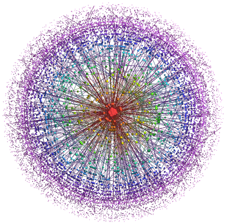
THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Algorithms and Data Structures

Lecture 10 Graphs: A Brief Recap

Jiamou Liu
The University of Auckland



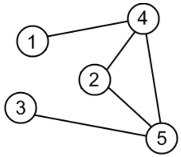
"Graphs are among the most ubiquitous models of both natural and human-made structures. They can be used to model many types of relations and process dynamics in physical, biological and social systems. Many problems of practical interest can be represented by graphs.

In computer science, graphs are used to represent networks of communication, data organization, computational devices, the flow of computation, etc. One practical example: The link structure of a website could be represented by a directed graph. The vertices are the web pages available at the website and a directed edge from page A to page B exists if and only if A contains a link to B. A similar approach can be taken to problems in travel, biology, computer chip design, and many other fields. The development of algorithms to handle graphs is therefore of major interest in computer science. "

WIKIPEDIA
The Free Encyclopedia

Definition [Graphs]

A graph is a pair $G = (V, E)$ where V is a finite set of nodes (or vertices) and E is a (possibly empty) set of unordered pairs of distinct vertices of G called edges.



$V=\{1,2,3,4,5\}$

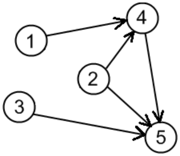
$E=\{ \{1,4\}, \{2,4\}, \{3,5\}, \{4,5\}, \{2,5\} \}$

Note:

- Having multiple edges between two vertices is not allowed.
- Self-loops are not allowed, i.e., E is an anti-reflexive relation.
- Each edge is bi-directional, i.e., E is a symmetric relation.

Definition [Digraphs]

A directed graph (or simply called a digraph) is a pair $G = (V, E)$ where V is a finite set of vertices (or nodes) and $E \subseteq V^2 \setminus \{(v, v) \mid v \in V\}$; elements in E are called directed edges (or arcs, or edges). We denote an edge from u to v using (u, v).



$V=\{1,2,3,4,5\}$

$E=\{ (1,4), (2,4), (3,5), (4,5) (2,5) \}$

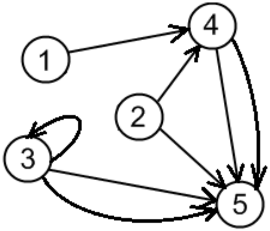
Note:

- Having multiple edges between two vertices is not allowed.
- Self-loops are not allowed, i.e., E is an anti-reflexive relation.
- Any graph is also a digraph, by considering every (undirected) edge as two arcs.

Variants of Graphs

We may enrich the notion of graphs in the following ways:

- Multigraphs: $G = (V, E)$ where $E \subseteq V^2$ is a multiset, i.e., a set where each element can appear more than once.
- Weighted (di)graphs: $G = (V, E, f, D)$ where (V, E) is a (di)graph and $f : E \to D$ is a weight function.



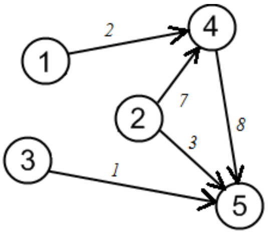
$V=\{1,2,3,4,5\}$

$E=\{(1,4),(2,4),(4,5)$
$(4,5),(2,5),(3,5)$
$(3,5),(3,3)\}$

Variants of Graphs

We may enrich the notion of graphs in the following ways:

- Multigraphs: $G = (V, E)$ where $E \subseteq V^2$ is a multiset, i.e., a set where each element can appear more than once.
- Weighted (di)graphs: $G = (V, E, f, D)$ where (V, E) is a (di)graph and $f : E \rightarrow D$ is a weight function.



$V=\{1,2,3,4,5\}$

$E=\{ (1,4), (2,4), (3,5), (4,5), (2,5) \}$

$f{:}E \rightarrow \mathbb{N}$ where

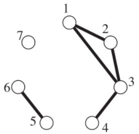
$f(1,4)=2, f(2,4)=7, f(4,5)=8$

$f(2,5)=3, f(3,5)=1$

Let $G = (V, E)$ be a graph.

- The order of a graph G is the number of nodes in G, that is, $|V|$; it is customary to use the letter n to denote the order.
- it is customary to use the letter m to denote the number of edges, i.e $m = |E|$.
- We say that u is adjacent to v if $\{u, v\} \in E$; in this case, u, v are neighbours.
- If $\{u, v\}$ is an edge, we call u and v the endpoints of the edge.

