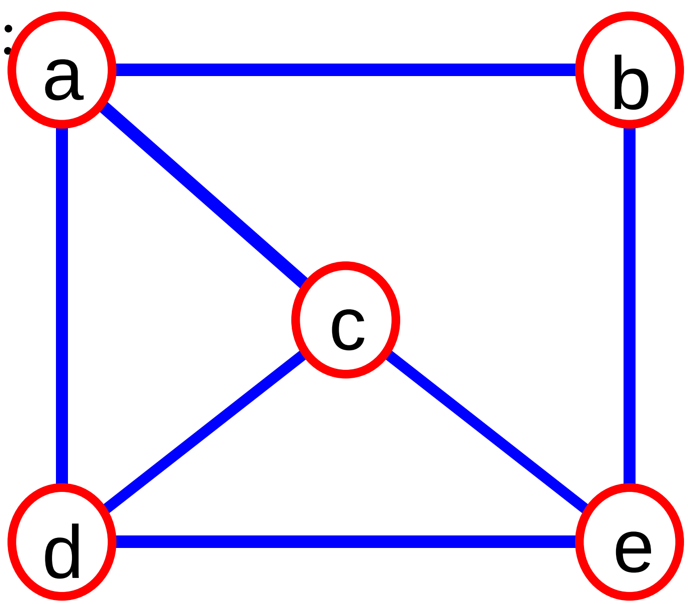
Graphs

What is a Graph?

- A graph G = (V,E) is composed of:

  V: set of vertices

  E: set of edges connecting the vertices in V
- An edge e = (u,v) is a pair of vertices
- Example:

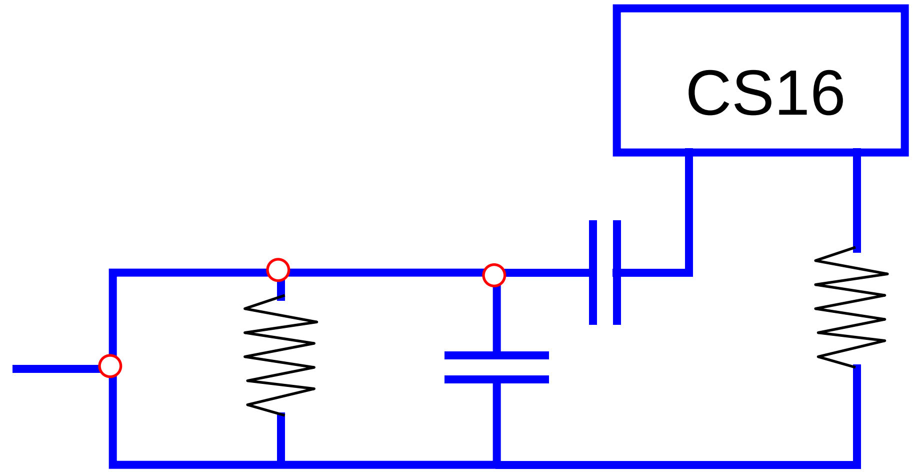


V= {a,b,c,d,e}

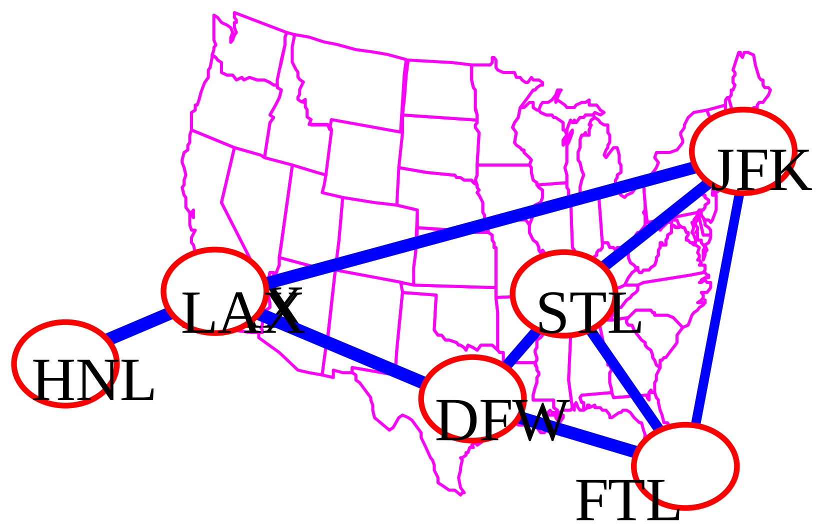
E= {(a,b),(a,c),(a,d),
(b,e),(c,d),(c,e),
(d,e)}

Applications

- electronic circuits



- networks (roads, flights, communications)



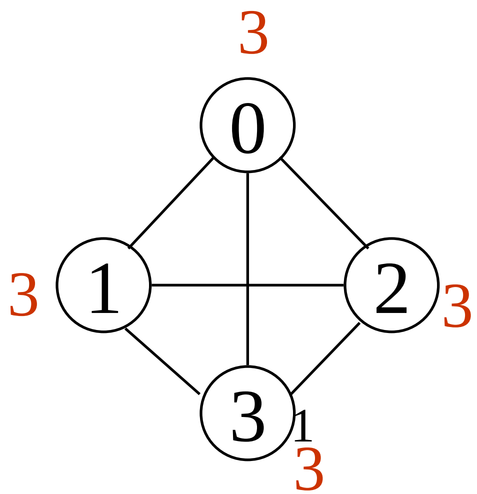
Terminology
Adjacent and Incident

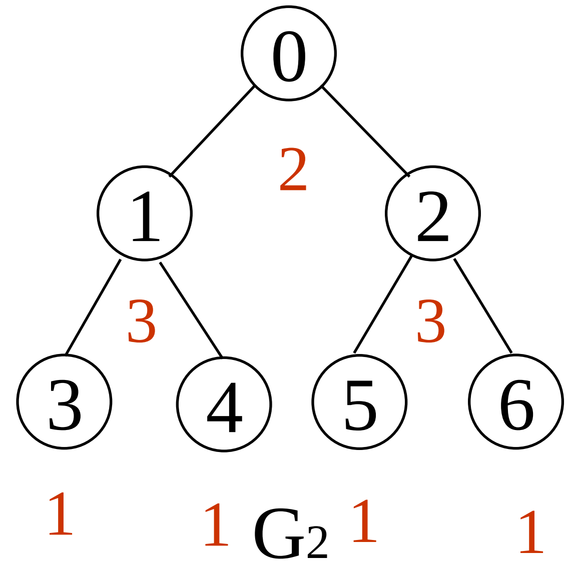
- If (v_0, v_1) is an edge in an undirected graph,
  - v_0 and v_1 are **adjacent (w.r.t.vertices)**
  - The edge (v_0, v_1) is **incident** on vertices v_0 and v_1 (w.r.t.edges)
- If $<v_0, v_1>$ is an edge in a directed graph
  - v_0 is adjacent to v_1, and v_1 is adjacent from v_0
  - The edge $<v_0, v_1>$ is incident on v_0 and v_1

