NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CX1107 Data Structures and Algorithms **2020/21 Semester 2**

10: Basic Graph

School of Computer Science and Engineering *Nanyang Technological University*

This is the outline of this lecture on graphs algorithms. First, we need to be familiar with the graph terminology. It is the vocabulary for technical communication about graphs. Secondly, we will learn how to represent a graph inside a computer, where we will see what kind of data structures we can use to represent a graph.

10.1 Graph Terminology

To begin with, let us first learn the graph terminologies, which form the foundation for learning the graph algorithms. Just like memorizing the English alphabet is the first step of learning English, learning graph terminologies is prerequisite for understanding graph algorithms.

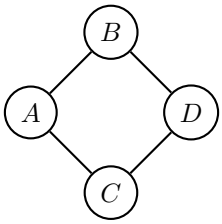
Definition 10.1 *A graph $G = (V, E)$ is a mathematical structure consisting of two finite sets, V and E. The elements of V are known as **vertices** (or **nodes**, **points**), denoted as $V = \{v_1, v_2, \ldots, v_n\}$. The elements of E are called **edges** (or **arcs**, **links**), denoted as $E = \{(x, y)|x, y \in V\}$. Each edge has a set of one or two vertices associated to it, which are called its **endpoints**.*

If we do not consider the graph with self-loop edges, we can redefine the edge as $E = \{(x, y)|x \neq y, x \in V, y \in V\}$. The number of vertices denotes as $|V|$ and the number edges which ranges from 0 to $\frac{|V|(|V|-1)}{2}$ denotes as $|E|$.

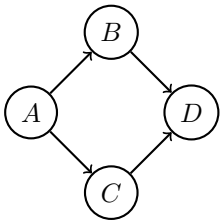
If an edge between vertices x and y is in an **undirected graph**, then we say that edge e is **incident** with x and y. So incident means an edge is connected with vertices. Then, for the connectivity between two vertices, we say that vertex x is **adjacent** to vertex y, and vice versa. If every edge in a graph G, say (x, y), is unordered, then we say that graph G is **undirected**. Otherwise, if the pair of vertices is ordered, then we call G a **directed graph**.

For example, the relationship of friendship is undirected, since it is a mutual relationship. I am your friend, and you are my friend. But the relationship between a father and his son is directed; you cannot reverse it. To draw a directed graph, we add an arrow to each edge to represent its direction. In a directed graph, if there is an edge e equal to a pair of vertices (x, y), then it means that y can be reached from x through the edge e. In this case, we say that y is adjacent to x. In other words, the target vertex is adjacent to the source vertex. However, it does not mean that x is adjacent to y. So the term adjacent implies the direction of an edge.

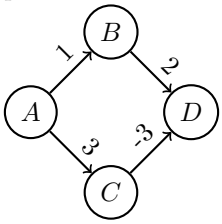
A **weighted graph** is a graph in which each edge is labelled with a numerical values known as weight or cost by $W : E \to R$. A **path** is a sequence of distinct vertices, each adjacent to the predecessor (except for the first vertex). A **cycle** is a path containing at least three vertices such that the last vertex on the path is the same as the first.



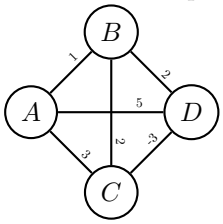
An Undirected Graph With Four Nodes And Four Edges



A Directed Graph



A Weighted Directed Graph



A Complete Weighted Graph

A graph is called **cyclic** if it contains one or more cycles. Otherwise, it is called **acyclic**. A typical example of an acyclic graph is the tree structure.

An undirected graph is called **connected**, if starting from any vertex, there is a path to any other vertex. And for a directed graph, we have a directed version called **strongly connected**. So a directed graph is called strongly connected if there is a path, which should be a directed path, from any vertex to any other vertex. Here, a directed path means every edge on the path must have the same direction along the path, following the order in the sequence of vertices.

A complete graph is a simple graph that contains exactly one edge between every two distinct vertices. In a complete graph with $|V|$ vertices, the number of edges is equal to $\frac{|V|(|V|-1)}{2}$. This is actually the formula of $|V|$ choose 2, $\binom{|V|}{2}$, which means in how many ways we can choose two different vertices from the collection of $|V|$ vertices. Because in a complete graph, each vertex is connected with the other $|V|$–1 vertices, so in total we can make n times $|V|$–1 connections. But in the undirected graph, we consider (x, y) and (y, x) as the same edge, so each edge is actually counted twice. To correct for the double counting, we need to divide it by two. That is how we get this formula for the number of edges in a complete graph.

