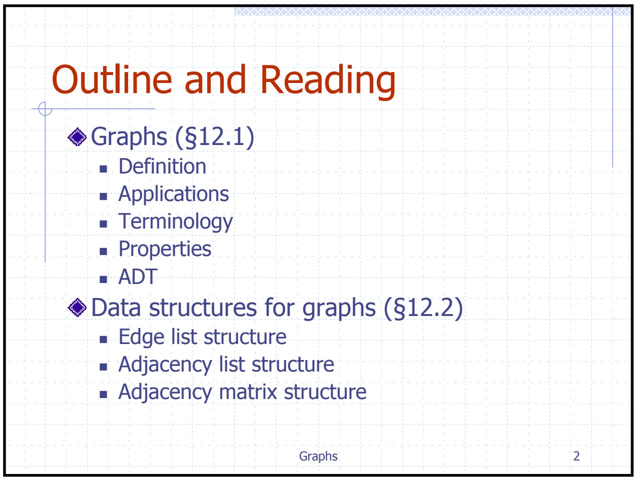


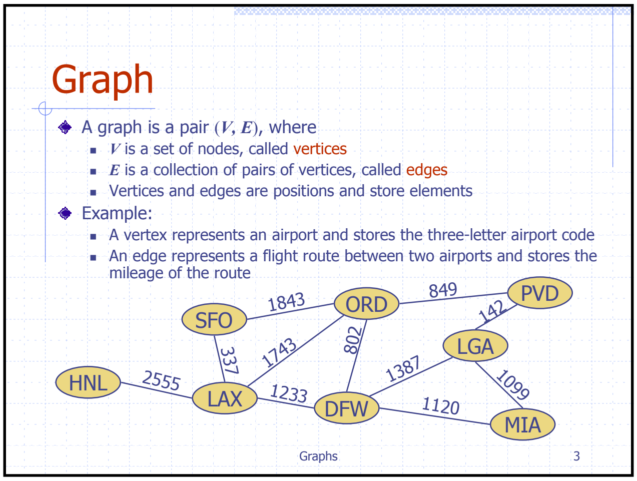
Graphs

SFO
1843
ORD
337
1743
802
LAX
1233
DFW



Outline and Reading

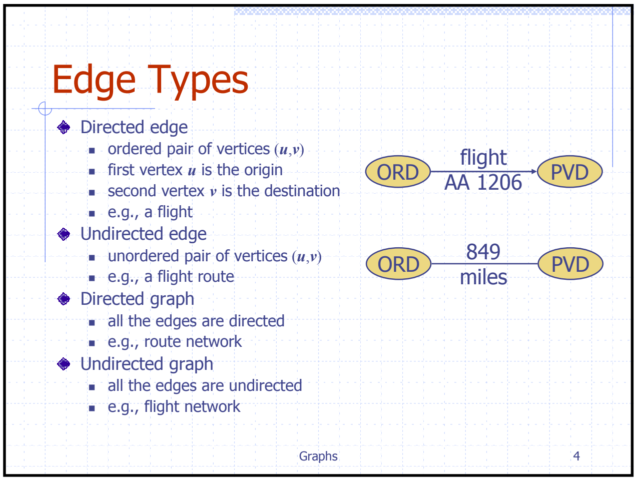
- Graphs (§12.1)
  - Definition
  - Applications
  - Terminology
  - Properties
  - ADT
- Data structures for graphs (§12.2)
  - Edge list structure
  - Adjacency list structure
  - Adjacency matrix structure



Graph

- A graph is a pair (V, E), where
  - V is a set of nodes, called vertices
  - E is a collection of pairs of vertices, called edges
  - Vertices and edges are positions and store elements
- Example:
  - A vertex represents an airport and stores the three-letter airport code
  - An edge represents a flight route between two airports and stores the mileage of the route