Degree of a Vertex

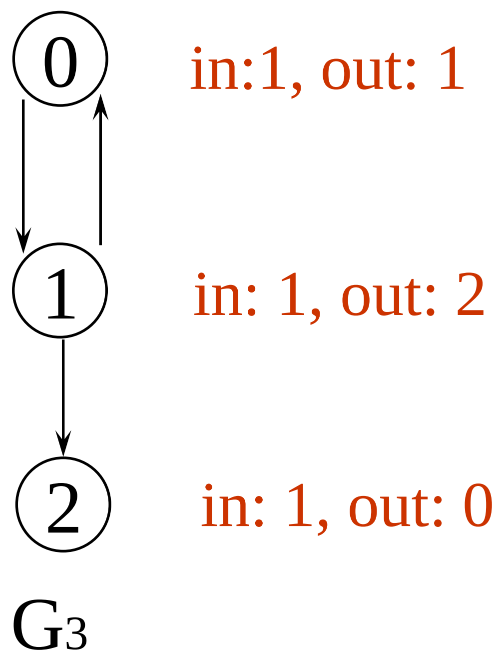
- The degree of a vertex is the number of edges incident to that vertex
- For directed graph,
  - the in-degree of a vertex *v* is the number of edges that have *v* as the head
  - the out-degree of a vertex *v* is the number of edges that have *v* as the tail

Examples



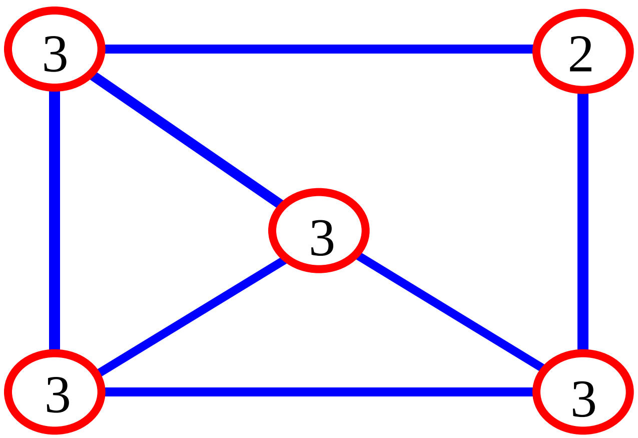


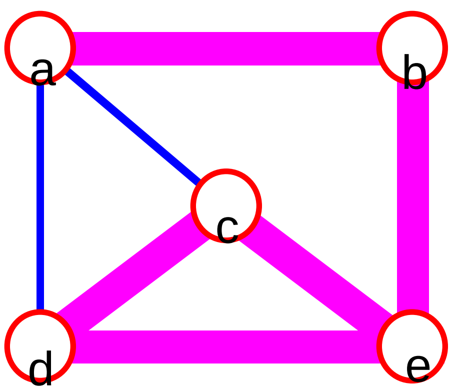
directed graph
in-degree
out-degree



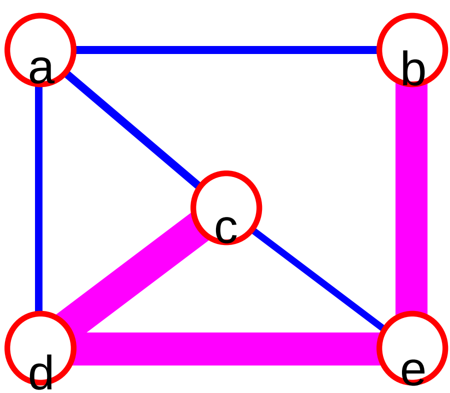
Path

- path: sequence of vertices $v_1, v_2, \ldots v_k$ such that consecutive vertices v_i and v_{i+1} are adjacent.





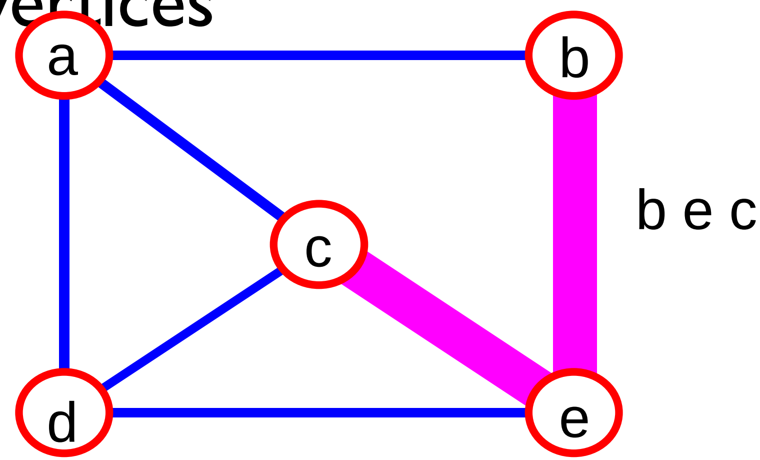
a b e d c



b e d c

More Terminology

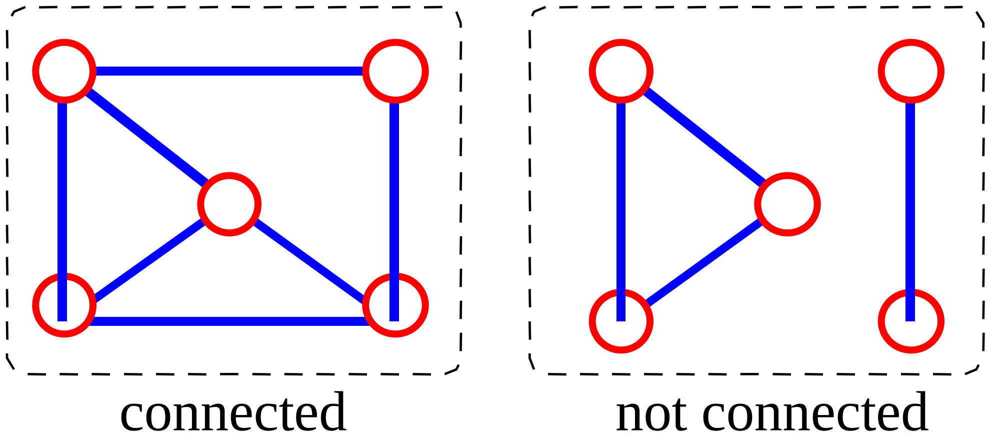
- simple path: no repeated vertices



- cycle: simple path, except that the last vertex is the same as the first vertex

