TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Discrete Mathematics

Nguyễn Khánh Phương

Department of Computer Science
School of Information and Communication Technology
E-mail: phuongnk@soict.hust.edu.vn

PART 1
COMBINATORIAL THEORY
(Lý thuyết tổ hợp)

PART 2
GRAPH THEORY
(Lý thuyết đồ thị)

Content of Part 2

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chapter 2

Graph representation

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Graph Representation

1. Incidence matrix
2. Adjacency matrix
3. Weight matrix
4. Adjacency list

Graph Representation

1. **Incidence matrix**
2. Adjacency matrix
3. Weight matrix
4. Adjacency list

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

1.Incidence Matrix

G = (V, E) is an unditected graph:

- $V = \{v_1, v_2, v_3, \ldots, v_n\}$
- $E = \{e_1, e_2, \ldots, e_m\}$

Then the incidence matrix with respect to this ordering of V and E is the n x m matrix $M = [m_{ij}]$, where

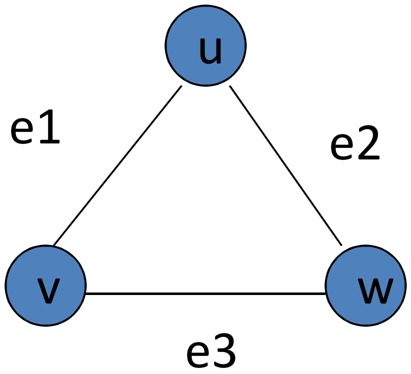
$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i \\ 0 & \text{otherwise} \end{cases}$$

Can also be used to represent :

- **Multiple edges:** by using columns with identical entries, since these edges are incident with the same pair of vertices
- **Loops:** by using a column with exactly one entry equal to 1, corresponding to the vertex that is incident with the loop

1.Incidence Matrix

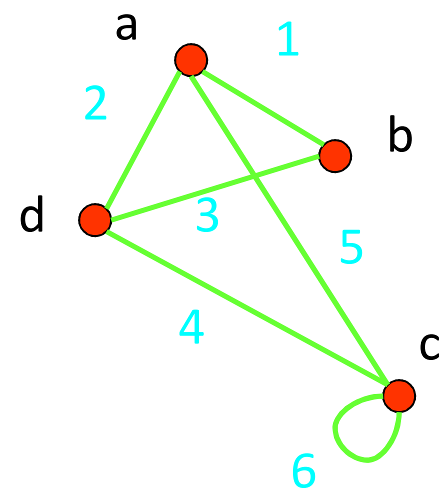
Example: G = (V, E)



	e_1	e_2	e_3
v	1	0	1
u	1	1	0
w	0	1	1

1.Incidence Matrix

Example: What is the incidence matrix M for the following graph G based on the order of vertices a, b, c, d and edges 1, 2, 3, 4, 5, 6?



Solution:

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

Note: Incidence matrices of undirected graphs contain two 1s per column for edges connecting two vertices and one 1 per column for loops.

Graph Representation

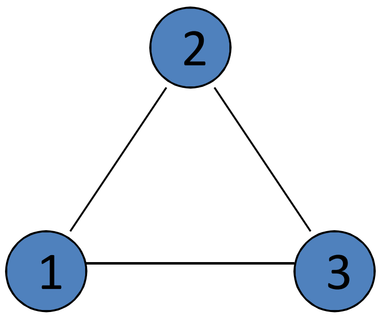
1. Incidence matrix
2. **Adjacency matrix**
3. Weight matrix
4. Adjacency list

2. Adjacency Matrix

The Adjacenct Matrix (NxN) $A = [a_{ij}]$ where $|V| = N$

For undirected graph

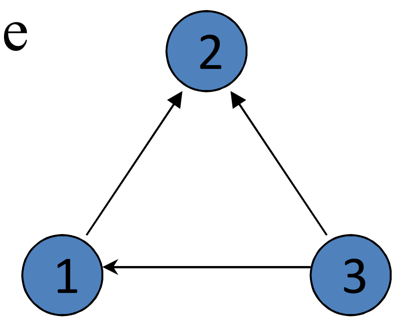
$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of G} \\ 0 & \text{otherwise} \end{cases}$$