Example:

	$V=\{1,2,3,4,5,6,7\}$ $E=\{\{1,2\},\{2,3\},\{1,3\},$ $\{3,4\},\{5,6\}\}$	*e.g.* *2 is adjacent to 3* *3 is adjacent to 2* *3 is adjacent to 4* *2 is not adjacent to 4*

Terminology

For (undirected) graph:

- The degree of a vertex is the number of its adjacent vertices.
- A walk is a sequence of vertices where each consecutive pair of vertices are joined by an edge.
- A path is a walk where no vertex is repeated.

For directed graph:

- The in-degree of a node is the number of incoming neighbours of the node.
- The out-degree of a node is the number of outgoing neighbours of the node.
- A (directed) walk is a sequence of nodes where each consecutive pair of nodes are connected by a directed edge.
- A (directed) path is a walk in which no node is repeated.

We define some common special types of graphs.

- (Complete graphs) A graph is complete if all pairs of distinct nodes are adjacent in G. A complete graph on n nodes is denoted by K_n.
- (Path graphs) A path graph is $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$, and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}\}$. A path graph on n nodes is denoted by P_n.
- (Cycle graphs) A cycle graph is a graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$, and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$; this cycle graph is denoted by C_n.
- (Wheel graphs) A wheel graph is a graph $G = (V, E)$ where $V = \{u, v_1, v_2, \ldots, v_n\}$, and

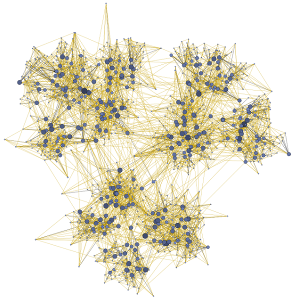
$$\begin{aligned} E = &\{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\} \cup \\ &\{\{u, v_1\}, \{u, v_2\}, \ldots, \{u, v_n\}\}. \end{aligned}$$

  This wheel graph is denoted by W_n.

Modeling with Graphs

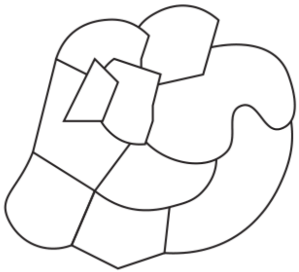
Many things interested by computer scientists are graphs:

- Network topology
- Binary relations
- Hyperlinks between webpages
- Transition diagram of systems
- Models of dependency between events



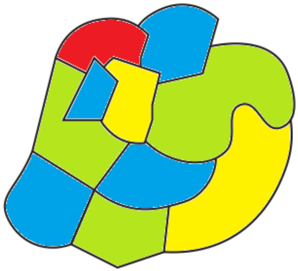
A Problem of Maps

Question 1 [Map Colouring] You are a cartographer charged with designing a map of a continent. You should fill regions using various colours, but want to use as few colours as possible, while giving different colours to neighbouring countries. How can you colour all the regions?



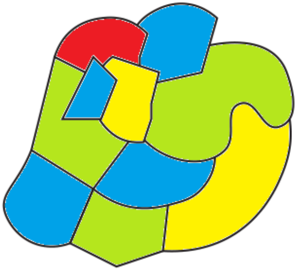
A Problem of Maps

Question 1 [Map Colouring] You are a cartographer charged with designing a map of a continent. You should fill regions using various colours, but want to use as few colours as possible, while giving different colours to neighbouring countries. How can you colour all the regions?



A Problem of Maps

Question 1 [Map Colouring] You are a cartographer charged with designing a map of a continent. You should fill regions using various colours, but want to use as few colours as possible, while giving different colours to neighbouring countries. How can you colour all the regions?



– Can this map be coloured with fewer than four colours?

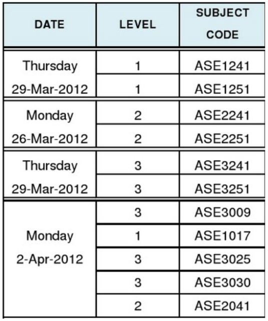
– Is there another map that requires more than four colours?

A Problem of Universities

Question 2 [Exam Scheduling]. A university needs to schedule exams at the end of the semester so that if a student is enrolled in two courses, these courses must get different examination periods. How can this be done?

A Problem of Universities

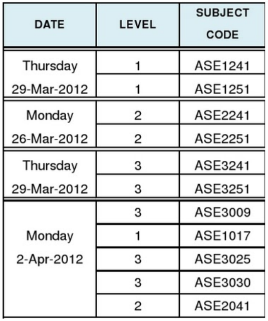
Question 2 [Exam Scheduling]. A university needs to schedule exams at the end of the semester so that if a student is enrolled in two courses, these courses must get different examination periods. How can this be done?



