The Graph Abstract Data Type

Joseph Chuang-Chieh Lin (林莊傑)

Department of Computer Science & Engineering,
National Taiwan Ocean University

Fall 2025

Outline

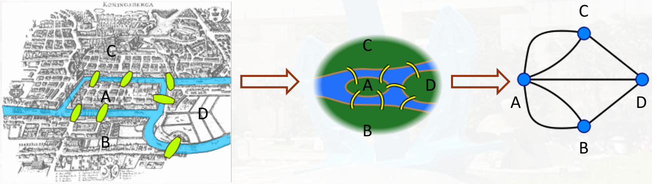
Outline

1. Introduction
   - Motivating Examples
   - Graphs
2. Graph Representations

Königsberg Bridge Problem

Question

Can we walk across all the bridges exactly once in returning back to the starting land area?

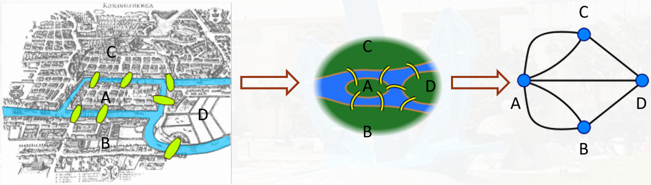


- Recall **graphs** from the Discrete Mathematics course.
  - Land $\mapsto$ vertex
  - Bridge $\mapsto$ edge

Königsberg Bridge Problem

Question (Eulerian Walk)

Can we walk across all the bridges exactly once in returning back to the starting land area?



- Recall **graphs** from the Discrete Mathematics course.
  - Land $\mapsto$ vertex
  - Bridge $\mapsto$ edge

Eulerian Walk

Euler's Theorem

A connected graph has an Euler cycle if and only if every vertex has even degree.

- **degree** of vertex v: number of neighbors of v in the graph.
- **connected**: there is a path connecting every two vertices in the graph.

Eulerian Walk

Euler's Theorem

A connected graph has an Euler cycle if and only if every vertex has even degree.

- **degree** of vertex v: number of neighbors of v in the graph.
- **connected**: there is a path connecting every two vertices in the graph.

So, what about the answer to the Königberg Bridge Problem?

Definition of a Graph

Graph

A graph $G = (V, E)$ consists of two sets V and E, such that

- V: a finite, nonempty set of vertices;
- E: a set of vertex pairs which are called edges.

Definition of a Graph

Graph

A graph $G = (V, E)$ consists of two sets V and E, such that

- V: a finite, nonempty set of vertices;
- E: a set of vertex pairs which are called edges.

- **Undirected** graph: (u, v) and (v, u) represent the same edge.
- **Directed** graph: a directed vertex pair $< u, v >$ has u as the tail and v as the head.

Definition of a Graph

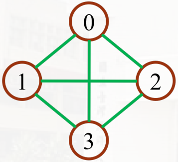
Graph

A graph $G = (V, E)$ consists of two sets V and E, such that

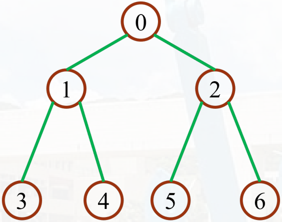
- V: a finite, nonempty set of vertices;
- E: a set of vertex pairs which are called edges.

- **Undirected** graph: (u, v) and (v, u) represent the same edge.
- **Directed** graph: a directed vertex pair $< u, v >$ has u as the tail and v as the head.
  - $< u, v >$ and $< v, u >$ indicate different edges.

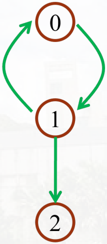
Examples



G_1



G_2



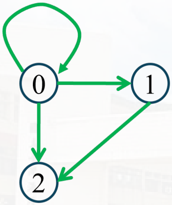
G_3

$V(G_1)=\{0, 1, 2, 3\}$
$E(G_1)=\{(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)\}$

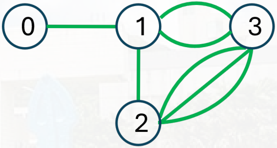
$V(G_2)=\{0, 1, 2, 3, 4, 5, 6\}$
$E(G_2)=\{(0, 1), (0, 2), (1, 3), (1, 4), (2, 5), (2, 6)\}$

$V(G_3)=\{0, 1, 2\}$
$E(G_3)=\{<0, 1>, <1, 0>, <1, 2>\}$

Self-Loop & Multigrpah



Graph with
a self-loop



multigraph

- (v, v): self-loop.
- multigraph: a graph with multiple occurrence of some edges.

Complete Graph

Complete Graph

An undirected graph $G = (V, E)$ is a complete graph if any pair of vertices (u, v) is an edge in E.