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

For directed graph

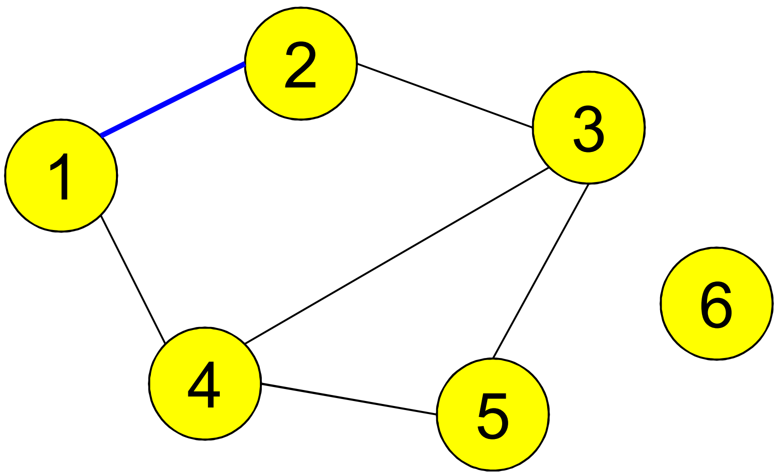
$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \text{ is an edge of G} \\ 0 & \text{otherwise} \end{cases}$$



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

This makes it easier to find subgraphs, and to reverse graphs if needed.

2. Adjacency Matrix

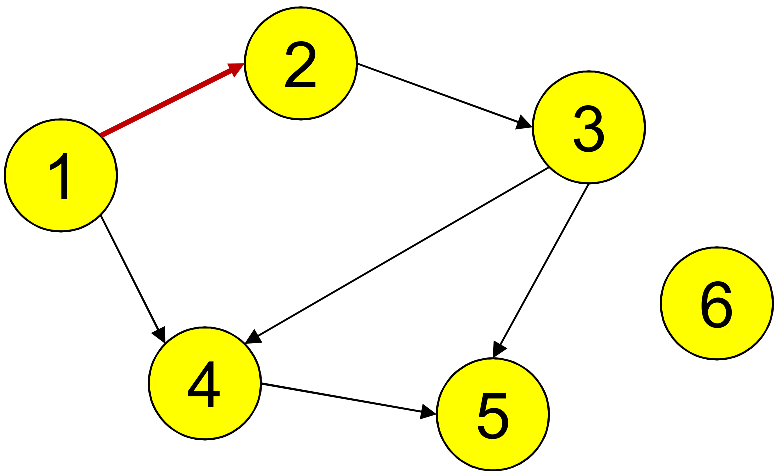


$$
\begin{array}{c|c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

$$
A[u,v] = \begin{cases} 1 \text{ if } \{u,v\} \in E \\ 0 \text{ otherwise} \end{cases}
$$

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Representation- Adjacency Matrix



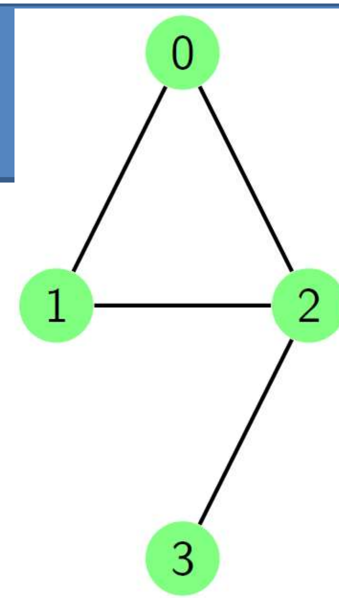
$$\begin{array}{c c} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$A[u,v] = \begin{cases} 1 \text{ if } (u,v) \in E \\ 0 \text{ otherwise} \end{cases}$$

Representation- Adjacency List

```
# Print the graph
def print_graph():
  global graph
  global vertices_no
  for i in range(vertices_no):
    for j in range(vertices_no):
      if graph[i][j] != 0:
        print(vertices[i], " -> ", vertices[j])
```

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

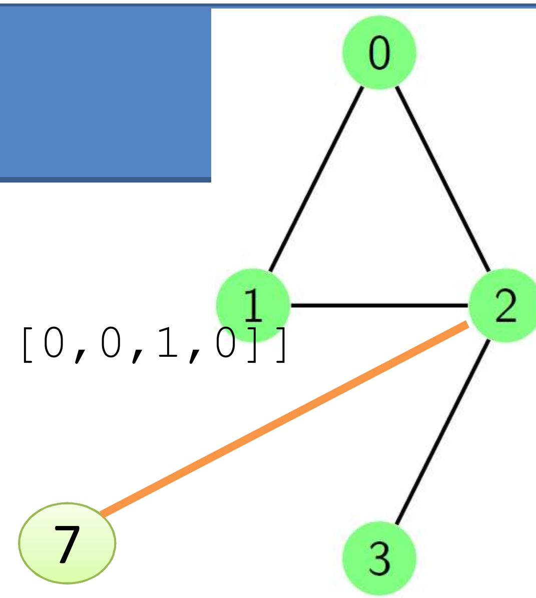



```
# Driver code
# stores the vertices in the graph
vertices = [0,1,2,3]
# stores the number of vertices in the graph
vertices_no = 4
graph = [[0, 1, 1, 0],[1, 0, 1, 0],[1, 1,
0,1],[0,0,1,0]]
print("List of edges: ")
print_graph()
```

```
List of edges:
0  ->  1
0  ->  2
1  ->  0
1  ->  2
2  ->  0
2  ->  1
2  ->  3
3  ->  2
```

N
B

Representation- Adjacency List




```
vertices = [0,1,2,3]
vertices_no = 4
graph = [[0, 1, 1, 0],[1, 0, 1, 0],[1, 1, 0,1],[0,0,1,0]]
v = 7
if v in vertices:
  print("Vertex ", v, " already exists")
else:
  vertices_no = vertices_no + 1
  vertices.append(v)
  for vertex in graph:
    vertex.append(0)
  temp = []
  for i in range(vertices_no):
     temp.append(0)
  graph.append(temp)
  index1 = vertices.index(v)
  index2 = vertices.index(2)
  graph[index1][index2] = 1
  graph[index2][index1] = 1
print("List of edges: ")
print_graph()
```

```
List of edges:
0  ->  1
0  ->  2
1  ->  0
1  ->  2
2  ->  0
2  ->  1
2  ->  3
2  ->  7
3  ->  2
7  ->  2
```

0	1	1	0	0
1	0	1	0	0
1	1	0	1	0
0	0	1	0	0
0	0	0	0	0

Representation- Adjacency Matrix

- The adjacency matrix of simple graphs are symmetric ($a_{ij} = a_{ji}$) (why?)
- When there are relatively few edges in the graph the adjacency matrix is a **sparse matrix**
- Directed Multigraphs can be represented by using a_{ij} = number of edges from v_i to v_j

Analyze the cost

- Memory Space
  - $|V|^2$ bits
- Time to answer the query
  - Two vertices i and j *are adjacent*? $O(1)$
  - Add or delete one edge $O(1)$
  - Add one vertice increase the size of matrix
  - Enumerate the adjacent vertices of u $O(|V|)$ (even when u is an isolated vertice).

Graph Representation

1. Incidence matrix
2. Adjacency matrix
3. **Weight matrix**
4. Adjacency list

3. Weight matrix

- **Weighted** graphs have values associated with edges.
- In the case weighted graphs, instead of adjacency matrix, we use weight matrix to represent the graph

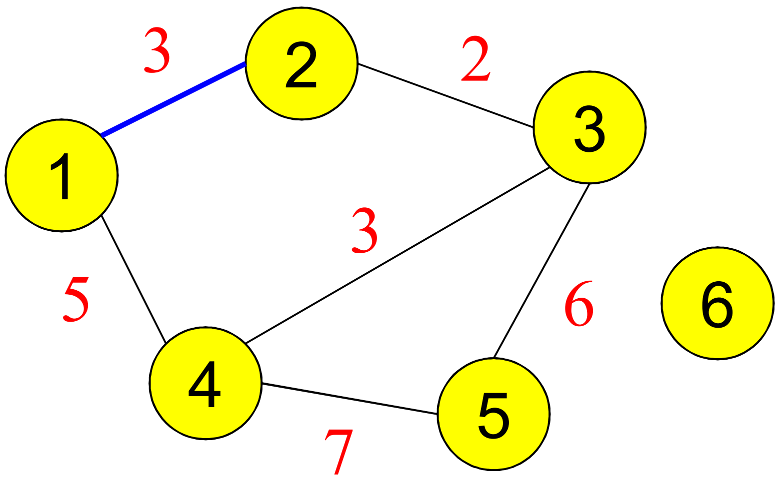
$$C = c[i, j], \ i, j = 1, 2, \ldots, n,$$

where

$$c[i,j] = \begin{cases} c(i,j), & \text{if } (i,j) \in E \\ \theta, & \text{if } (i,j) \notin E, \end{cases}$$

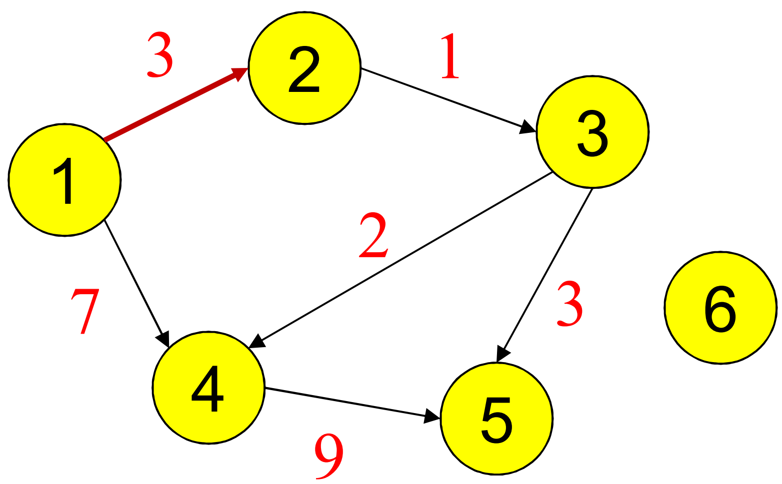
- θ: special value to identify (i, j) is not an edge; depends on the case, the value of θ could be: 0, $+\infty$, $-\infty$.

Weight matrix of undirected graph



$$
\begin{array}{c}
 \\
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{pmatrix}
0 & 3 & 0 & 5 & 0 & 0 \\
3 & 0 & 2 & 0 & 0 & 0 \\
0 & 2 & 0 & 3 & 6 & 0 \\
5 & 0 & 3 & 0 & 7 & 0 \\
0 & 0 & 6 & 7 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

Weight matrix of directed graph



$$
\begin{array}{c|c}
 & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} &
\begin{pmatrix}
0 & 3 & 0 & 7 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 3 & 0 \\
0 & 0 & 0 & 0 & 9 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\end{array}
$$

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Graph Representation

1. Incidence matrix
2. Adjacency matrix
3. Weight matrix
4. **Adjacency list**

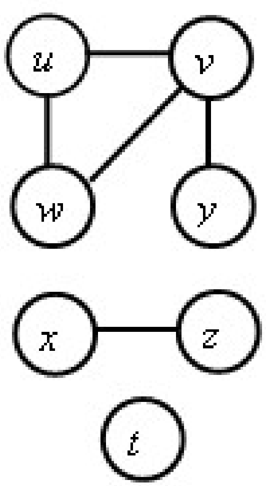
3. Adjacency List

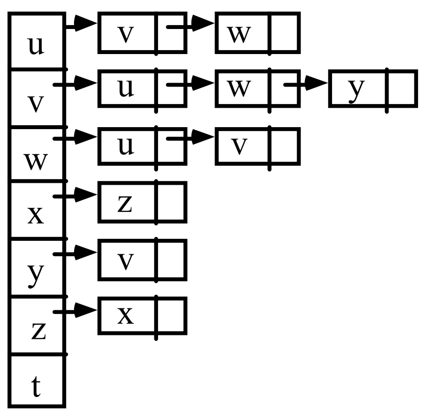
Adjacency list: each vertex has a list of which vertices it is adjacent

- Is an array Adjacency consiststing of $|V|$ list
- Each vertex has 1 list
- Each vertex $u \in V$: Adjacency[u] consists of nodes that are adjacent to u.