DATE	LEVEL	SUBJECT CODE
Thursday 29-Mar-2012	1	ASE1241
	1	ASE1251
Monday 26-Mar-2012	2	ASE2241
	2	ASE2251
Thursday 29-Mar-2012	3	ASE3241
	3	ASE3251
Monday 2-Apr-2012	3	ASE3009
	1	ASE1017
	3	ASE3025
	3	ASE3030
	2	ASE2041

A Problem of Universities

Question 2 [Exam Scheduling]. A university needs to schedule exams at the end of the semester so that if a student is enrolled in two courses, these courses must get different examination periods. How can this be done?



DATE	LEVEL	SUBJECT CODE
Thursday 29-Mar-2012	1	ASE1241
	1	ASE1251
Monday 26-Mar-2012	2	ASE2241
	2	ASE2251
Thursday 29-Mar-2012	3	ASE3241
	3	ASE3251
Monday 2-Apr-2012	3	ASE3009
	1	ASE1017
	3	ASE3025
	3	ASE3030
	2	ASE2041

We would like to use a mathematical tool, called *graph*, to connect this problem with the map colouring problem

Exam Scheduling Problem: We turn the problem into a problem on graphs.

- **Nodes**: The nodes in V are exams.
- **Edges**: Two nodes are adjacent if they “share” a student.
- **Scheduling**: A schedule is a function $s : V \to T$ that maps nodes to a set T such that $\forall\{u, v\} \in E\colon s(u) \neq s(v)$

Exam Scheduling Problem: We turn the problem into a problem on graphs.

- **Nodes**: The nodes in V are exams.
- **Edges**: Two nodes are adjacent if they “share” a student.
- **Scheduling**: A schedule is a function $s : V \to T$ that maps nodes to a set T such that $\forall \{u, v\} \in E \colon s(u) \neq s(v)$
- **Problem**: The problem asks for a smallest set T such that the graph (V, E) admits a scheduling $s : V \to C$.

Exam Scheduling Problem: We turn the problem into a problem on graphs.

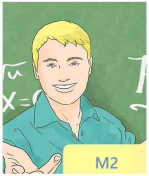
- **Nodes**: The nodes in V are exams.
- **Edges**: Two nodes are adjacent if they "share" a student.
- **Scheduling**: A schedule is a function $s : V \to T$ that maps nodes to a set T such that $\forall \{u, v\} \in E : s(u) \neq s(v)$
- **Problem**: The problem asks for a smallest set T such that the graph (V, E) admits a scheduling $s : V \to C$.

Note: From an abstract point of view, the graph colouring problem and exam scheduling problem are the same problem!

Dining Mathematicians

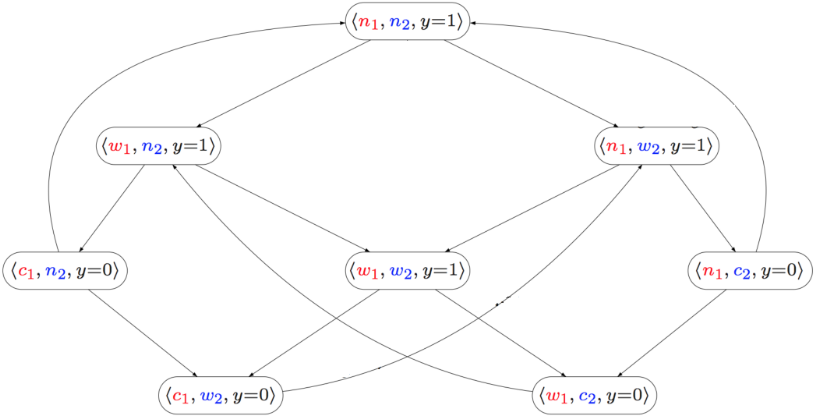
Question 3. Two mathematicians are working on different problems. Sometimes they get hungry and need to eat. However, there is only one (very large) bowl of noodles and only one person can eat at a time. They decide to share a flag. When the flag is raised, one person is eating, and the other has to wait. Each mathematician will repeat the following procedures:

```
Work on mathematics
while flag is raised do
   Wait to eat
Raise the flag
Eat!
Put down the flag
```