Complete Graph

Complete Graph

An undirected graph $G = (V, E)$ is a complete graph if any pair of vertices (u, v) is an edge in E.

- The number of edges in a complete undirected graph of n vertices:

Complete Graph

Complete Graph

An undirected graph $G = (V, E)$ is a complete graph if any pair of vertices (u, v) is an edge in E.

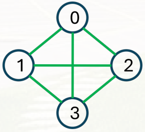
- The number of edges in a complete undirected graph of n vertices: $n(n-1)/2$.
- The number of edges in a complete directed graph of n vertices:

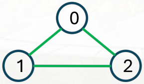
Complete Graph

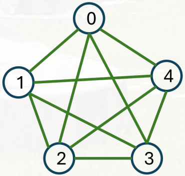
Complete Graph

An undirected graph $G = (V, E)$ is a complete graph if any pair of vertices (u, v) is an edge in E.

- The number of edges in a complete undirected graph of n vertices: $n(n-1)/2$.
- The number of edges in a complete directed graph of n vertices: $n(n-1)$.







Subgraph and Induced Subgraph

- If (u, v) is an edge in $E(G)$, then the vertices u and v are adjacent and that the edge (u, v) is incident on vertices u and v.

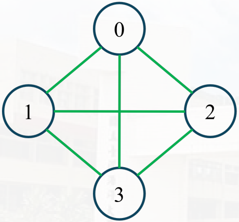
Subgraph

A subgraph of G is a graph G' such that $V(G') \subseteq V(G)$ and $E(G') \subseteq EG$.

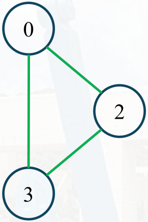
Induced Subgraph

A graph G' is an induced subgraph of G if G' is a subgraph of G and for any two vertices $u, v \in V(G')$, $(u, v) \in E(G)$ if and only if $(u, v) \in E(G')$.

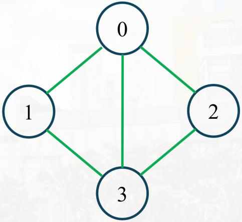
Examples



G

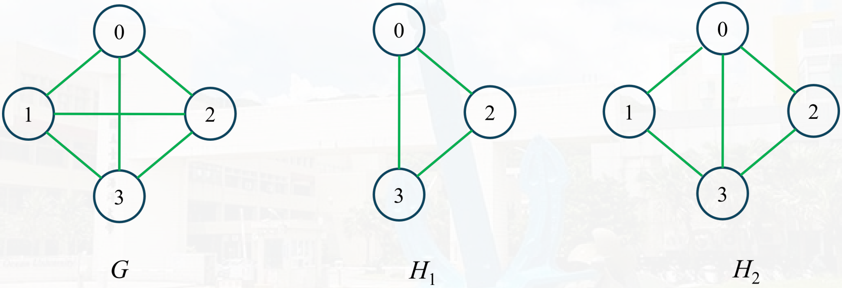


H_1



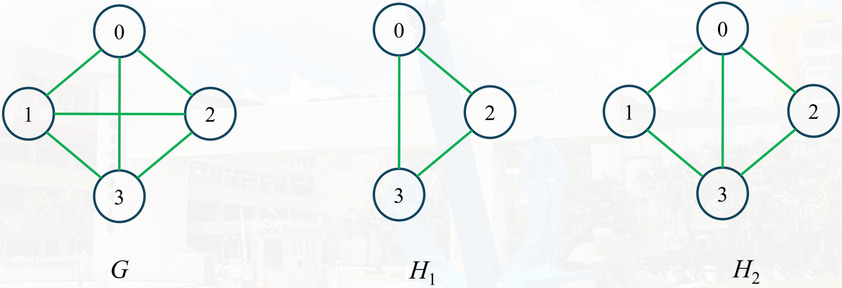
H_2

Examples



- H_1, H_2: subgraphs of G.

Examples



- H_1, H_2: subgraphs of G.
- H_1 is an induced subgraph of G, but H_2 is NOT.

Path (1/2)

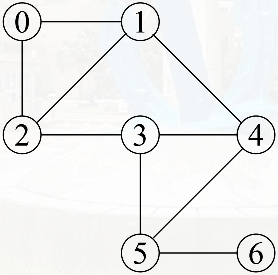
Path

A (directed or undirected) path from vertex u to vertex v in graph G is a sequence of vertices $u, i_1, i_2, \ldots, i_k, v$, such that $(u, i_1), (i_1, i_2), \ldots, (i_k, v)$ are edges in $E(G)$.