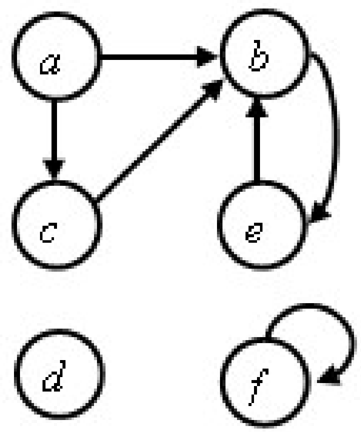
Example:

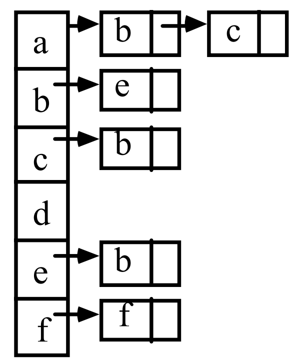
Undirected graph



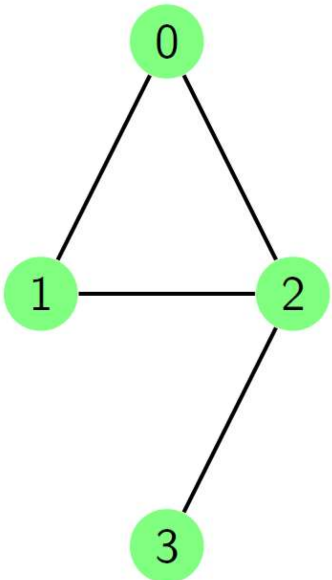


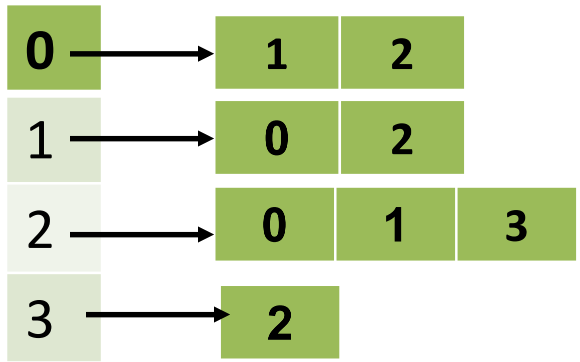
Directed graph





Representation- Adjacency List



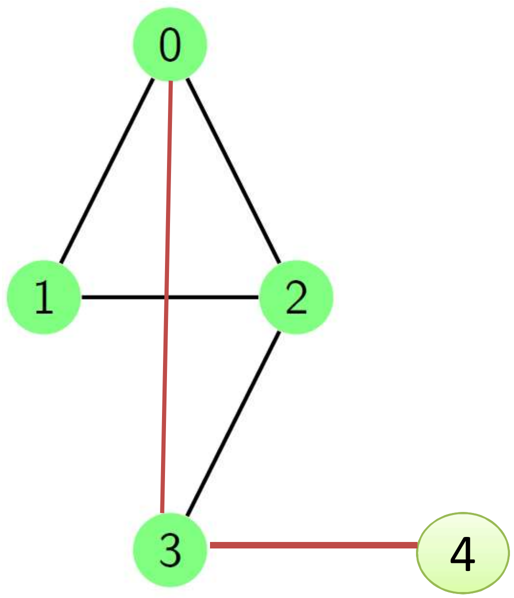



```
def print_graph():
  global graph
  for vertex in graph:
    for edges in graph[vertex]:
        print(vertex, "->", edges)

graph = {}
graph[0] = [1, 2]
graph[1] = [0, 2]
graph[2] = [0, 1, 3]
graph[3] = [2]
print ("List of edges: ")
print_graph()
```

```
List of edges:
0 -> 1
0 -> 2
1 -> 0
1 -> 2
2 -> 0
2 -> 1
2 -> 3
3 -> 2
```

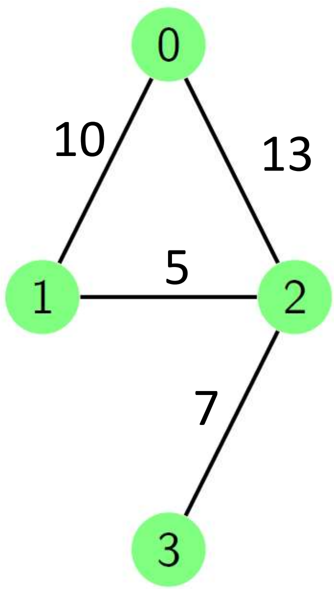

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Representation- Adjacency List




```
def print_graph():
  global graph
  for vertex in graph:
    for edges in graph[vertex]:
        print(vertex, "->", edges)

graph = {}
graph[0] = [1, 2]
graph[1] = [0, 2]
graph[2] = [0, 1, 3]
graph[3] = [2]
graph[0].append(3)
graph[3].append(0)

graph[4]=[]
graph[4].append(3)
graph[3].append(4)

print ("List of edges: ")
print_graph()
```