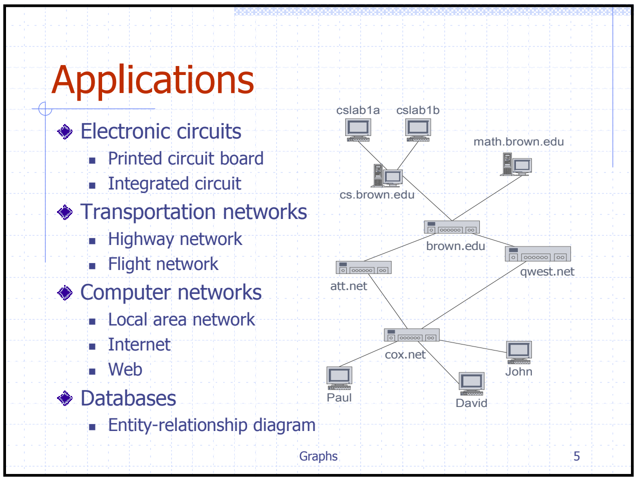
SFO
1843
ORD
849
PVD
142
337
1743
802
LGA
HNL
2555
LAX
1233
DFW
1387
1099
1120
MIA



Edge Types

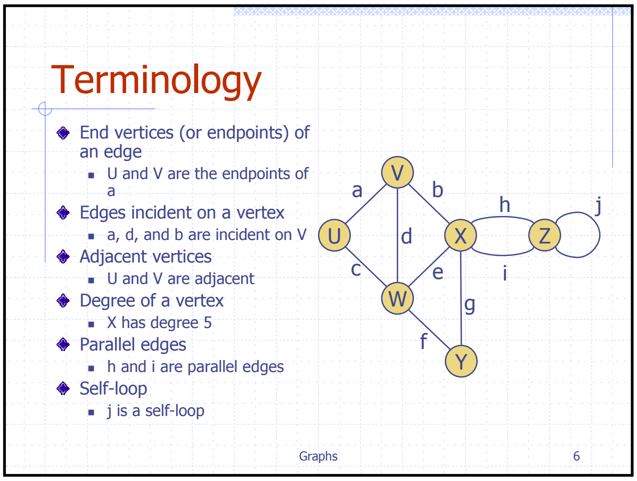
- Directed edge
  - ordered pair of vertices (u,v)
  - first vertex u is the origin
  - second vertex v is the destination
  - e.g., a flight
- Undirected edge
  - unordered pair of vertices (u,v)
  - e.g., a flight route
- Directed graph
  - all the edges are directed
  - e.g., route network
- Undirected graph
  - all the edges are undirected
  - e.g., flight network

ORD
flight
AA 1206
PVD
ORD
849
miles
PVD



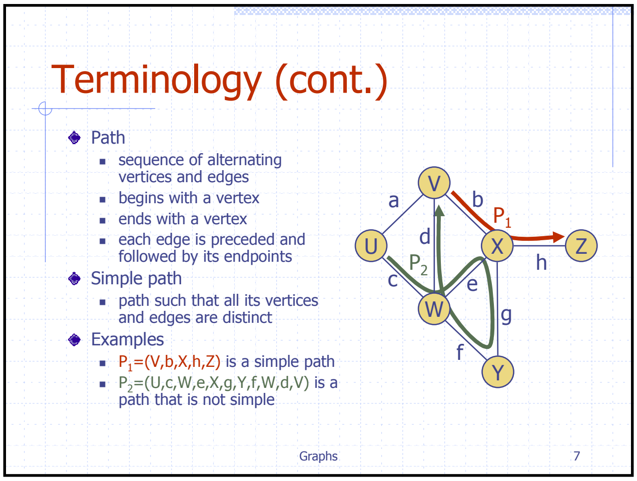
Applications

- Electronic circuits
  - Printed circuit board
  - Integrated circuit
- Transportation networks
  - Highway network
  - Flight network
- Computer networks
  - Local area network
  - Internet
  - Web
- Databases
  - Entity-relationship diagram