Path (1/2)

Path

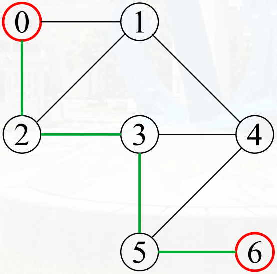
A (directed or undirected) path from vertex u to vertex v in graph G is a sequence of vertices $u, i_1, i_2, \ldots, i_k, v$, such that $(u, i_1), (i_1, i_2), \ldots, (i_k, v)$ are edges in $E(G)$.



Path (1/2)

Path

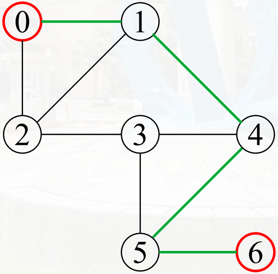
A (directed or undirected) path from vertex u to vertex v in graph G is a sequence of vertices $u, i_1, i_2, \ldots, i_k, v$, such that $(u, i_1), (i_1, i_2), \ldots, (i_k, v)$ are edges in $E(G)$.



Path (1/2)

Path

A (directed or undirected) path from vertex u to vertex v in graph G is a sequence of vertices $u, i_1, i_2, \ldots, i_k, v$, such that $(u, i_1), (i_1, i_2), \ldots, (i_k, v)$ are edges in $E(G)$.



Path (2/2)

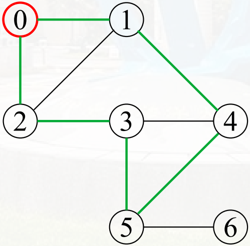
- The length of a path is the number of edges on it.
- A simple path a path in which all vertices, except possibly the first and the last, are distinct.

Path (2/2)

- The length of a path is the number of edges on it.
- A simple path a path in which all vertices, except possibly the first and the last, are distinct.
- A cycle is a simple path in which the first and last vertices are the same.

Path (2/2)

- The length of a path is the number of edges on it.
- A simple path a path in which all vertices, except possibly the first and the last, are distinct.
- A cycle is a simple path in which the first and last vertices are the same.
    - A simple path from v to v.



Connected and Connected Component

Connected

- In an undirected graph G, two vertices u and v are connected iff there is a path in G from u to v.
- An undirected graph is **connected** iff for every pair of district vertices u and v in $V(G)$ there is a path from u to v in G.

Connected Component

A **connected component** (or simply a component) H of an undirected graph is a maximal connected subgraph.

Connected and Connected Component

Connected

- In an undirected graph G, two vertices u and v are connected iff there is a path in G from u to v.
- An undirected graph is **connected** iff for every pair of district vertices u and v in $V(G)$ there is a path from u to v in G.

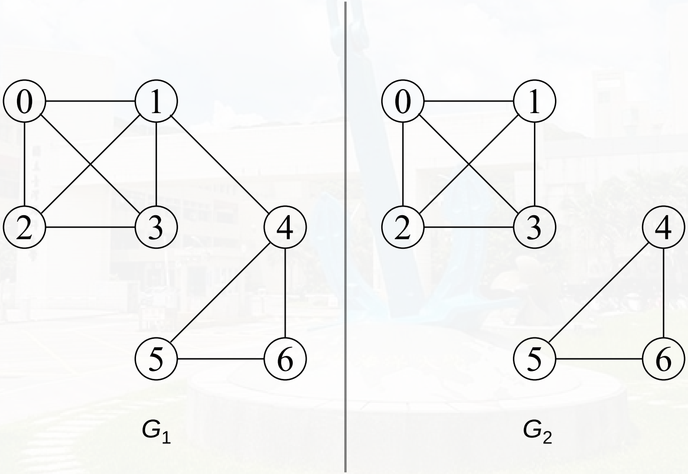
Connected Component

A **connected component** (or simply a component) H of an undirected graph is a maximal connected subgraph.

tree

A tree is a connected acyclic (i.e., has no cycles) graph.

Example of Connected Components



Strongly Connected Graph (強連通圖)

Strongly Connected Graph

A directed graph G is said to be **strongly connected** iff for every pair of district vertices $u, v \in V(G)$, there is directed path from u to v and also from v to u.

Strongly Connected Graph (強連通圖)

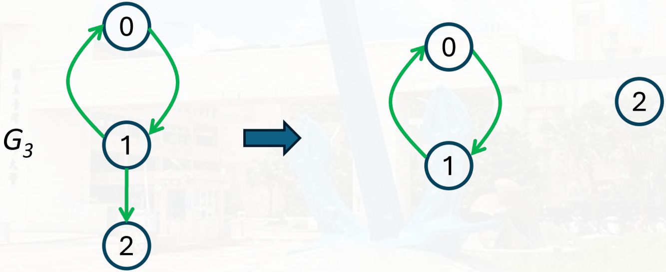
Strongly Connected Graph

A directed graph G is said to be **strongly connected** iff for every pair of district vertices $u, v \in V(G)$, there is directed path from u to v and also from v to u.