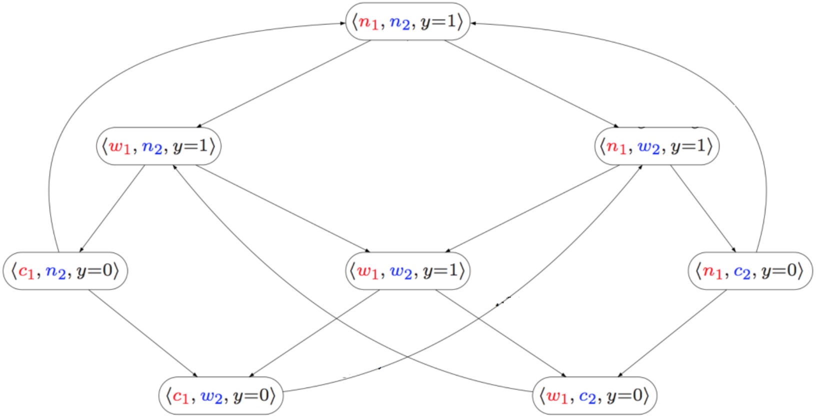


We can represent the situation by a graph:

- **States**: A state is a tuple (x_1, x_2, y) where $x_i \in \{n_i, w_i, c_i\}$ denotes the current action of mathematician $i \in \{1, 2\}$, and $y \in \{0, 1\}$ denotes the current state of the flag.
    - n_i: Mi is working
    - w_i: Mi is waiting to eat
    - c_i: Mi is eating
    - $y = 0$: The flag is raised. $y = 1$: The flag is down.
- **Nodes**: The nodes are all states.
- **Edges**: Add a directed edge from one state s_1 to another state s_2 if it is possible to transit from s_1 to s_2 in one step.

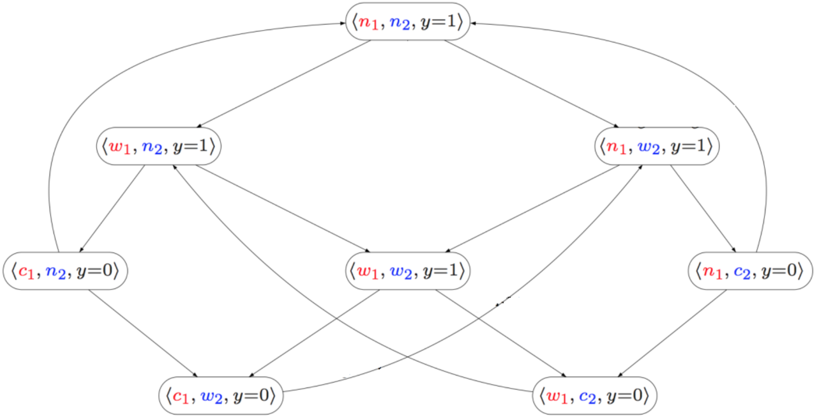

$\langle n_1, n_2, y{=}1\rangle$
$\langle w_1, n_2, y{=}1\rangle$
$\langle n_1, w_2, y{=}1\rangle$
$\langle c_1, n_2, y{=}0\rangle$
$\langle w_1, w_2, y{=}1\rangle$
$\langle n_1, c_2, y{=}0\rangle$
$\langle c_1, w_2, y{=}0\rangle$
$\langle w_1, c_2, y{=}0\rangle$

- Is it possible for two mathematicians to eat at the same time?

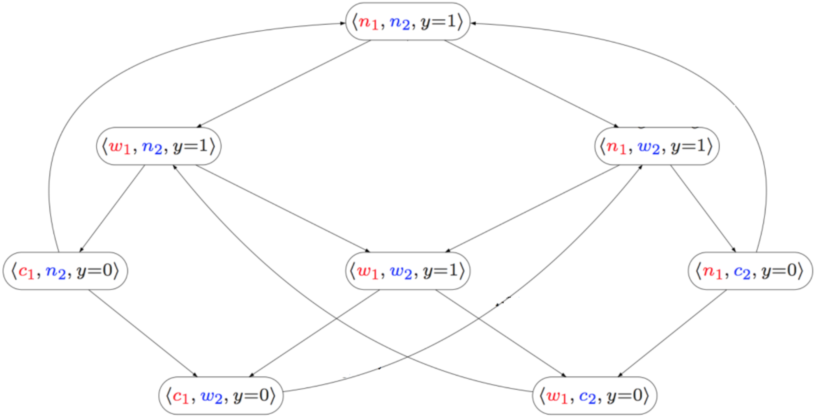


$\langle n_1, n_2, y{=}1\rangle$
$\langle w_1, n_2, y{=}1\rangle$
$\langle n_1, w_2, y{=}1\rangle$
$\langle c_1, n_2, y{=}0\rangle$
$\langle w_1, w_2, y{=}1\rangle$
$\langle n_1, c_2, y{=}0\rangle$
$\langle c_1, w_2, y{=}0\rangle$
$\langle w_1, c_2, y{=}0\rangle$

- Is it possible for two mathematicians to eat at the same time? No, since there is no state that contains both c_1, c_2.