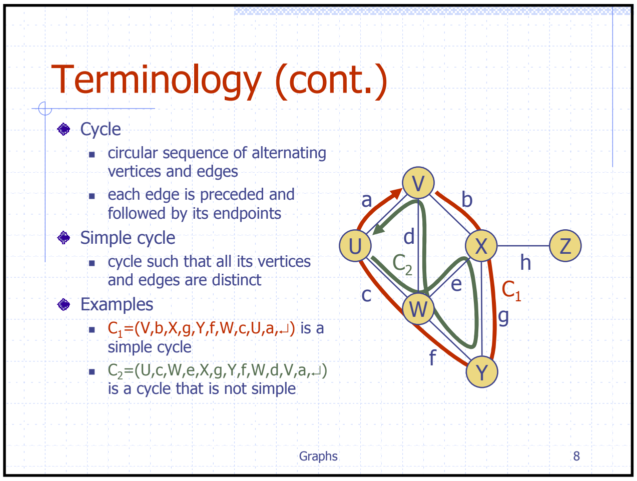
Terminology

- End vertices (or endpoints) of an edge
  - U and V are the endpoints of a
- Edges incident on a vertex
  - a, d, and b are incident on V
- Adjacent vertices
  - U and V are adjacent
- Degree of a vertex
  - X has degree 5
- Parallel edges
  - h and i are parallel edges
- Self-loop
  - j is a self-loop



Terminology (cont.)

- Path
  - sequence of alternating vertices and edges
  - begins with a vertex
  - ends with a vertex
  - each edge is preceded and followed by its endpoints
- Simple path
  - path such that all its vertices and edges are distinct
- Examples
  - P_1=(V,b,X,h,Z) is a simple path
  - P_2=(U,c,W,e,X,g,Y,f,W,d,V) is a path that is not simple



Terminology (cont.)

- Cycle
  - circular sequence of alternating vertices and edges
  - each edge is preceded and followed by its endpoints
- Simple cycle
  - cycle such that all its vertices and edges are distinct
- Examples
  - C_1=(V,b,X,g,Y,f,W,c,U,a,↵) is a simple cycle
  - C_2=(U,c,W,e,X,g,Y,f,W,d,V,a,↵) is a cycle that is not simple

Properties

Property 1

$\sum_v \deg(v) = 2m$

Proof: each endpoint is counted twice

Property 2

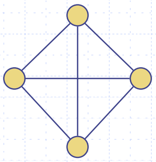
In an undirected graph with no self-loops and no multiple edges

$m \le n\,(n-1)/2$

Proof: each vertex has degree at most $(n-1)$

Notation

n	number of vertices
m	number of edges
$\deg(v)$	degree of vertex v

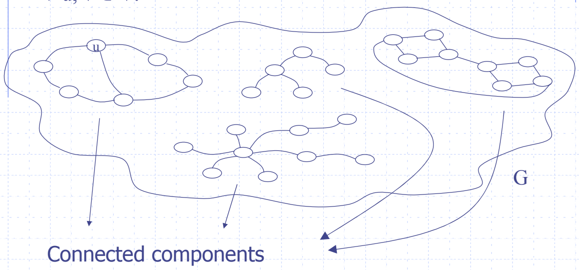


Example

- $n = 4$
- $m = 6$
- $\deg(v) = 3$

Connected Graphs

A (non-directed) graph is connected if there exists a path $\forall$ u, v $\in$ V.



Connected components

Main Methods of the Graph ADT

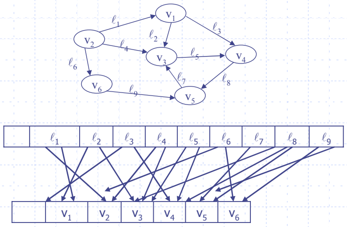
- Vertices and edges
  - are positions
  - store elements
- Accessor methods
  - aVertex()
  - incidentEdges(v)
  - endVertices(e)
  - isDirected(e)
  - origin(e)
  - destination(e)
  - opposite(v, e)
  - areAdjacent(v, w)
- Update methods
  - insertVertex(x)
  - insertEdge(v, w, x)
  - insertDirectedEdge(v, w, x)
  - removeVertex(v)
  - removeEdge(e)
- Generic methods
  - numVertices()
  - numEdges()
  - vertices()
  - edges()

There could be other methods …..

