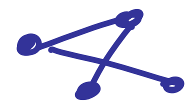
Graphs

B.4
ch. 22



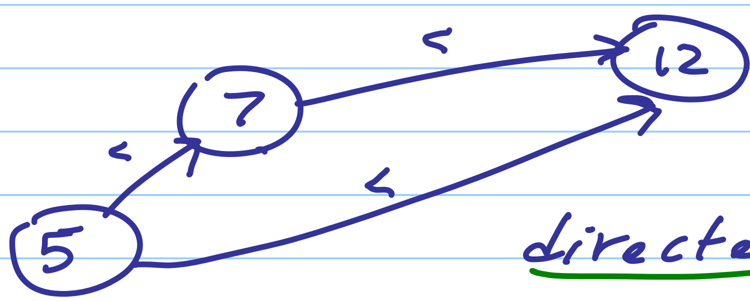
Note Title 11/27/2007

graph
- set of vertices V
- set of edges $E : V \times V$

edges must be between vertices in the defined vertex set.

many different types of relationships can be represented as graphs.

example: sorting.

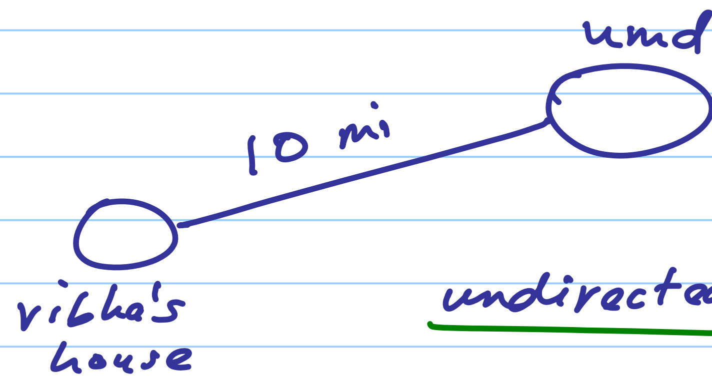


directed graph

vertices connected by an edge are adjacent

Weighted edges
are edges with a value associated with them.

example: distance in miles



undirected graph

path: list of sequentially connected edges.

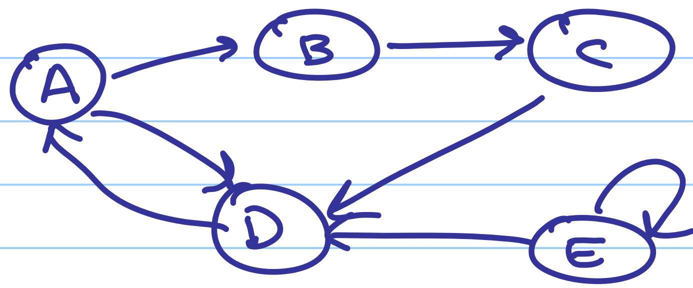
path from A to B exists → B is reachable from A.

cycle: path that starts and ends in the same place

Hamiltonian path: visit every vertex exactly once

Eulerian Path: visit every edge exactly once

Graph Representation



(1) adjacency list representation
good for sparse graphs
(when $|E| < |V|^2$)

A → B → D
B → C
C → D
D → A
E → D → E

Total memory consumed:
$|V|$ linked lists
total number of elements in all linked lists: $|E|$
so memory is $O(\max(|V|, |E|) = O(|V| + |E|)$

Adjacency Matrix Representation

good when $|E| \approx |V|^2$

to

from	A	B	C	D	E
A	0	1	0	1	0
B	0	0	1	0	0
C	0	0	0	1	0
D	1	0	0	0	0
E	0	0	0	1	1

amt of memory: $O(|V|^2)$

note: search for whether edge is present is fast.

If undirected, you can store approx. half the matrix.

Set representation.

$V = \{A, B, C, D, E\}$

$E = \{(A,B), (A,D), (B,C), (C,D), (D,A), (E,D), (E,E)\}$

Which representation is best for finding all neighbors of a vertex? adjacency list

Which representation is best to test for existance of a specific edge?

matrix

Undirected Graphs

Adjacency Matrix:

If undirected, you only need to store approx. half the matrix.

	A	B	C	D	E
A	0	1	0	1	0
B	1	0	1	0	0
C	0	1	0	1	0
D	1	0	1	0	1
E	0	0	0	1	1

Can you save space if using an ~~adjacency list~~?

A → B → D
B → ~~A~~ → C
C → B → D
D → A → C → E
E → D → E

How do you prove things about graphs?

