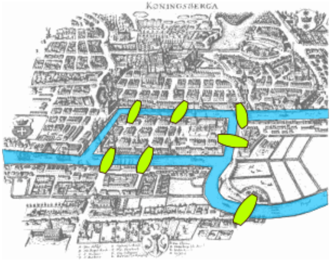
Motivation

Königsberg Seven Bridges Problem

In Königsberg, there were two islands connected to each other and the mainland by seven bridges, as shown in the figure below.



Question:
Is it possible to take a walk and cross over each bridge exactly once?

Euler showed that it is not possible, but he proved it?

(image source: `https://simple.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg`)

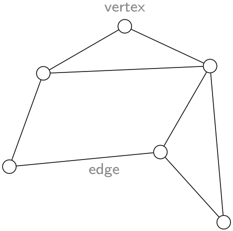
CPT108 Data Structures and Algorithms

Lecture 21

Graphs

Graphs

- One of the *MOST* useful tool in modelling problems



Vertex can be considered as "sites" or "locations"

Edge represents connections

Graphs

Applications

(image source: `https://www.travelchinaguide.com/cityguides/jiangsu/suzhou/subway/map.htm`)

Railway Travel

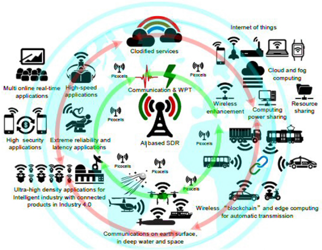
- Each vertex represent a station
- Each edge represents a direct travel between two stations
- A query on direct travel
  = a query on whether an edge exists
- A query on how to get to a location = does a path exist from station A to station B
- We can even associate costs to edges (weighted graphs), then ask “What is the cheapest path from station A to station B”

Graphs

Applications (cont.)

Wireless Communication

- Vertices are stations
- Edges represent the Euclidean distance d_{ij} between two station
- Each station uses certain power to transmit messages. Given this power *i*, only a few nodes can be reached. A station reachable by *i* then uses its own power to relay the message to other stations not reachable by *i*.
- A typical (wireless) communication problem is: how to broadcast between *all* stations such that they are all connected and the power consumption is minimized



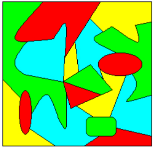
(image source: `https://www.microwavejournal.com/articles/33966-wireless-communication-beyond-5g`)

Graphs

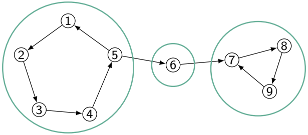
Applications (cont.)

- Graph algorithms might be very difficult!
- E.g.,

Four color problem



Strongly connected components (SCC)



Word ladder problem

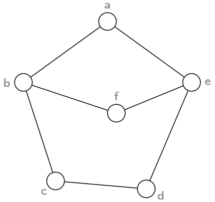
- The player is given a start word and an end word, and the player is required to change the start word into the end word progressively by substituting a single letter in each step
- e.g., if the start word is “WARM” and the end word is “COLD”, we can do it as follows:

  WARM $\rightarrow$ WARD $\rightarrow$ CARD $\rightarrow$ CORD $\rightarrow$ COLD

Graphs

Formal definitions

A graph G is specified by an ordered pair (V, E), where V is the set of vertices and E is the set of edges



$$V = \{a, b, c, d, e, f\}$$

$$E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, a\}, \{b, f\}, \{f, e\}\}$$

Terminologies

- If v_1 and v_2 are connected, then they are said to be *adjacent vertices*
  - v_1 & v_2 are *neighbors* of each other
  - v_1 & v_2 are *endpoints* of the edge $\{v_1, v_2\}$
- If an edge e is connected to v,
  then v is said to be *incident* to e.
  The edge e is *incident* to v.
- If the pair is unordered, i.e., $\{v_1, v_2\} = \{v_2, v_1\}$, the graph is *undirected*; otherwise it is *directed*
- If edge has direction, then it can be drawn as an arrow (called arc)

(v_1, v_2)

Graphs

Formal definitions: Some Examples

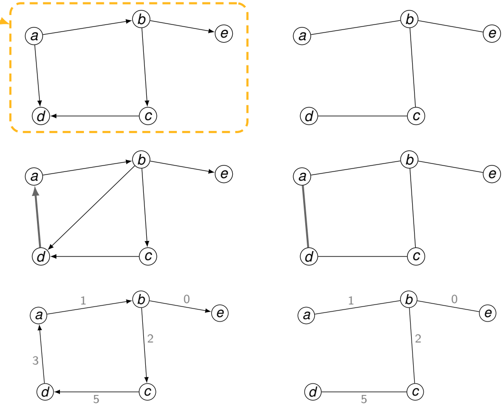
Directed Acyclic Graph — also known as "DAG"

Directed | Undirected

Acyclic:

Cyclic:

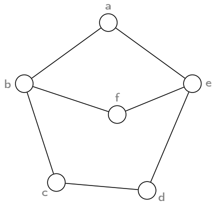
With edge labels:



Graphs

Formal definitions (cont.)