10.1.1 Graph Application

First, a map can be represented as a graph, in which the vertices are the train stations in the MRT (mass rapid transit) system of Singapore, and the edges are the underground route. Also, we can use graphs to represent electrical circuits. Here the vertices are electrical devices, and the edges represent linkage between the devices. In organic chemistry, the vertices are atoms and edges are bonds between atoms. In computational chemistry, scientists use graph algorithms to search for interesting patterns in the chemical structures. Of course, the computer networks which are familiar to us are also represented by graphs. The vertices correspond to computers and the edges are connections between computers in the network. So you can see that the mathematical concept of graphs has broad applications in many different fields.

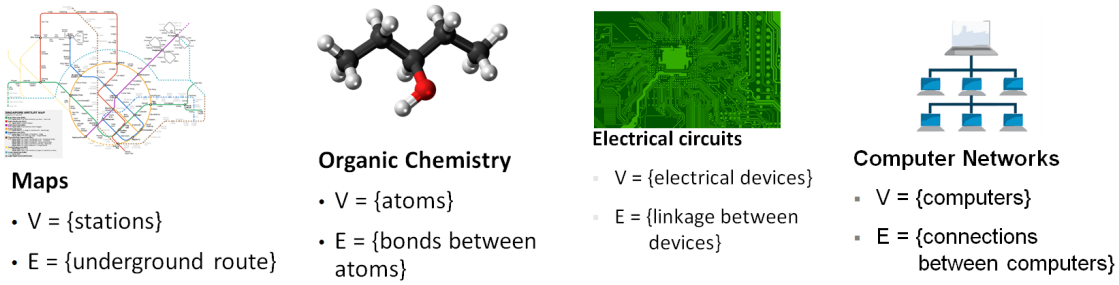


Figure 10.1: Some Graph Application

10.2 Graph Representation

In some real applications, the graphs could be huge, consisting of millions even billions of nodes, data structures for the graphs should be very efficient. Here we introduce two types of graph representation

- Adjacency Matrix
- Adjacency List

10.2.1 Adjacency Matrix

Basically it is a two-dimensional array of $|V|$ rows and $|V|$ columns, where $|V|$ is the number of vertices in a graph.

How can the structure of a graph map to the data in an adjacency matrix? If there is an edge between vertices u and v, then the corresponding entry in the matrix will be set to 1. The entry for the edge is located at the row corresponding to vertex u, and at the column corresponding to vertex v of the matrix. Otherwise we will put 0 to the entry. If the graph is directed, then the order of two vertices for an edge will matter. So for the edge from u to v, we will put the entry at u^{th} row and v^{th} column to 1. But it does not mean that the edge from v to u exists in the graph; for that we need to check the entry at v^{th} row and u^{th} column. For an undirected graph, the adjacency matrix should be symmetric. That means, $A[u][v] = A[v][u]$. Either both are 1 or both are 0, because the order of two vertices does not matter for an edge in an undirected graph. The strength of the adjacency matrix is that it takes only constant time to access an entry (or an edge) when the indices of the vertices are given. This is because an adjacency matrix is implemented as a two-dimensional array. However, you might waste space if the graph is **sparse**, which means there are not many edges. Even so, we still need $|V|^2$ entries. A lot of entries will contain zeros, meaning that the two vertices are not connected.

If the graph is a weighted graph, denoted as $G(V, E, W)$, the weights can be stored in the adjacency matrix. eg. $A[u][v] = W(u, v)$ where $W(u, v)$ is the weight of $(u, v) \in E$.

Depending on the application, we will assign either 0 or ∞ to elements of the matrix which its corresponding edge $(u, v) \notin E$.

row-to-column: if 9→6, then (9,6)

Vertices.