Representations

•Edge List
•Adjacency List
•Adjacency Matrix
•Incidence Matrix

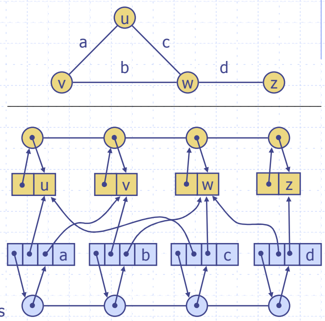
Edge List Structure (example)



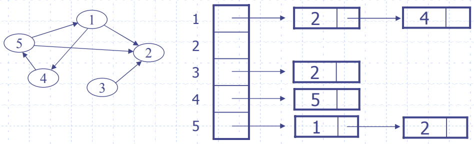
Space: $\boxed{n + m}$

Edge List Structure

- Vertex object
  - element
  - reference to position in vertex sequence
- Edge object
  - element
  - origin vertex object
  - destination vertex object
  - reference to position in edge sequence
- Vertex sequence
  - sequence of vertex objects
- Edge sequence
  - sequence of edge objects

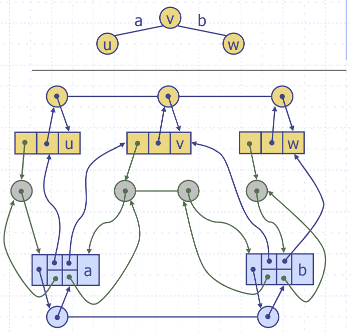


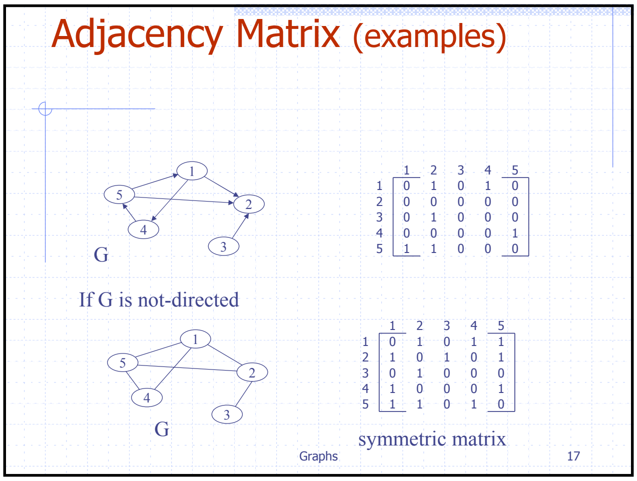
Adjacency List (example)



Adjacency List Structure

- Edge list structure
- Incidence sequence for each vertex
  - sequence of references to edge objects of incident edges
- Augmented edge objects
  - references to associated positions in incidence sequences of end vertices





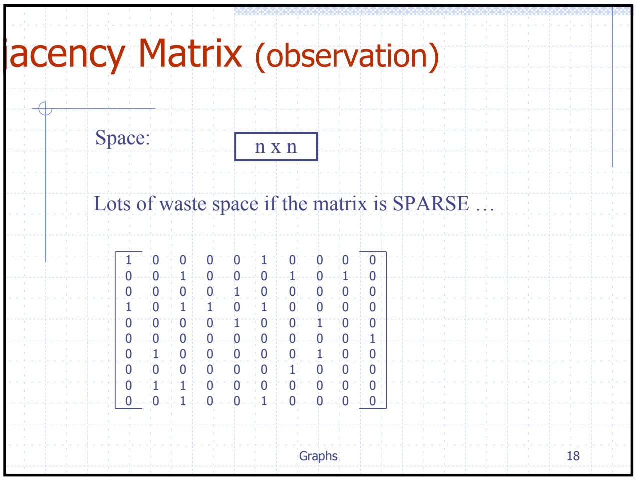
Adjacency Matrix (examples)

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 \\ 5 & 1 & 1 & 0 & 0 & 0 \end{array}$$

If G is not-directed

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 & 0 & 1 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 & 0 & 1 \\ 5 & 1 & 1 & 0 & 1 & 0 \end{array}$$

symmetric matrix

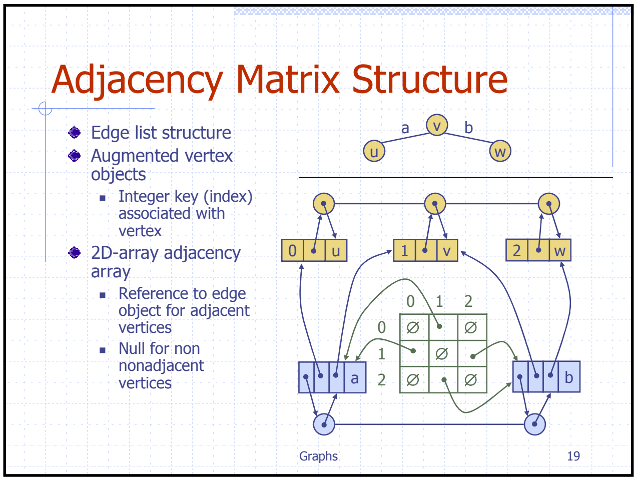


jacency Matrix (observation)

Space: $\boxed{n \times n}$

Lots of waste space if the matrix is SPARSE …

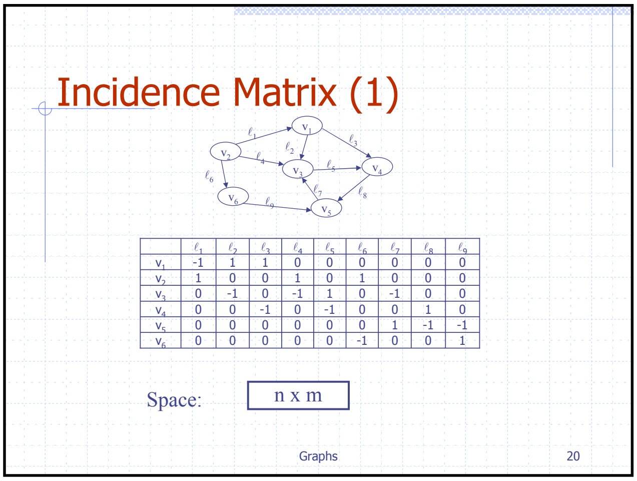
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$



Adjacency Matrix Structure

- Edge list structure
- Augmented vertex objects
  - Integer key (index) associated with vertex
- 2D-array adjacency array
  - Reference to edge object for adjacent vertices
  - Null for non nonadjacent vertices

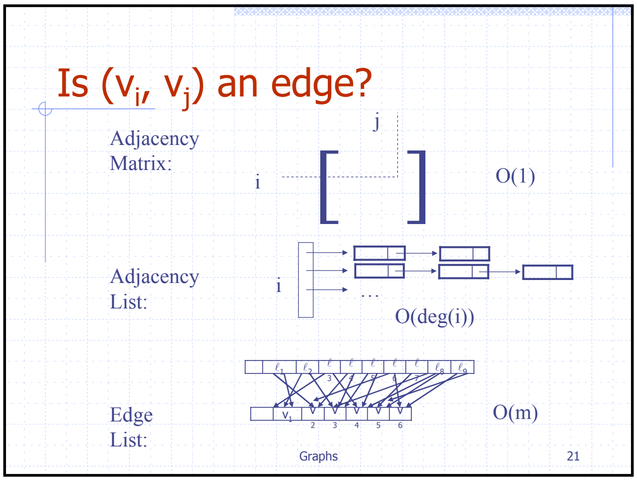
	0	1	2
0	∅	•	∅
1	•	∅	•
2	∅	•	∅



Incidence Matrix (1)

	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5	ℓ_6	ℓ_7	ℓ_8	ℓ_9
v_1	-1	1	1	0	0	0	0	0	0
v_2	1	0	0	1	0	1	0	0	0
v_3	0	-1	0	-1	1	0	-1	0	0
v_4	0	0	-1	0	-1	0	0	1	0
v_5	0	0	0	0	0	0	1	-1	-1
v_6	0	0	0	0	0	-1	0	0	1

Space: $\boxed{n \times m}$



Is (v_i, v_j) an edge?

Adjacency Matrix:

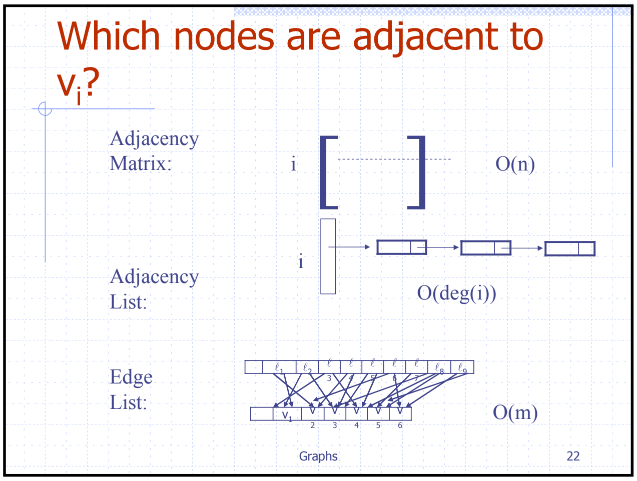
O(1)

Adjacency List:

O(deg(i))

Edge List:

O(m)



Which nodes are adjacent to v_i?

Adjacency Matrix:

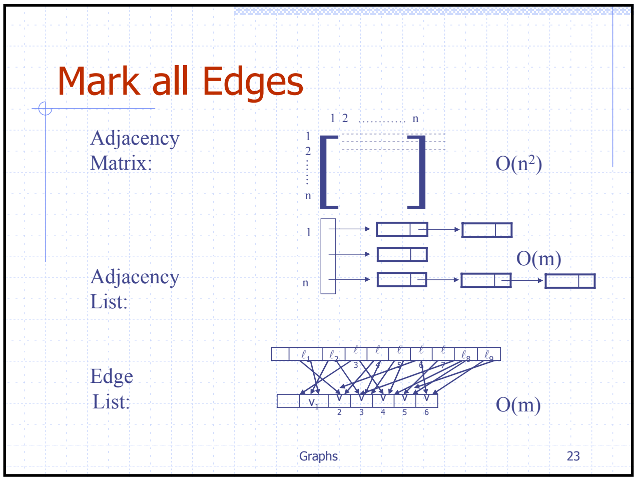
O(n)

Adjacency List:

O(deg(i))

Edge List:

O(m)



Mark all Edges

Adjacency Matrix:

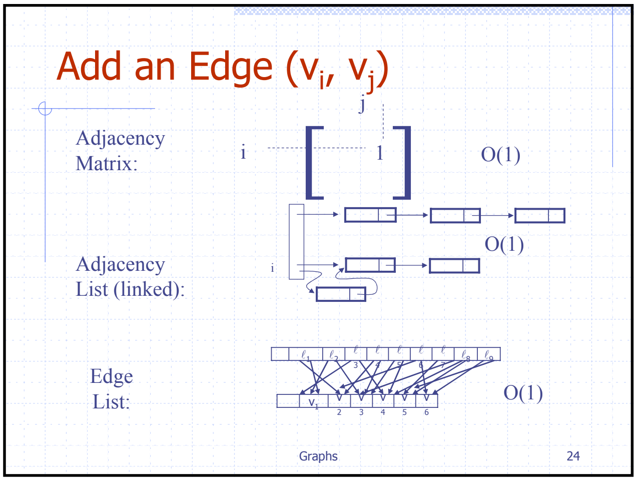
$O(n^2)$

Adjacency List:

O(m)

Edge List:

O(m)



Add an Edge (v_i, v_j)

Adjacency Matrix:

O(1)

Adjacency List (linked):

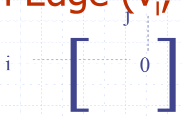
O(1)

Edge List:

O(1)