A graph G is specified by an ordered pair (V, E), where V is the set of vertices and E is the set of edges



$V = \{a, b, c, d, e, f\}$

$E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, a\}, \{b, f\}, \{f, e\}\}$

Terminologies

- Degree of a vertex v, $\texttt{deg}(v)$, is the number of edges incident to v.
  - e.g., $\texttt{deg}(a) = 2$, $\texttt{deg}(e) = 3$
- An edge $e = \{u, v\}$ of the graph contributes:
  - a count of 1 to $\texttt{deg}(u)$, and
  - another count of 1 to $\texttt{deg}(v)$
- Therefore,

$$\Sigma_{v \in V} \texttt{deg}(v) = 2m,$$

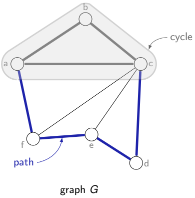
where m is the total number of edges

Graphs

Formal definitions (cont.)

Path

- A *path* is a sequence of vertices $\{v_0, \ldots, v_n\}$ such that $\{v_i, v_{i+1}\}$, $0 \leq i < n$, is an edge.
  - length, $n =$ number of edges on the path
  - e.g., the path $\{a, f, e, d, c\}$ is a path with length 4
- A cycle is a path without repeated edges leading from a node back to itself (following arrows if directed) (Or: A path is a *cycle* if and only if $v_0 = v_n$)
  - e.g., the path $\{a, b, c, a\}$ is a cycle of length 3
- A path is *simple* if and only if it does *not* contain the same vertex twice

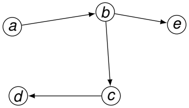


Graphs

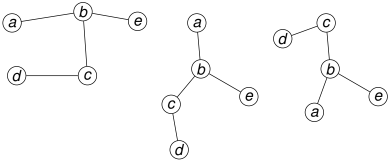
Trees are Graphs

Connectivity

- A graph is *connected* if there is a (possibly directed) path between every pair of distinct vertices
  - i.e., if one vertex of the pair is *reachable* from the other
- A *directed acyclic graph* (DAG) is a (rooted) tree *iff* it is connected, and every vertex but the root has exactly one parent



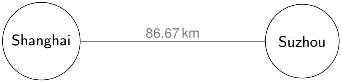
- A *connected, acyclic, undirected graph* is also called a free tree, i.e., we are free to pick any node as the root



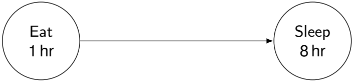
Graphs

Examples of Use

- Edge = Connecting road, with length



- Edge = Must be completed before (dependencies); Vertex label=time to complete

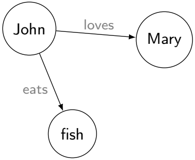


- Edge = Begat

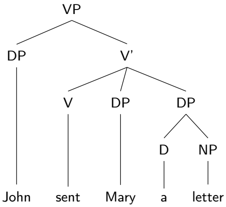
Graphs

Examples of Use (cont.)

- Edge = some relationship



- Edge = word/phrase relationship in a sentence

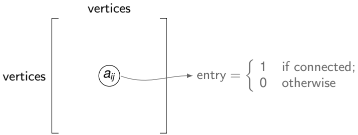


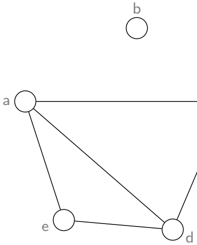
Graphs

Graph Representation

Adjacency matrix

- 2-D array, where n is the number of vertices





	a	b	c	d	e
a	0	0	1	1	1
b	0	0	0	0	0
c	1	0	0	1	0
d	1	0	1	0	1
e	1	0	0	1	0

Detect in $O(1)$ time whether two vertices are connected

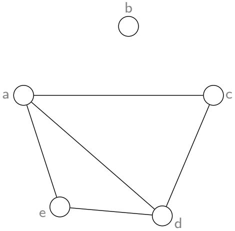
$O(n^2)$ *storage*

Graphs

Graph Representation (cont.)

Adjacency list

- If the graph is not dense, in other words, sparse, a better solution is an adjacency list
- Can be implemented using array and linked list



a → c d e

b

c → a d

d → a c e

e → a d

$O(n+m)$ *storage*, where $n = |V|$ and $m = |E|$

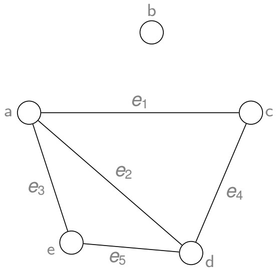
However, one cannot tell in $O(1)$ time whether two vertices are connected!

Graphs

Graph Representation (cont.)

Incidence matrix (not commonly used)

- Each edge has a name



	e_1	e_2	e_3	e_4	e_5
a	1	1	1	0	0
b	0	0	0	0	0
c	1	0	0	1	0
d	0	1	0	1	1
e	0	0	1	0	1

$O(mn)$ *storage*
where $n = |V|$ and $m = |E|$

Reading

- Chapter 20, Cormen (2022)