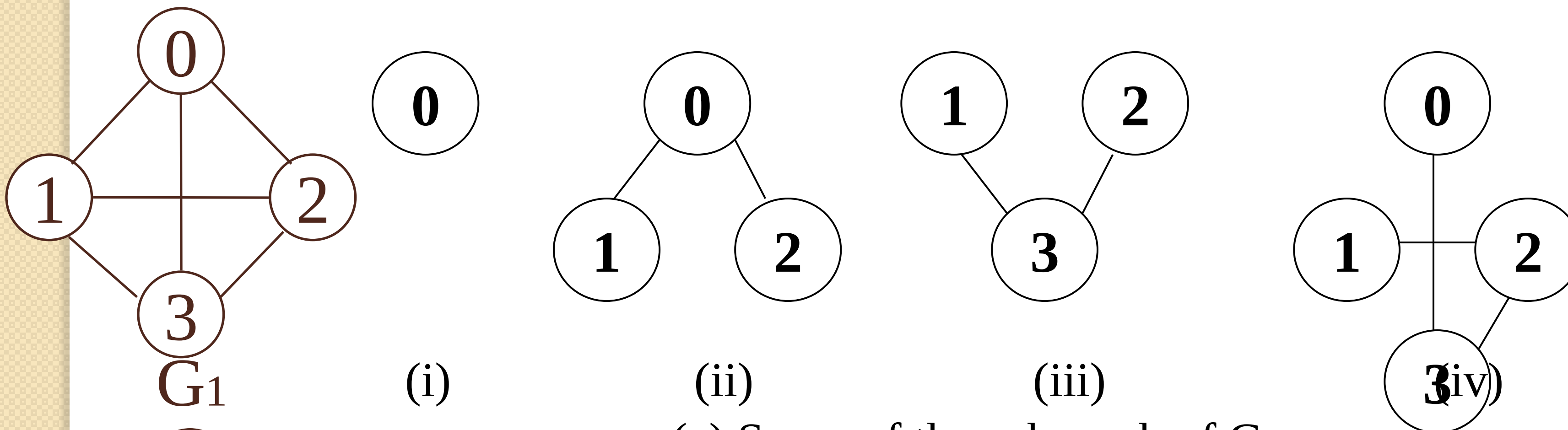
Even More Terminology

- connected graph: any two vertices are connected by some path

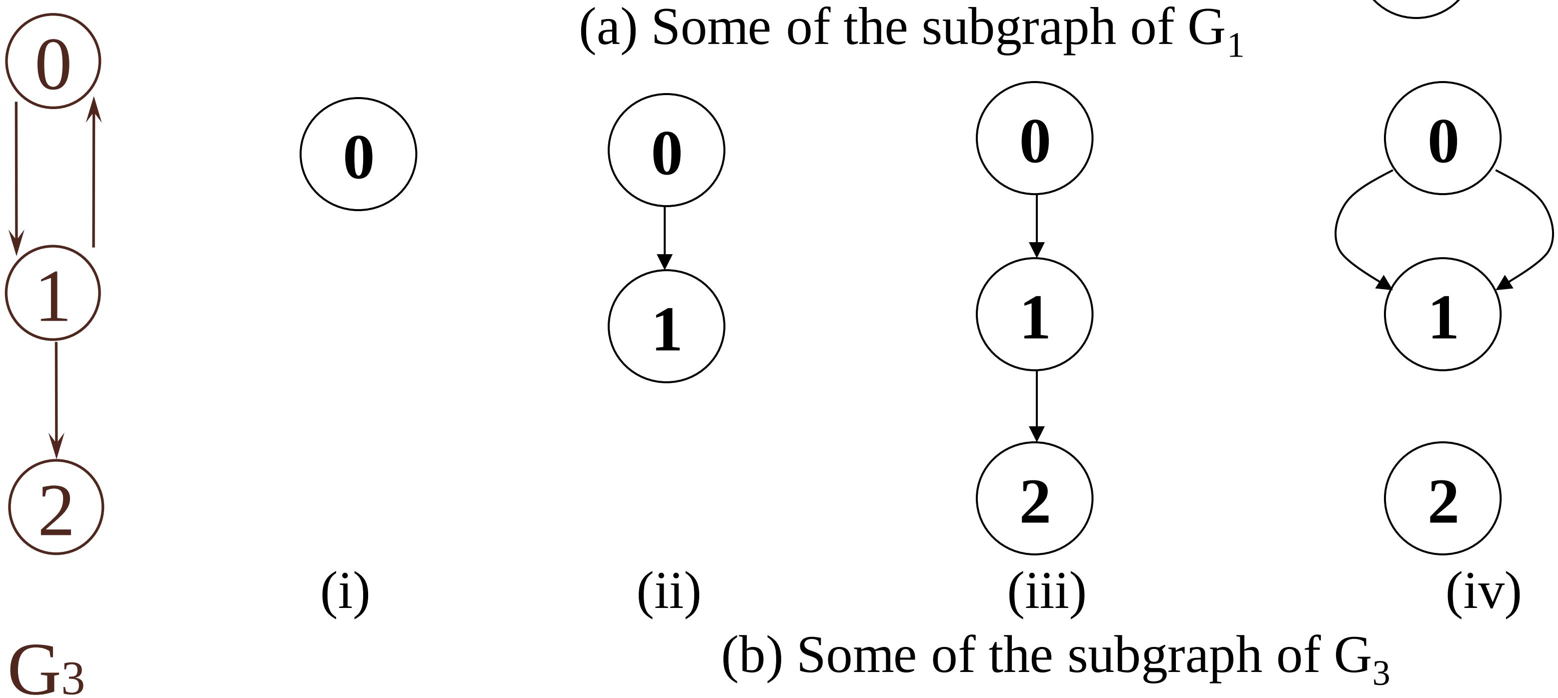


- subgraph: subset of vertices and edges forming a graph

Subgraphs Examples



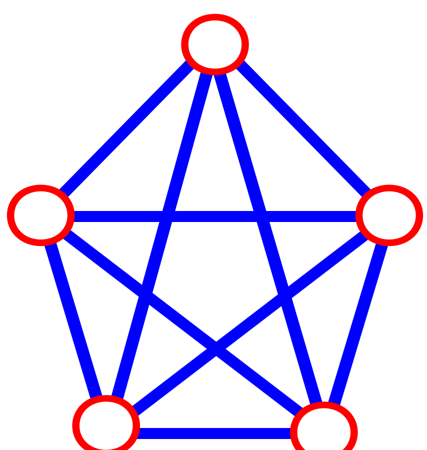
(a) Some of the subgraph of G_1



(b) Some of the subgraph of G_3

Connectivity

- Let **n** = #vertices, and **m** = #edges
- **A complete graph**: one in which all pairs of vertices are adjacent
- *How many total edges in a complete graph?*
  - Each of the n vertices is incident to **n-1** edges, however, we would have counted each edge twice! Therefore, intuitively, m = **n**(**n** -1)/2.
- Therefore, if a graph is not complete, m < **n**(**n** -1)/2



$\mathbf{n} = 5$

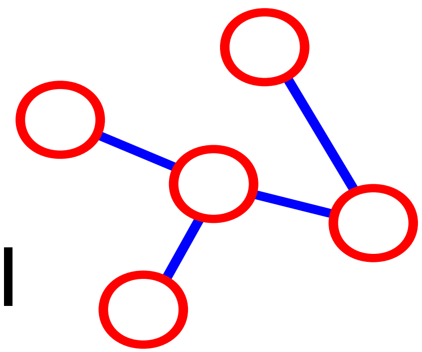
$\mathbf{m} = (5 * 4)/2 = 10$

More Connectivity

n = #vertices