If weighted matrix: symmetric

	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

Table 10.1: The Adjacency Matrix of The Graph in Figure 10.2

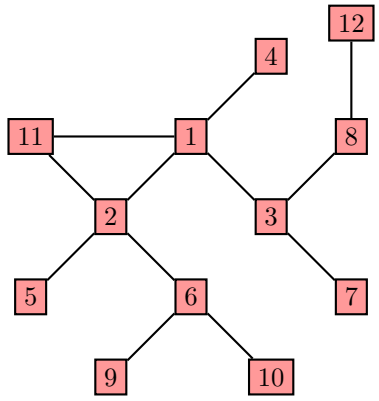


Figure 10.2: An example of a 12-node graph

10.2.2 Adjacency List

The second graph representation is called the **array of adjacency lists**. The idea is that we use an array to represent the vertices, where each entry of the array corresponds to a vertex in the graph. Then, for each vertex, we use an adjacency list to represent which other vertices are connected with this vertex. In other words, the neighbours of a vertex will be put in the linked list following the array entry corresponding to that vertex. This is a flexible data structure. The length of an adjacency list is not fixed but it is dependent on the actual connectivity of a vertex. If two vertices are not connected, we do not need to represent their relation in the linked list at all. In addition, we can store additional information, such as the edge weights, into the linked list nodes. Note that, here the node does not mean the vertex in a graph, but a node structure in a linked list.

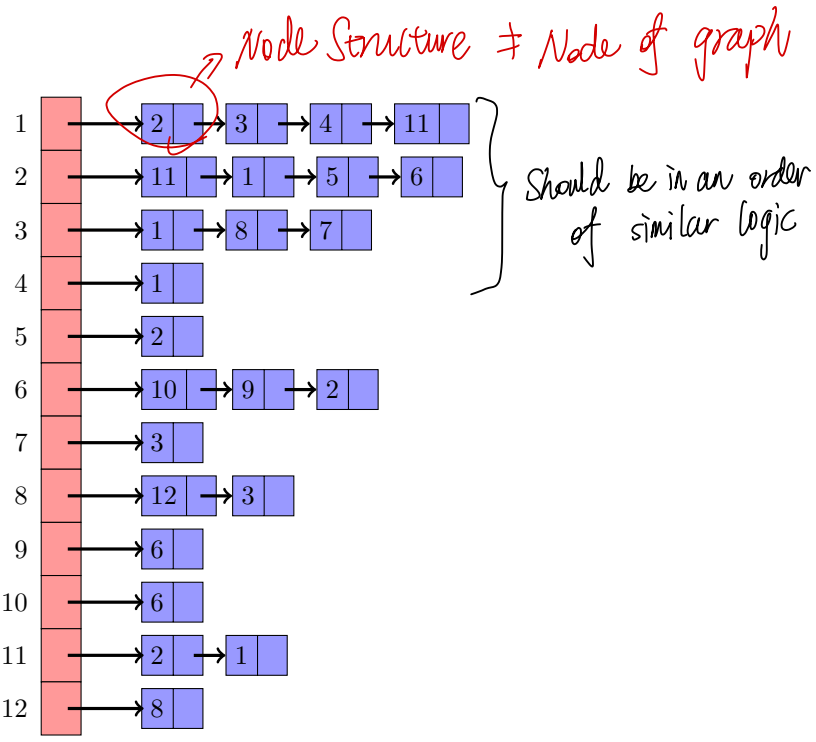


Figure 10.3: The Adjacency List of The Graph in Figure 10.2

10.2.3 Example of A Weighted Directed Graph

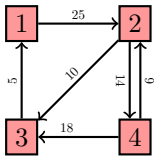


Figure 10.4: An example of a weighted directed graph

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

Table 10.2: The Adjacency Matrix of The Graph in Figure 10.4

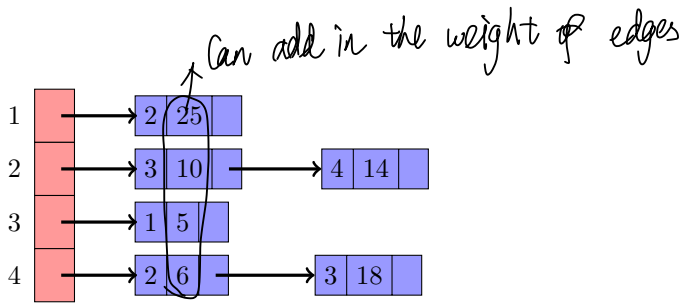


Figure 10.5: The Adjacency List of The Graph in Figure 10.4