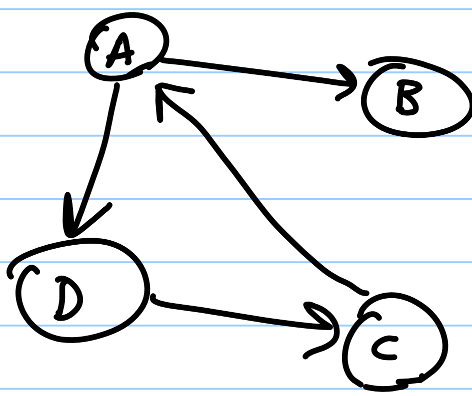
example

in-degree: # edges going into a vertex

out-degree: # edges coming out of a vertex.

Theorem:

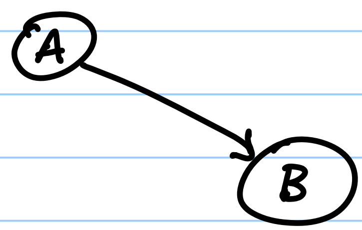
$$\sum_{i=1}^{|V|} \text{in-degree}(v_i) = \sum_{i=1}^{|V|} \text{out-degree}(v_i)$$



	in-deg	out-deg
A	1	2
B	1	$\emptyset$
C	1	1
D	1	1
	4	4

Induction Proof

Base case



$$\sum_{i=1}^{2} \text{out degree}(v_i) = \sum_{i=1}^{2} \text{indegree}(v_i) = 1$$

Inductive Hypothesis

- assume theorem holds when $|E| = k$.

Inductive Step.

$|E| = k+1$

Let G be a graph with $k+1$ edges.

Select an edge e and remove it from G to make G'.

$$\sum_{i=1}^{|G.V|} \text{indegree}(v_i) = \sum_{i=1}^{|G'.V|} \text{indegree}(v_i) + 1$$

$$\sum_{i=1}^{|G.V|} \text{outdegree}(v_i) = \sum_{i=1}^{|G'.V|} \text{outdegree}(v_i) + 1$$

From I.H.

$$\sum_{i=1}^{|G'.V|} \text{outdegree}(v_i) = \sum_{i=1}^{|G'.V|} \text{indegree}(v_i)$$

so

$$\sum_{i=1}^{|G.V|} \text{outdegree}(v_i) = \underbrace{\sum_{i=1}^{|G'.V|} \text{indegree}(v_i) + 1}_{\sum_{i=1}^{|G.V|} \text{indegree}(v_i)}.$$

Thought Question

degree of a vertex: number of edges incident to it

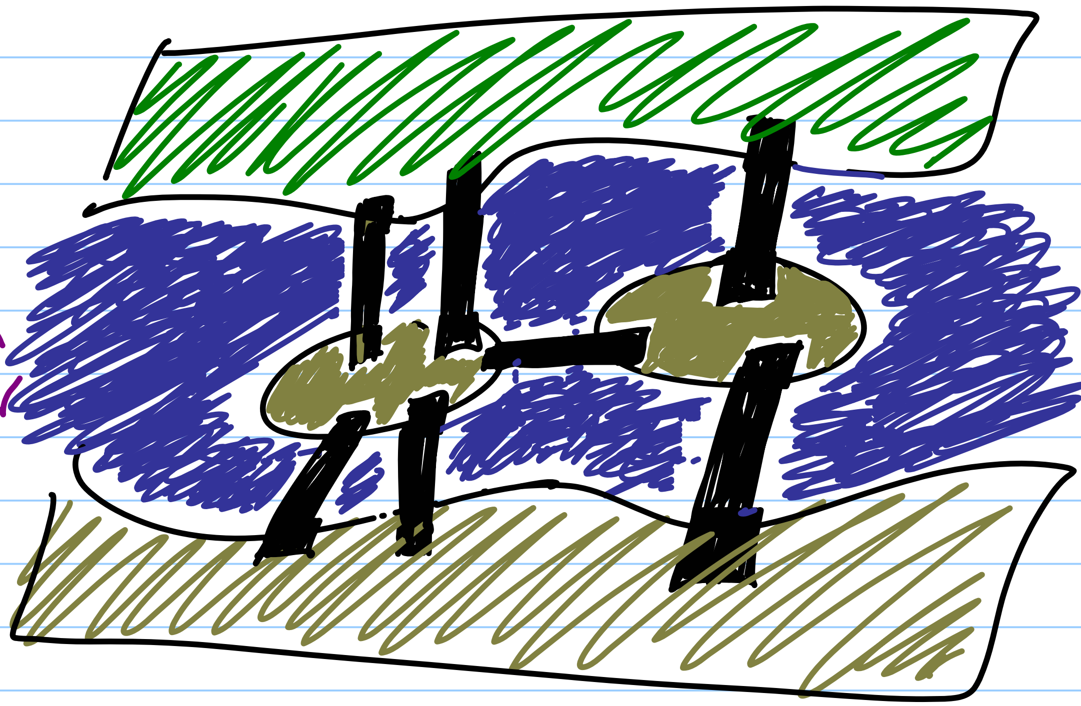
Prove the theorem: $\left[\sum_{v \in V} \text{degree}(v)\right]$ is even.