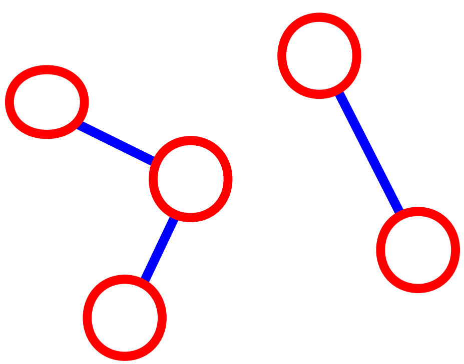
m = #edges

- For a tree **m** = **n** - 1



n = 5
m = 4

If **m** < **n** - 1, G is not connected



n = 5
m = 3

Directed vs. Undirected Graph

- An **undirected graph** is one in which the pair of vertices in a edge is unordered, $(v_0, v_1) = (v_1, v_0)$
- A **directed graph** is one in which each edge is a directed pair of vertices, $<v_0, v_1> \; != \; <v_1, v_0>$ tail $\longrightarrow$ head

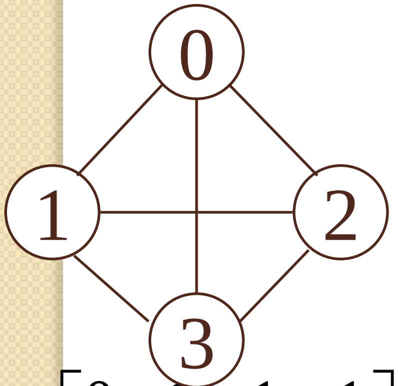
Graph Representations

- Adjacency Matrix
- Adjacency Lists

Adjacency Matrix

- Let G=(V,E) be a graph with n vertices.
- The adjacency matrix of G is a two-dimensional n by n array, say adj_mat
- If the edge (vi, vj) is in E(G), adj_mat[i][j]=1
- If there is no such edge in E(G), adj_mat[i][j]=0
- The adjacency matrix for an undirected graph is symmetric; the adjacency matrix for a digraph need not be symmetric