$\langle n_1, n_2, y{=}1\rangle$
$\langle w_1, n_2, y{=}1\rangle$
$\langle n_1, w_2, y{=}1\rangle$
$\langle c_1, n_2, y{=}0\rangle$
$\langle w_1, w_2, y{=}1\rangle$
$\langle n_1, c_2, y{=}0\rangle$
$\langle c_1, w_2, y{=}0\rangle$
$\langle w_1, c_2, y{=}0\rangle$

- Is it possible for two mathematicians to eat at the same time? No, since there is no state that contains both c_1, c_2.
- Is it possible for a mathematician to starve to death?


$\langle n_1, n_2, y{=}1\rangle$
$\langle w_1, n_2, y{=}1\rangle$
$\langle n_1, w_2, y{=}1\rangle$
$\langle c_1, n_2, y{=}0\rangle$
$\langle w_1, w_2, y{=}1\rangle$
$\langle n_1, c_2, y{=}0\rangle$
$\langle c_1, w_2, y{=}0\rangle$
$\langle w_1, c_2, y{=}0\rangle$

- Is it possible for two mathematicians to eat at the same time?
  No, since there is no state that contains both c_1, c_2.
- Is it possible for a mathematician to starve to death?
  Yes, since there is a circuit in which one mathematician does not eat in any state.

Transition Diagrams

- A (software/hardware) system can be formally represented using a set of states. The operation of the system involve transitions between the states.
- A transition diagram is a digraph that represents states of a system and transitions between states.
- The goal is to verify that the system satisfies certain important properties:
  - Certain states will never be achieved.
  - Certain states will be visited repeatedly.
- The field of *formal verification* studies methods of verifying if such properties are met.

Graph as a Data Structure

Graph Representations

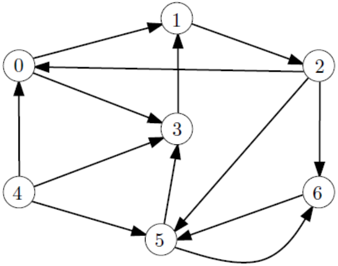
- There are two common ways of implementing a graph.
- These two implementations are based on two representations of graphs:
  1) Adjacency matrix
  2) Adjacency list
- The performance of the graph operations depends on the type of implementation we adopt.
- When we analyze graph operations, we measure the complexity in terms of the graphs' order n and size m.

Graph Representations

Suppose G is a digraph with nodes $\{0, 1, 2, \ldots, n-1\}$.

Definition [adjacency matrix]

The adjacency matrix of G is the $n \times n$ boolean matrix (encoded using 0,1) such that $(i, j) \in E$ if and only if the entry (i, j) is 1.

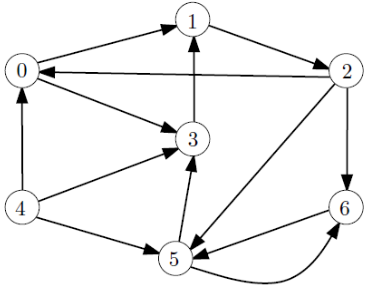


$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Suppose G is a digraph with nodes $\{0, 1, 2, \ldots, n-1\}$.

Definition [adjacency list]

The adjacency list of a digraph G is a list of n lists, $L_0, L_1, \ldots, L_{n-1}$, where the ith list L_i contains all the out-neighbors of node i.



```
7
1 3
2
0 5 6
1
0 3 5
3 6
5
```

Note: It is important to keep in mind the representations of graphs when implementing algorithm.

Size of representation

- Adjacency Matrix:
- Adjacency List:

Therefore adjacency list is suitable for sparse graphs, i.e., graphs that have must smaller number of edges than n^2.

Note: It is important to keep in mind the representations of graphs when implementing algorithm.

Size of representation

- Adjacency Matrix: $O(n^2)$
- Adjacency List: $O(n + m)$

Therefore adjacency list is suitable for sparse graphs, i.e., graphs that have must smaller number of edges than n^2.

Summary

- Basic definition: graph, digraph, multigraph, weighted graph
- Basic terminology: size, order, adjacency, neighbourhood, degree
- Basic graph types: complete graphs, cycles, etc.
- Graphs as models for: network topology, transitions, etc.
- Graph representations: adjacency matrix, adjacency list

THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND