NANYANG TECHNOLOGICAL UNIVERSITY

# CE2001/ CZ2001: Algorithms

## Graphs

Ke Yiping, Kelly



NANYANG TECHNOLOGICAL UNIVERSITY

## Table of Contents





NANYANG TECHNOLOGICAL UNIVERSITY

## Graph Terminology



NANYANG TECHNOLOGICAL UNIVERSITY

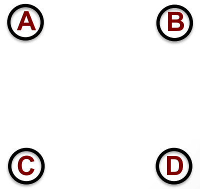
## Learning Objectives

At the end of this lecture, students should be able to:

- Use graph terminologies accurately
- Explain and use basic graph representation methods:
  - Adjacency matrix
  - Array of adjacency lists
- Compare the strengths and weaknesses between the two graph representation methods

# Graph Terminology

- A graph G is defined formally as **G = (V, E)**
- **V** is a set of **vertices**
  - **vertex** also called **node**, **point**
  - notation: V = {$V_1$, $V_2$, . . . , $V_n$}



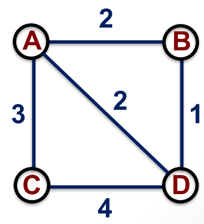





# Graph Terminology

- **E** is a set of **edges**
  - also called **arc**, **link**
  - notation: E={(x, y) | x≠y, x∈V, y∈V}
  - edges may be labelled with numerical values called **weight** or **cost** by a function W: E → R

no self-loop



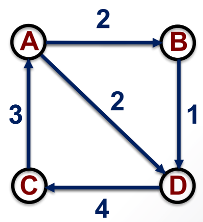

**In this case, it is called a weighted graph.**





# Graph Terminology

- If *e* = (*x, y*) is an edge in an undirected graph, then *e* is **incident** with *x* and *y*; *x* is **adjacent** to *y* and vice versa.
- If edge (*x, y*) is unordered, then G is **undirected**; otherwise, G is a **directed** graph.
- If *e* = (*x, y*) is an edge in a directed graph, then *y* can be reached from *x* through one edge, so target *y* is **adjacent** to source *x* (but it doesn't mean *x* is adjacent to *y*).

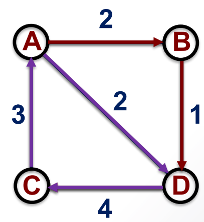







# Graph Terminology

- A **path** is a sequence of distinct vertices, each adjacent to the predecessor (except for the first vertex)
  - E.g. **ABDC**
- A **cycle** is a path containing at least three vertices such that the last vertex on the path is the same as the first
  - E.g. **ADCA**



## Graph Terminology

- An undirected graph is **connected** if there is a path from any vertex to any other vertex.
- A directed graph is **strongly connected** if there is a path from any vertex to any other vertex.
- A graph is **cyclic** if it contains one or more cycles; otherwise it is **acyclic**.
- A **complete** graph on n vertices is a simple undirected graph that contains exactly one edge between each pair of distinct vertices. The number of edges in a complete graph is **n(n-1)/2**.



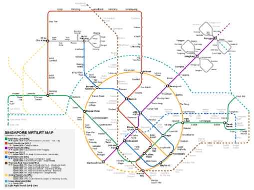


## Graph Terminology

### Application Areas

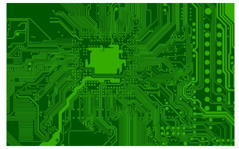
**Maps**

- V = {stations}
- E = {underground route}




**Electrical circuits**

- V = {electrical devices}
- E = {linkage between devices}



References:
- Aforl. (2014). A map of Singapore's Mass Rapid Transit (MRT) and Light Rail Transit (LRT) systems [Image]. Retrieved from https://commons.wikimedia.org/wiki/File:Singapore_MRT_and_LRT_System_Map.svg
- File: Electric circuit [Image]. (2013). Retrieved from https://pixabay.com/en/board-chip-circuit-electric-158973



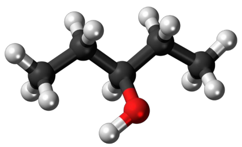


## Graph Terminology

### Application Areas

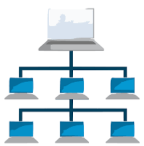
**Organic Chemistry**

- V = {atoms}
- E = {bonds between atoms}



**Computer Networks**

- V = {computers}
- E = {connections between computers}



References:
- Chemistry-atoms [Image]. (2015). Retrieved from https://pixabay.com/en/pentanol-molecule-chemistry-atoms-867210/
- Network tiered LAN server [Image]. (2014). Retrieved from https://pixabay.com/en/computer-network-tiered-lan-server-311339/



# Graph Representations

# Graph Representations

## Adjacency Matrix Representation

**Declare a 2-D array:** int A[N][N]; **/*N is no. of nodes*/**

- Edge (u, v) ∈ E implies A[u][v] == 1;
  otherwise, A[u][v] == 0.
- If a graph is directed, then A[u][v] == 1 iff (u, v)∈E;
  it does not imply (v, u) ∈ E.
- If a graph is undirected, then A is symmetric,
  i.e., A[u][v] == A[v][u].