Examples for Adjacency Matrix

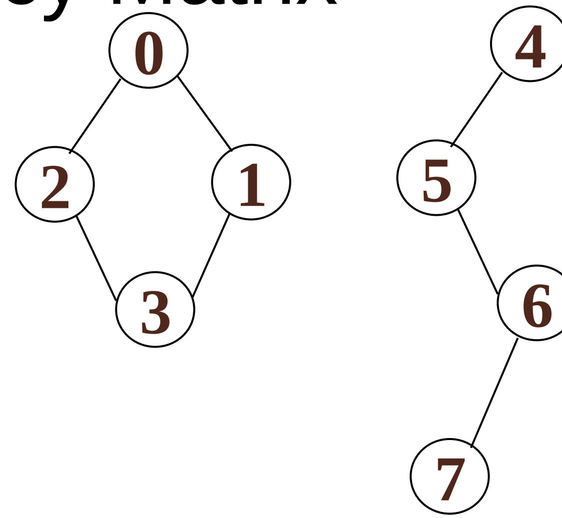


$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

G_1

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

G_2



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

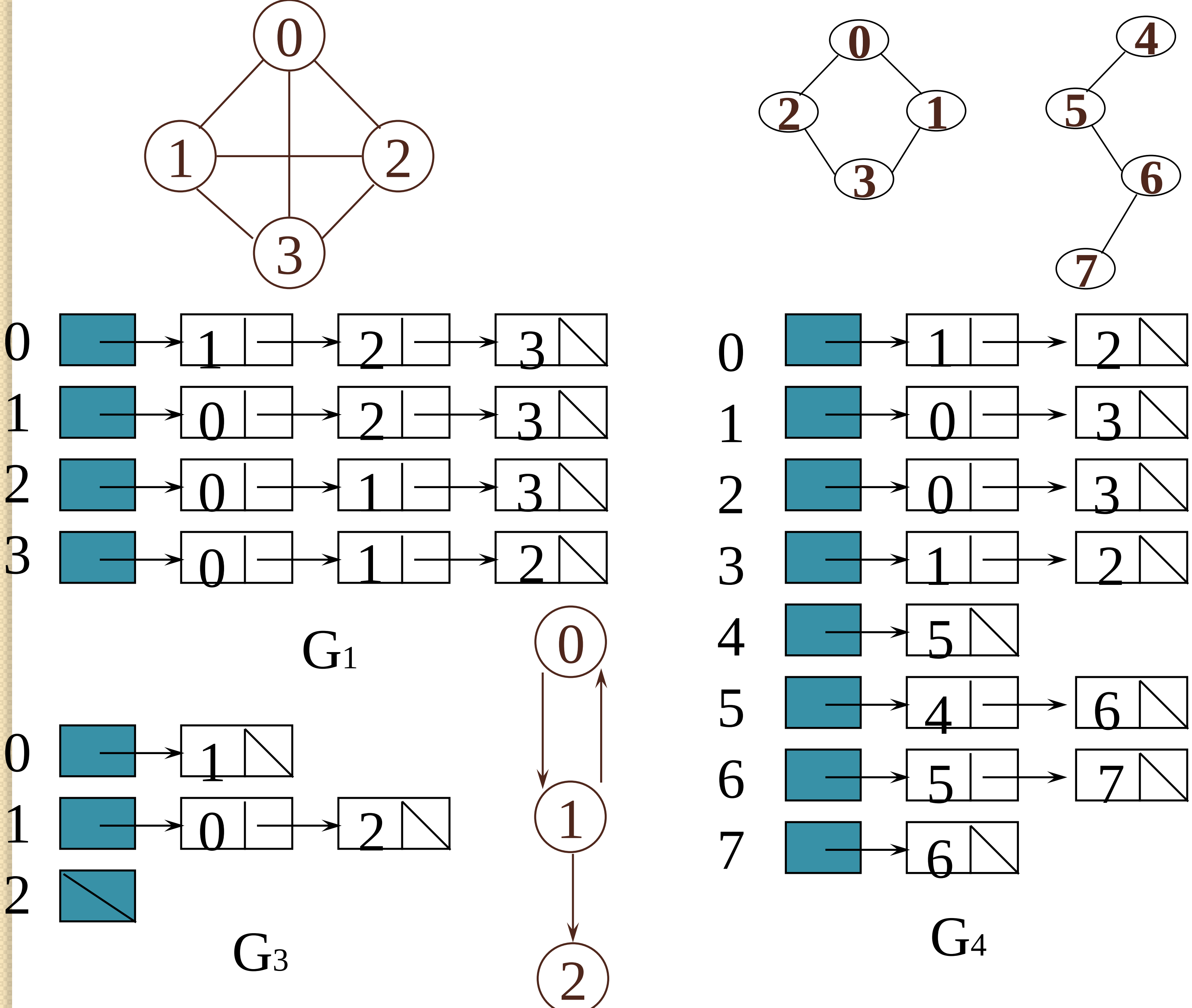
symmetric

G_4

Adjacency Lists (data structures

Each row in adjacency matrix is represented as an adjacency list.

```
#define MAX_VERTICES 50
typedef struct node *node_pointer;
typedef struct node {
    int vertex;
    struct node *link;
};
node_pointer graph[MAX_VERTICES];
```

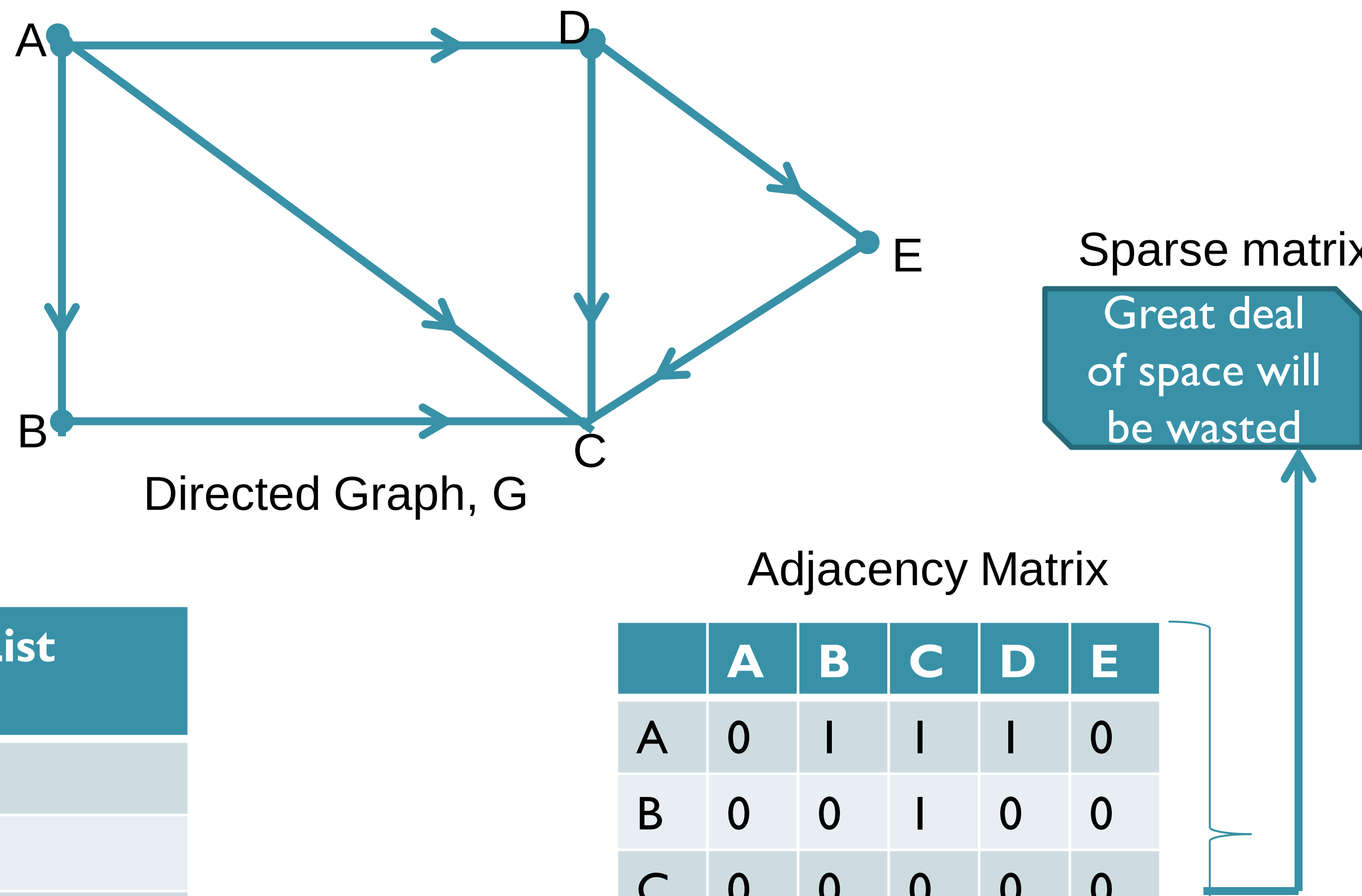


An undirected graph with n vertices and e edges ==> n head nodes and 2e list nodes