Representation- Adjacency List

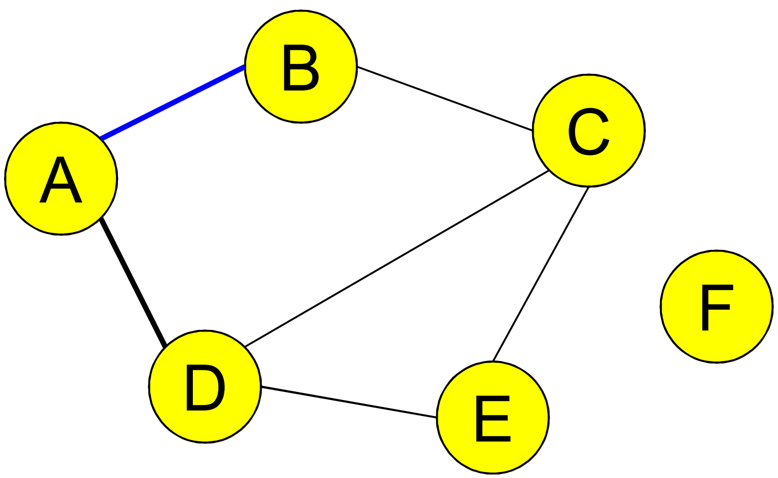


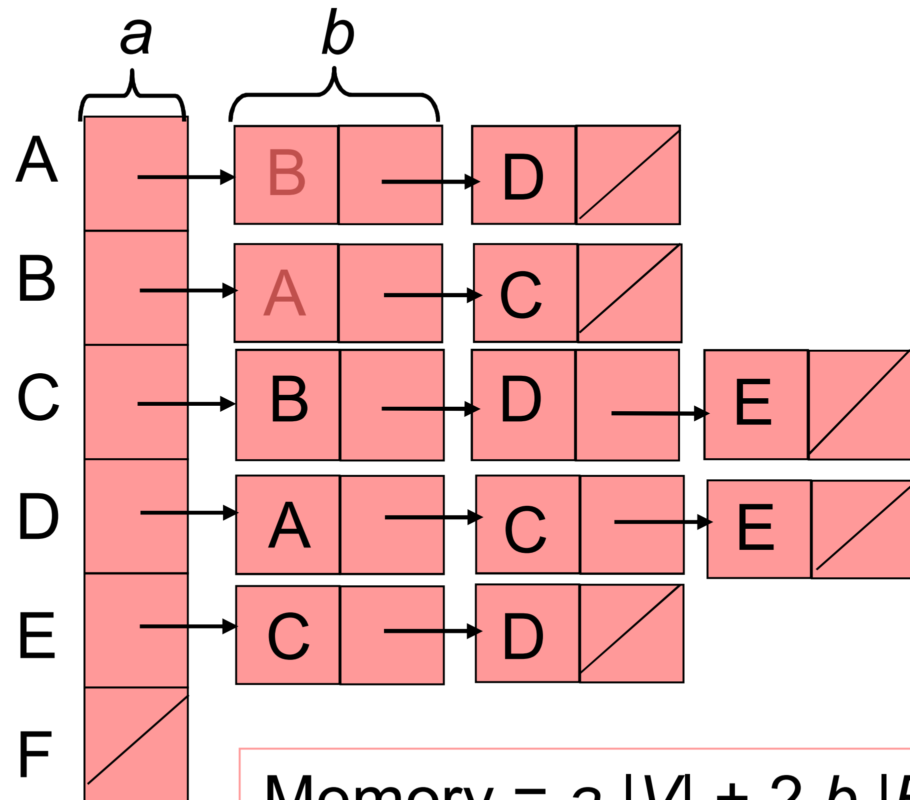

```
# Print the graph
def print_graph():
  global graph
  for vertex in graph:
    for edges in graph[vertex]:
      print(vertex, " -> ",edges[0]," edge weight: ", edges[1])

graph = {}
graph[0] = [[1, 10]]
graph[1] = [[0, 10]] #undirected graph

graph[0].append([2, 13])
graph[2] = []
graph[2].append([0, 13]) #undirected graph

graph[1].append([2, 5])
graph[2].append([1, 5]) #undirected graph

graph[2].append([3, 7])
graph[3] = []
graph[3].append([2,7]) #undirected graph
print ("List of edges: ")
print_graph()
```

NGUYỄN KHÁNH PHƯƠNG
Bộ môn KHMT – ĐHBK HN

Adjacency List of an undirected graph

Each vertex $v \in V$: Adjacency(v) = list of vertices u: $\{v, u\} \in E$

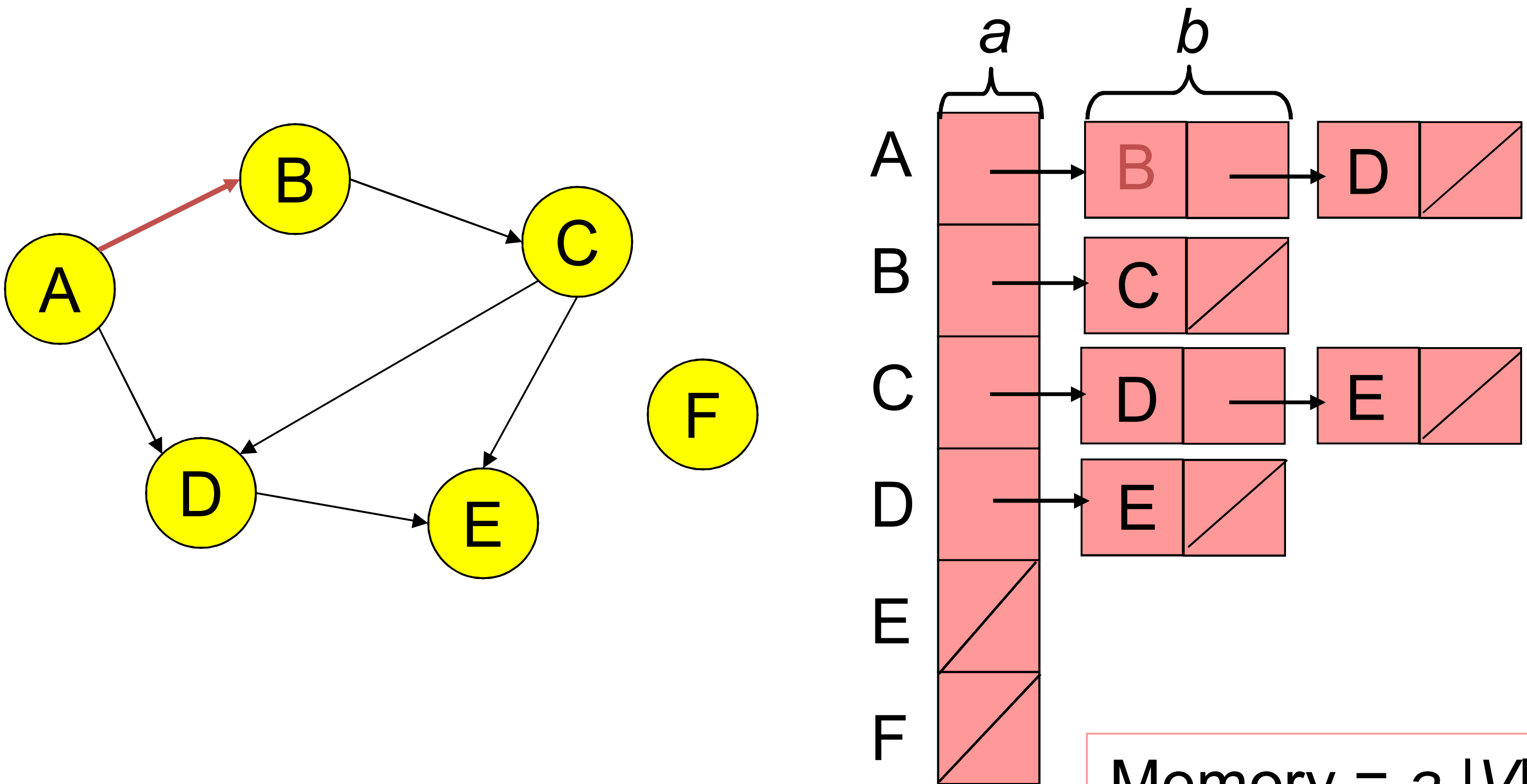




Memory = $a\,|V| + 2\,b\,|E|$

Adjacency List of a directed graph

Each vertex $v \in V$: Adjacency(v) = $\{\ u\!: (v, u) \in E\ \}$



Memory = $a\,|V| + b\,|E|$

Analyze the cost

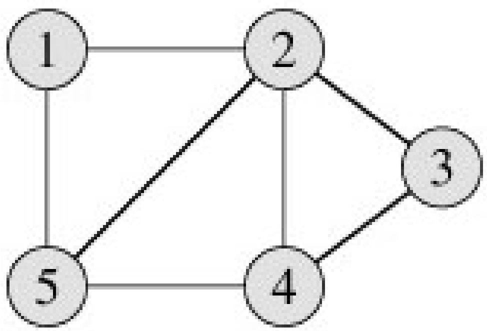
Memory Space

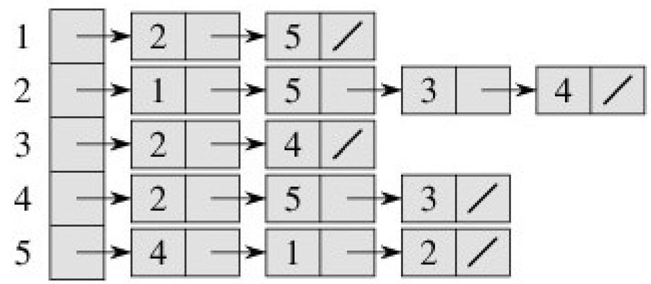
- $\Theta(|V|+|E|)$
- Is often much smaller copmpared to $|V|^2$, especially for sparse graph
- Sparse graph: $|E| \leq k\,|V|$ where $k < 10$.
- Note: Most of the graph in real-world application is sparse graph! ➔ Adjacency list representation is usually preferred since it is more efficient in representing sparse graphs.
- Time to answer the query
  - Add an edge $O(1)$
  - Delete an edge go through the Adjacency lists of initial vertex and terminal vertex
  - Enumerate all adjacent vertex of v: O(<#adjacent vertices>) (better than adjacency matrix)
  - Two vertices i and j *are adjacent*?
    - Search on the Adjacency[i]: $\Theta(\text{degree}(i))$. In the worst case $O(|V|)$ => worse than adjacency matrix

Graph Representation

- **Incidence Matrix:** Most useful when information about edges is more desirable than information about vertices.
- **Adjacency (Matrix/List):** Most useful when information about the vertices is more desirable than information about the edges. This representation is also more popular since information about the vertices is often more desirable than edges in most applications

Graph representation



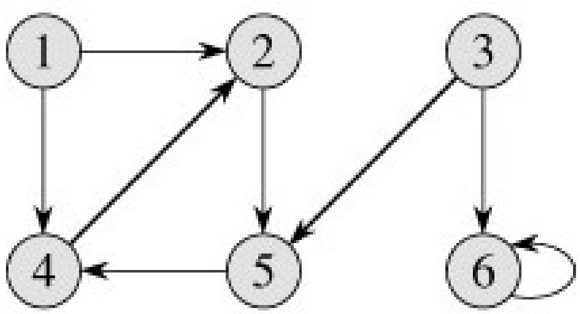


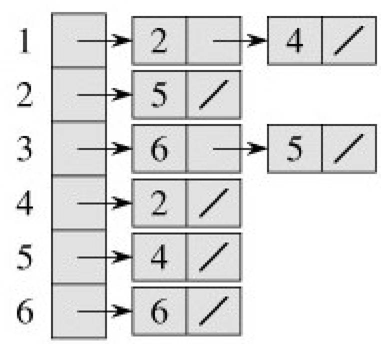
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

graph

Adjacency list

Adjacency matrix





	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1