# Graph Representations

## Adjacency Matrix Representation

**Performance:**

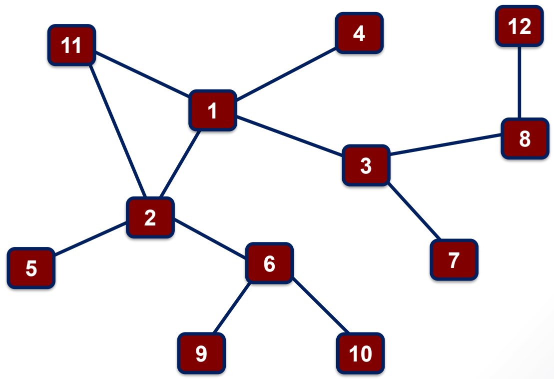
- Good because access time for A[u][v] is constant.
- Bad when graph is sparsely connected,
  i.e., most of the entries in A are zeros (|E| « $|V|^2$).





# Graph Representations

## Undirected Graph








# Graph Representations

## Adjacency Matrix Representation

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **2** | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| **3** | 1 | 0 | 0 | 0 | 0 | 0 | $\boxed{1}$ | 1 | 0 | 0 | 0 | 0 |
| **4** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **7** | 0 | 0 | $\boxed{1}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **9** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **10** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **11** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **12** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Lower left triangle is symmetric with upper right one.

# Graph Representations

## Array of Adjacency Lists

- Use an array to represent the vertices
- For each vertex, use a linked list to represent the connections to other vertices
- Commonly used, flexible structure
- If the edges have different weights, they can be stored in the nodes of the linked lists

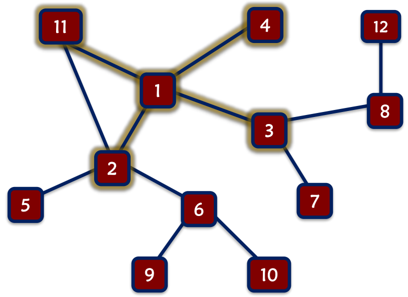




# Graph Representations

## Array of Adjacency Lists

Based on the same undirected graph seen earlier:




1 → 2 → 3 → 4 → 11
2 → 11 → 1 → 5 → 6
3 → 1 → 8 → 7
4 → 1
5 → 2
6 → 10 → 9 → 2
7 → 3
8 → 12 → 3
9 → 6
10 → 6
11 → 2 → 1
12 → 8

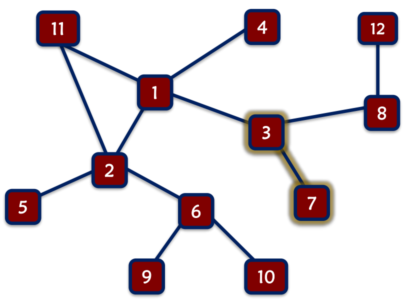




# Graph Representations

## Array of Adjacency Lists

Based on the same undirected graph seen earlier:




1 → 2 → 3 → 4 → 11
2 → 11 → 1 → 5 → 6
3 → 1 → 8 → 7
4 → 1
5 → 2
6 → 10 → 9 → 2
7 → 3
8 → 12 → 3
9 → 6
10 → 6
11 → 2 → 1
12 → 8





# Graph Representations

**Adjacency Matrix**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 9 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

**Array of Adjacency Lists**

1 → 2 → 3 → 4 → 11
2 → 11 → 1 → 5 → 6
3 → 1 → 8 → 7
4 → 1
5 → 2
6 → 10 → 9 → 2
7 → 3
8 → 12 → 3
9 → 6
10 → 6
11 → 2 → 1
12 → 8

# Represent Weighted Graphs

- If $G = (V, E, W)$ is a weighted graph, the weights of edges can be stored in the data structures.
- In the adjacency matrix, the element at the $i$-th row and the $j$-th column can be defined as:

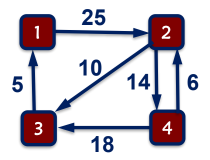
$$A[i][j] = \begin{cases} W(v_i v_j) & \text{if } (v_i, vj) \in E \\ c & \text{otherwise} \end{cases}$$

- Constant $c$ can be defined as 0 (weight as capacity) or some very large number $\infty$ (weight as cost)
- In the array of adjacency lists, the weight can be stored as a data field in each list node





# Example of Weighted Graph




| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 25 | 0 | 0 |
| 2 | 0 | 0 | 10 | 14 |
| 3 | 5 | 0 | 0 | 0 |
| 4 | 0 | 6 | 18 | 0 |

1 → (2, 25)
2 → (3, 10) → (4, 14)
3 → (1, 5)
4 → (2, 6) → (3, 18)





# Summary

- This lecture is a basic introduction to graphs
- Concepts and terminologies of graph, such as
  - A graph consists of a set of vertices and a set of edges
  - Directed vs. undirected graphs
  - The definitions of path and cycle, etc.
- Two data structures used to represent graphs:
  - Adjacency matrix
  - Array of adjacency lists
  - Their advantages and disadvantages for different applications