Linked Representation of Graph

S. Graceline Jasmine, SCSE, VIT

Drawbacks of sequential representation of G in memory



Adjacency List

Node	Adjacency List
A	B, C, D
B	C
C	
D	C, E
E	C

Adjacency Matrix

	A	B	C	D	E
A	0	1	1	1	0
B	0	0	1	0	0
C	0	0	0	0	0
D	0	0	1	0	1
E	0	0	1	0	0

Linked Representation

NODE LIST

NODE	NEXT	ADJ	

Key Value of the node

Pointer to the next node in the list Node

Pointer to the first element in the adjacency list of the node

Other information, like INDEGREE, OUTDEGREE and the STATUS of the node during execution of an algorithm

Linked Representation

EDGE LIST

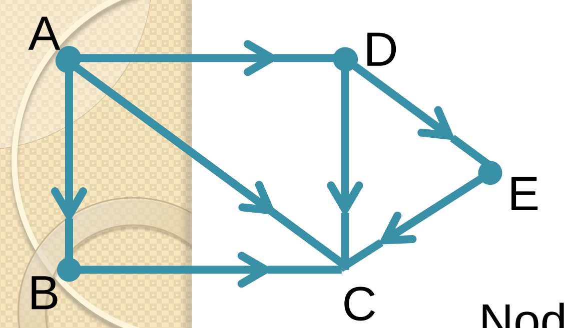
DEST	LINK	

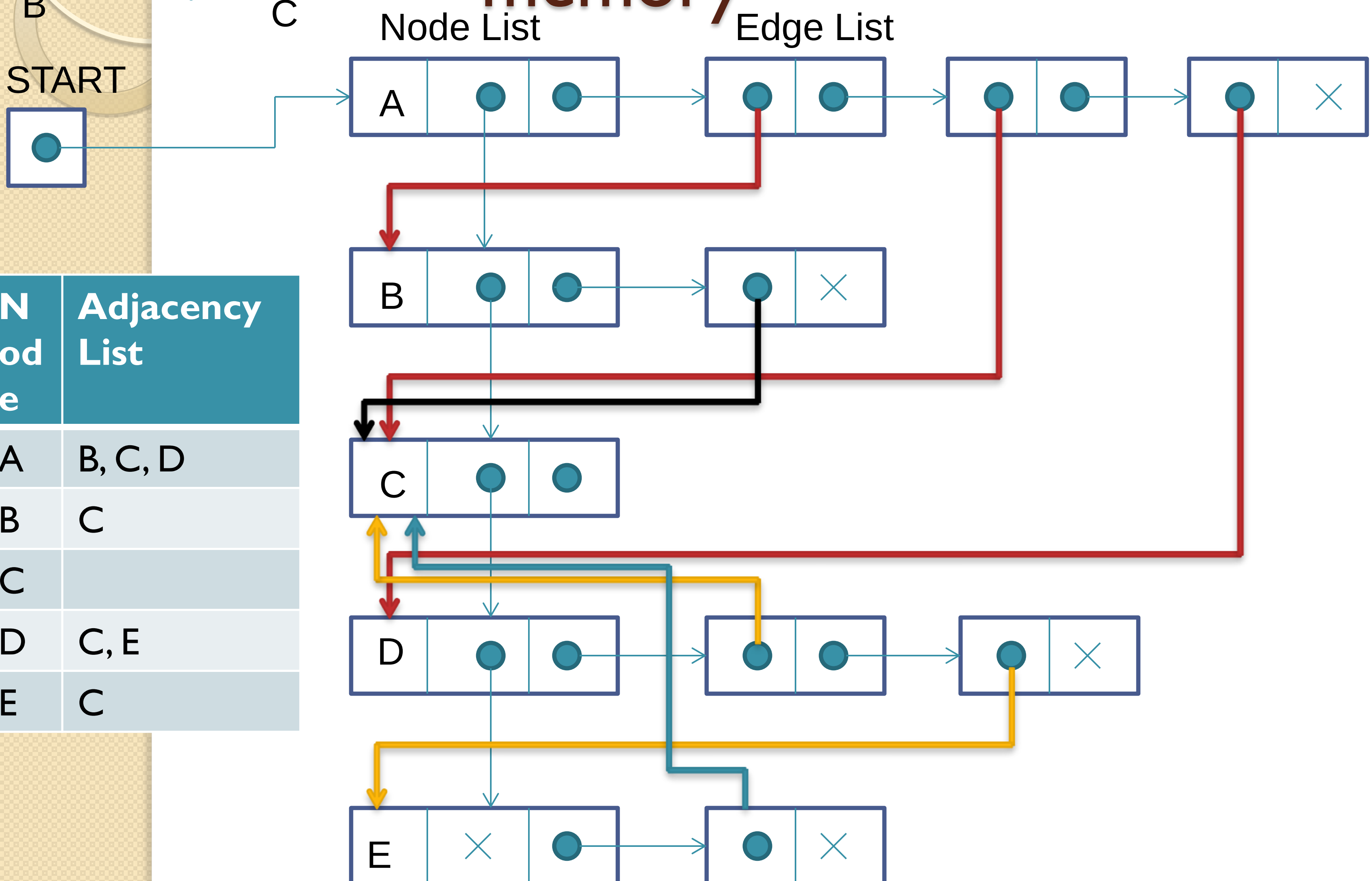
Point to the location in the NODE List of the *destination* node of the edge

Pointer to the next node in the same adjacency list

Other information, like INDEGREE, OUTDEGREE and the STATUS of the node during execution of an algorithm