Another Classic Graph Problem

Königsberg Bridge Problem.

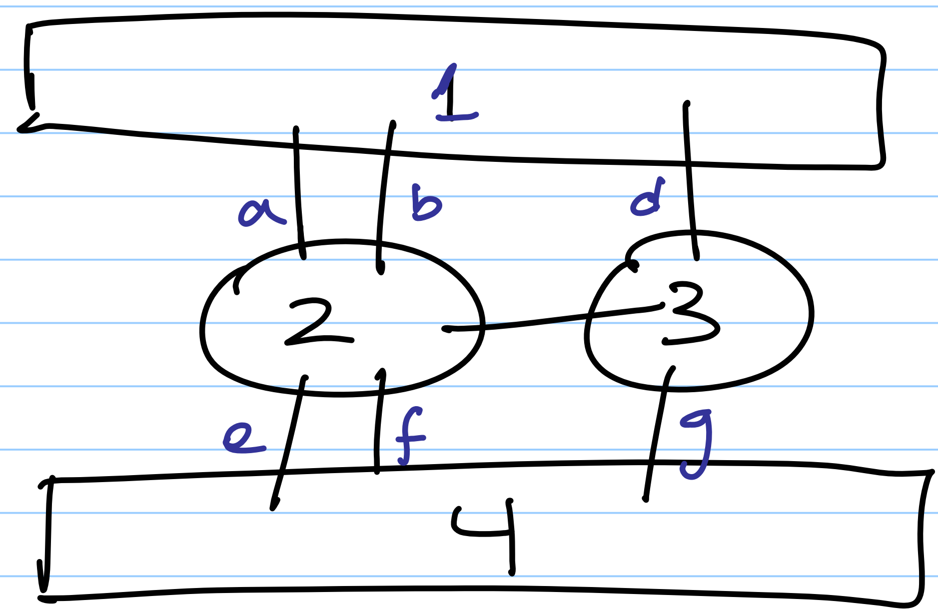
Kaliningrad, Russia

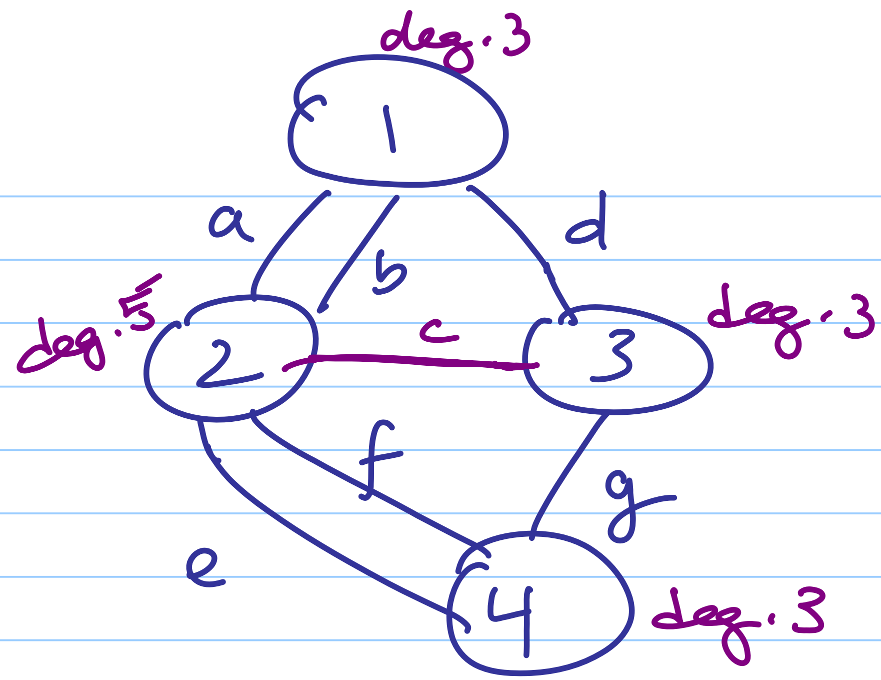
Pregel River



Can you cross all seven bridges without re-using any?

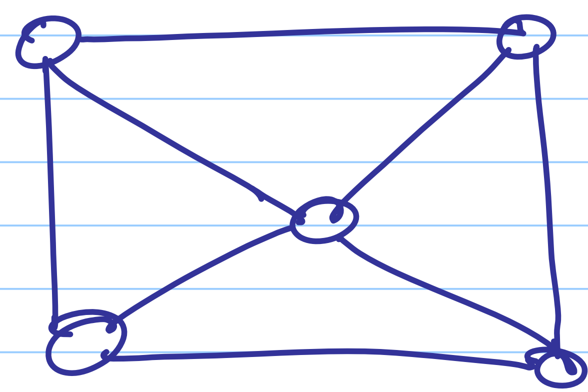
Express as a graph





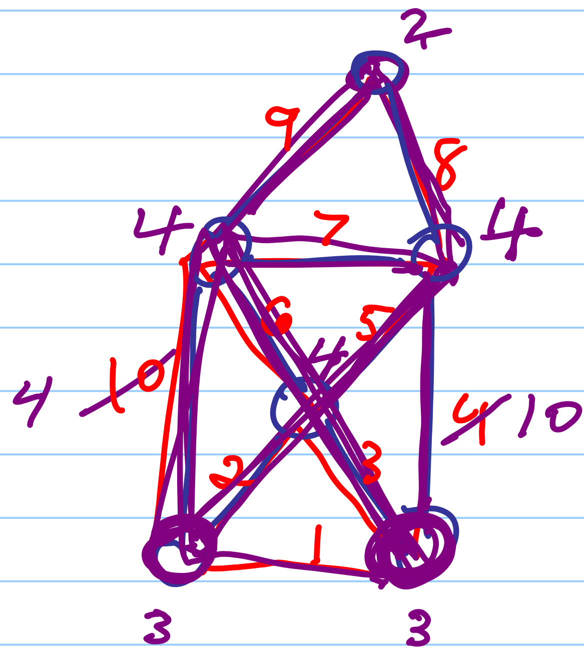
Find the Eulerian Path

What about this? Eulerian path?



no

What about this? Eulerian path?



yes

How to look for an eulerian path

→ start with one edge

try an adjacent edge

keep trying until you reach an edge you've seen before

→ backtrack if this happens

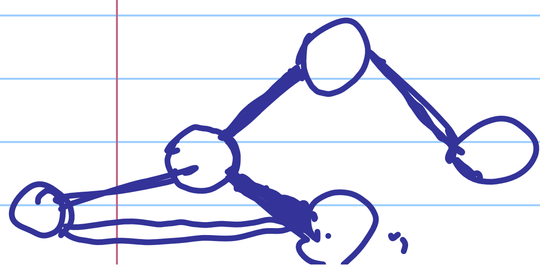
if you reach all nodes, you have an eulerian path!

If you've tried all possible paths, then no eulerian path exist's.

lower bound: you never backtrack!

$\Theta(|E|)$

upper bound : you visit some edges
backtrack
you visit some edges
backtrack
...



worst case → you visit <u>all</u> paths!

worst case # of paths

$$= (|E|)(|E|-1)(|E|-2) \cdots$$

$$= |E|! \in O(|E|^{|E|})$$

$$|E|! \in \Omega(2^{|E|})$$

Say we don't have to know
what exactly the Eulerian path
is.

We just need to know whether one
exists.

What is an algorithm for that?

Euler proved:

if all vertices are of even degree,
then an <u>Eulerian cycle</u> exists.

→ includes every edge exactly once and you return where you start

if there are exactly <u>two</u> vertices of odd degree,
then an <u>Eulerian path</u> exists.

if starts and ends at odd-degree vertices.

Can we find the Eulerian Path in less than $|E|!$ time?

Fleury's Algorithm

1. Start at a vertex with odd degree
2. Choose an edge out of that vertex whose removal will not cut you off from any vertices that still have edges to them.

   reachability $O(|V|+|E|)$

   Traverse and delete the edge.

   done $O(|E|)$ times
3. Repeat. $|E|$ times.

$O(|E||E|(|V|+|E|))$