Strongly Connected Component

A strongly connected component is a maximal subgraph that is strongly connected.

Strongly Connected Components



Vertex Degree

- The degree of a vertex is the number of edges **incident to** that vertex.
- For a directed graph G,
  - The in-degree of a vertex is the number of edges for which vertex is head.
  - the out-degree of a vertex is the number of edges for which the vertex is the tail.

Vertex Degree

- The degree of a vertex is the number of edges **incident to** that vertex.
- For a directed graph G,
  - The in-degree of a vertex is the number of edges for which vertex is head.
  - the out-degree of a vertex is the number of edges for which the vertex is the tail.

Let d_i be the degree of vertex i in an n-vertex graph $G = (V, E)$, then

$$|E| = \frac{1}{2}\sum_{i=1}^{n} d_i.$$

Outline

Graph Representations

- Two most commonly used representation for a graph:

Graph Representations

- Two most commonly used representation for a graph:
  - Adjacency Matrices
  - Adjacency Lists

Graph Representations

- Two most commonly used representation for a graph:
  - Adjacency Matrices
  - Adjacency Lists
- The choice of the representation:
  - the application
  - the functions one expects to perform on the graph
  - characteristics of the input graph

Adjacency Matrix

The adjacency matrix of an n-vertex graph G is a two-dimensional $n \times n$ array a, with the property that

- $a[i][j] = 1$ iff $(i, j) \in E(G)$;
- $a[i][j] = 0$ iff there is no such edge (i, j) in G.

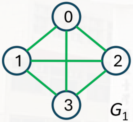
Remark

The adjacency matrix for an undirected graph is symmetric.

Adjacency Matrix (2/2)

- For an undirected graph the degree of any vertex i is its row sum.
- For an directed graph the row sum is its out-degree and the column sum is its in-degree.

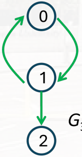
Adjacency Matrices (Examples)



G_1

The adjacency matrix of G_1

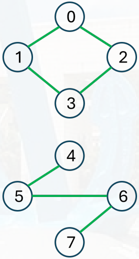
	0	1	2	3
0	0	1	1	1
1	1	0	1	1
2	1	1	0	1
3	1	1	1	0



G_3

	0	1	2
0	0	1	0
1	1	0	1
2	0	0	0

The adjacency matrix of G_3



G_4

	0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0
2	1	0	0	1	0	0	0	0
3	0	1	1	0	0	0	0	0
4	0	0	0	0	0	1	0	0
5	0	0	0	0	1	0	1	0
6	0	0	0	0	0	1	0	1
7	0	0	0	0	0	0	1	0

The adjacency matrix of G_4

Adjacency Lists

- The n rows of the adjacency matrix are represented as n chains.

Adjacency Lists

- The n rows of the adjacency matrix are represented as n chains.
- One chain for each vertex in G.

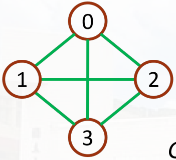
Adjacency Lists

- The n rows of the adjacency matrix are represented as n chains.
- One chain for each vertex in G.
- The nodes in chain i represent the vertices that are adjacent from vertex i.

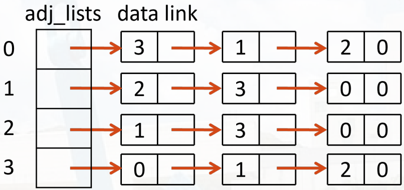
Adjacency Lists

- The n rows of the adjacency matrix are represented as n chains.
- One chain for each vertex in G.
- The nodes in chain i represent the vertices that are adjacent from vertex i.
- The data field of a chain node stores the index of an adjacent vertex.

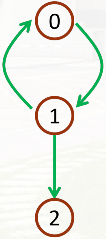
Adjacency Lists Examples



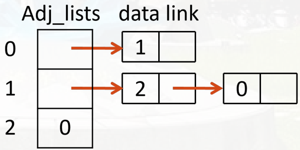
G_1



Adjacent list of G_1



G_3



Adjacent list of G_3

Remark: Weighted Edges

- In many applications, the edges of a graph have weights associated with them.
  - importance, costs, distance, etc.
- The adjacency matrix entries a[i][j] would keep this information.
- When adjacency lists are used, we can introduce an additional field **weight** in the list nodes.

```
typedef struct node Node;
struct node {
    int data;
    int weight;
    Node *link;
};
```

Discussions