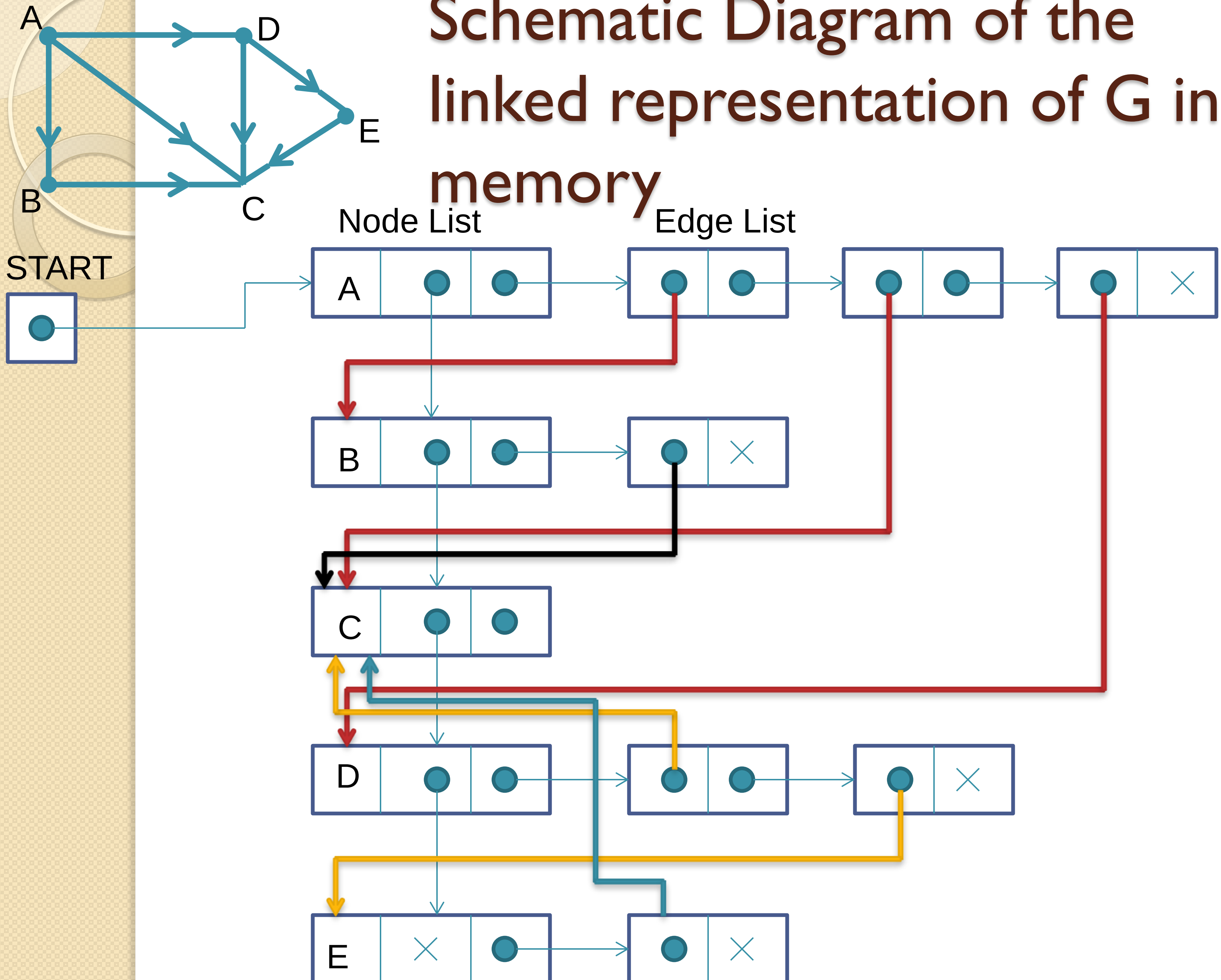
Schematic Diagram of the linked representation of G in memory



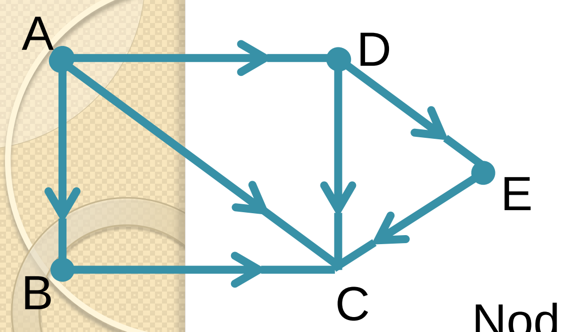


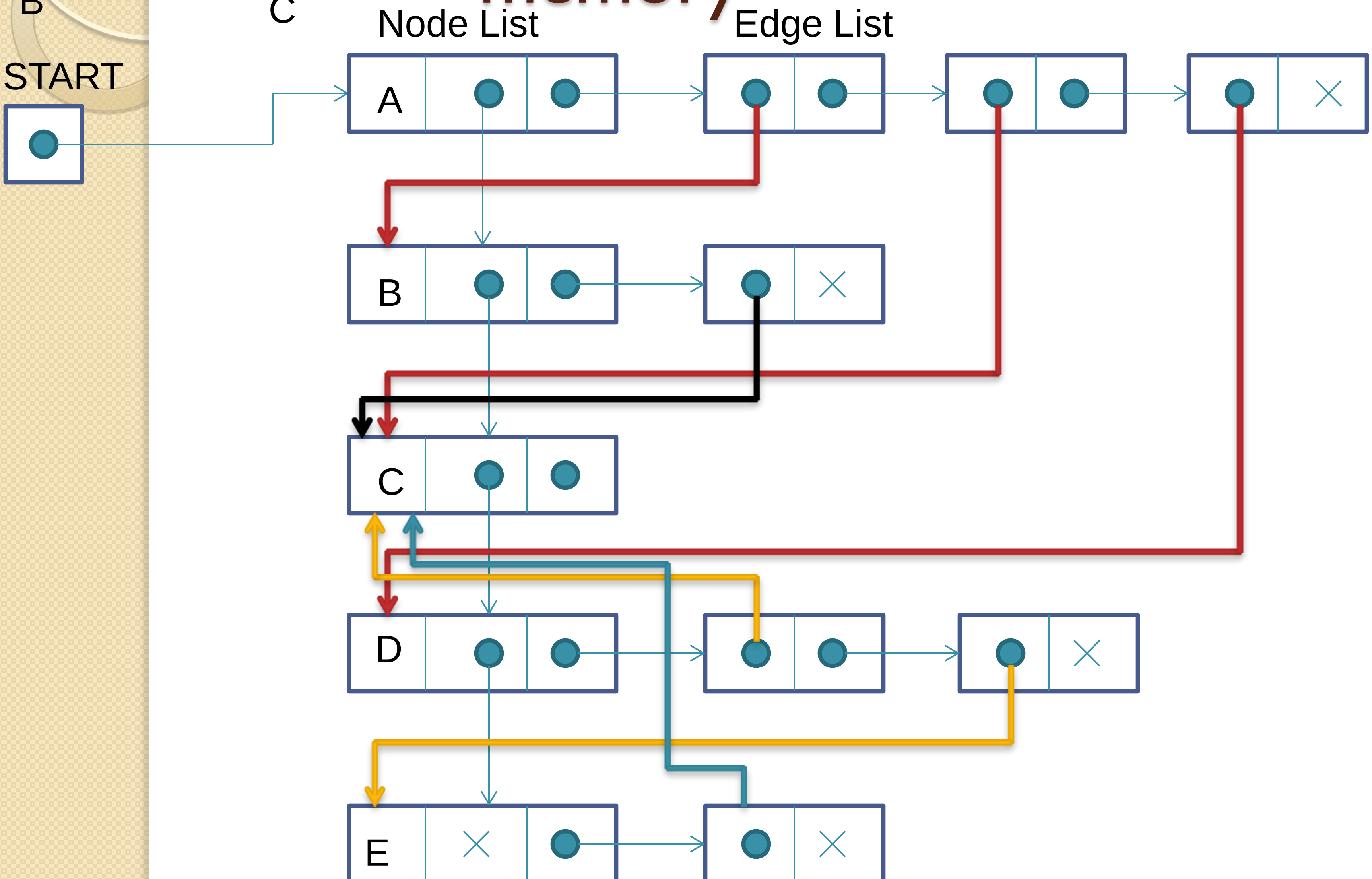
Node	Adjacency List
A	B, C, D
B	C
C	
D	C, E
E	C

Schematic Diagram of the linked representation of G in memory

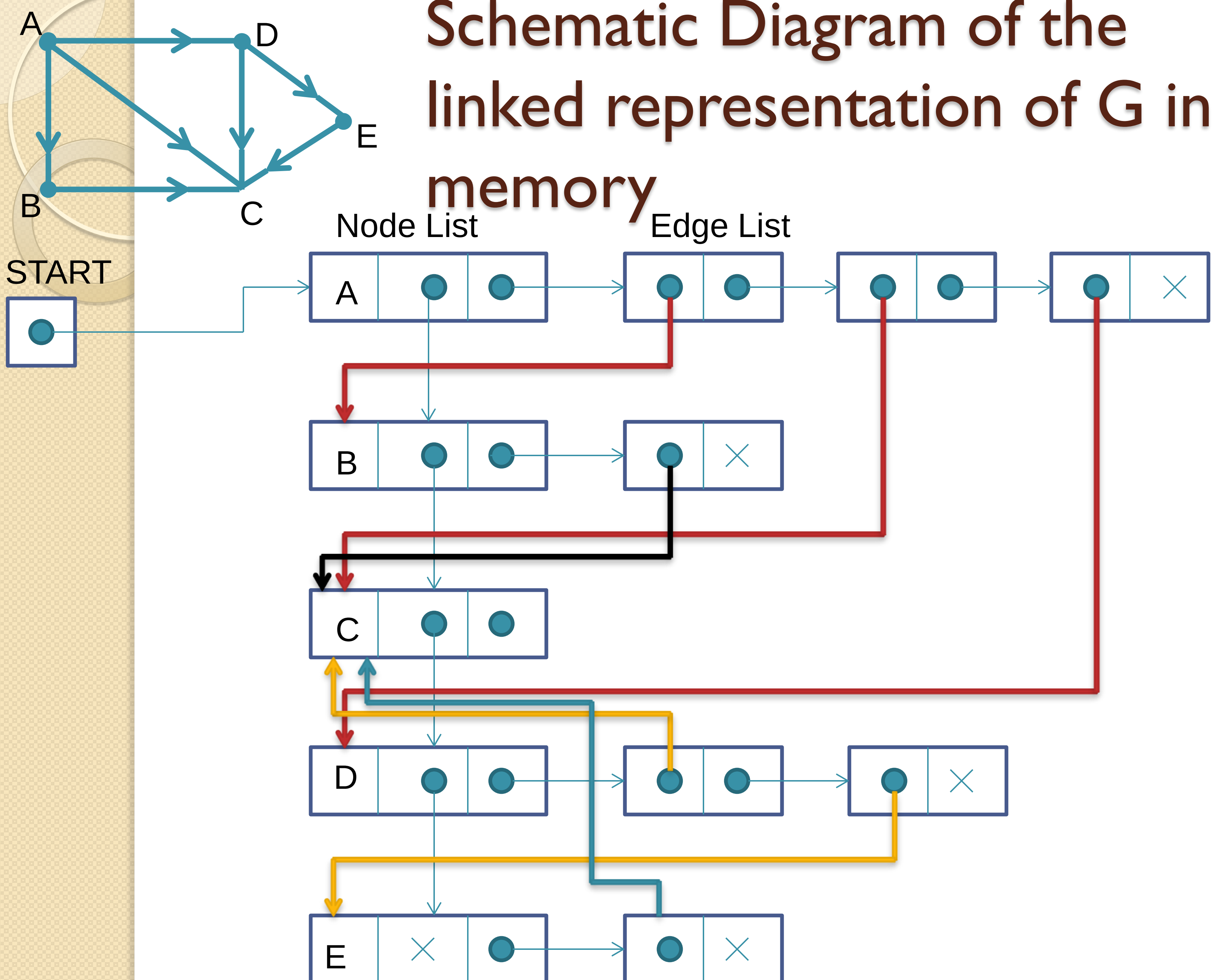


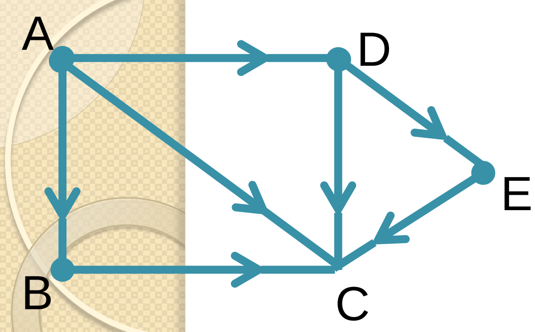
Schematic Diagram of the linked representation of G in memory





Schematic Diagram of the linked representation of G in memory





Graph G in memory

START

4

	Node	NEXT	ADJ
1		3	
2	C	9	0
3		8	
4	A	7	3
5		1	
6	E	0	11
7	B	2	6
8		10	
9	D	6	1
10		0	

	DEST	LINK
1	2 (c)	7
2		5
3	7 (B)	10
4	9 (D)	0
5		8
6	2 (C)	0
7	6 (E)	0
8		9
9		12
10	2 (C)	4
11	2 (c)	0
12		0