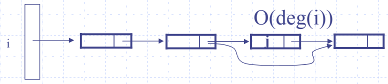
Remove an Edge (v_i, v_j)

Adjacency Matrix:



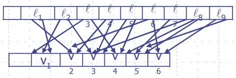
O(1)

Adjacency List :



O(deg(i))

Edge List:



O(1)

	Adjacency Matrix	Adjacency List
Is (v_i, v_j) an edge?	O(1)	O(deg(i))
Which nodes are adjacent to v_i?	O(n)	O(deg(i))
Mark all edges	O(n^2)	O(m)
Add edge (v_i, v_j)	O(1)	O(1)
Remove edge (v_i, v_j)	O(1)	O(deg(i))

O(deg(i)) = OUT-degree of node vi

G is directed

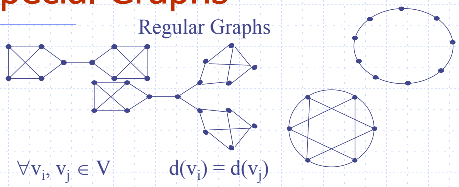
What are the predecessors of v_i?

Performance

- n vertices - m edges - no parallel edges - no self-loops	Edge List	Adjacency List	Adjacency Matrix
Space	$n + m$	$n + m$	n^2
incidentEdges(v)	m	deg(v)	n
areAdjacent (v, w)	m	min(deg(v), deg(w))	1
insertVertex(x)	1	1	n^2
insertEdge(v, w, x)	1	1	1
removeVertex(v)	m	deg(v)	n^2
removeEdge(e)	1	1	1

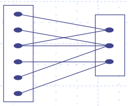
Special Graphs

Regular Graphs

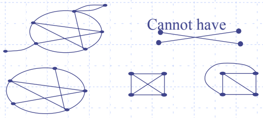


$\forall v_i, v_j \in V$ $\quad d(v_i) = d(v_j)$

Bipartite Graphs



Planar Graphs



$$\boxed{\begin{array}{c} n-1 \le m \le \frac{n(n-1)}{2} \\ 1 \le deg(i) \le n-1 \end{array}}$$

connected,
non-directed

degree

n = | V | m = | E |

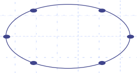
$$\boxed{\begin{array}{c} n-1 \le m \le n(n-1) \\ 1 \le deg(i) \le n-1 \end{array}}$$

connected,
directed

OUT-degree

Some Regular Graphs

— Ring —

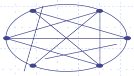


m = n

$d_i = 2 \quad \forall i$

Tree
m = O(n)

— Complete Graph —

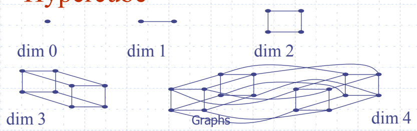


$m = \frac{n(n-1)}{2}$

$m = O(n^2)$

$d_i = n - 1 \quad \forall i$

— Hypercube —



Hypercube

	n	m
h_0	1	0
h_1	2	1
h_2	4	1x2+2 = 4
h_3	8	4x2+4 = 12
h_4	16	12x2+8 = 32

h_i:

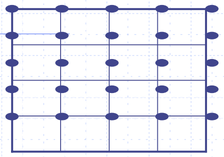
$$\left\{\begin{array}{l} n_0 = 1 \\ n_i = 2\, n_{i-1} \end{array}\right. \begin{array}{l} \longrightarrow n_i = 2^i \\ \longrightarrow m_i = i \bullet 2^{i-1} \end{array}$$

$m = \frac{n \log n}{2}$

d = log n

m = O(n log n)

— Grid —

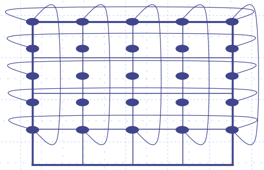


Not regular

m = O(n)

deg(i) = 4 v_i = internal
deg(i) = 3 v_i = border
deg(i) = 2 v_i = corner

— Torus —



m = O(n)

$deg(i) = 4 \quad \forall i$