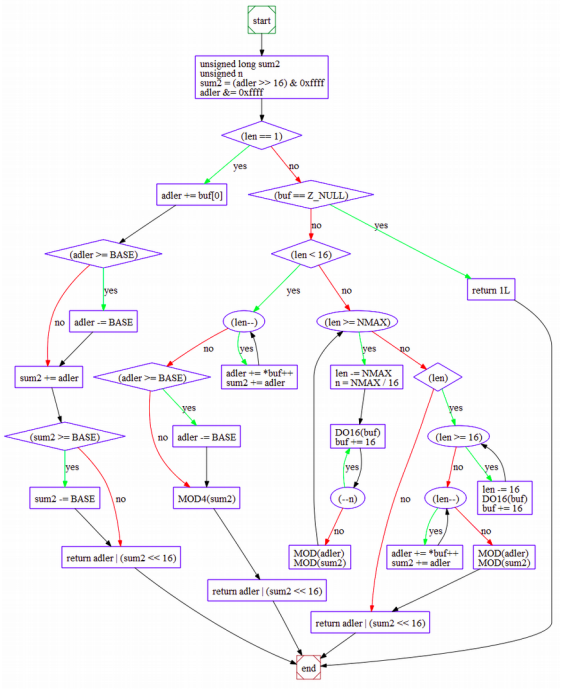
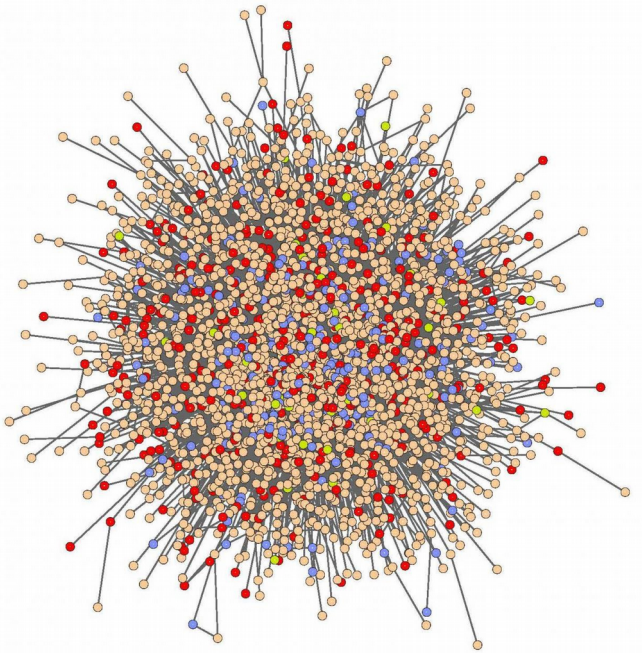
Graphs

Rupesh Nasre.
rupesh@iitm.ac.in

July 2019



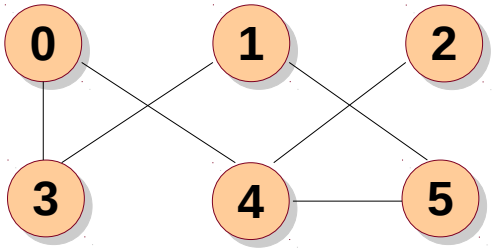
Graphs are Everywhere!



Source: Google images

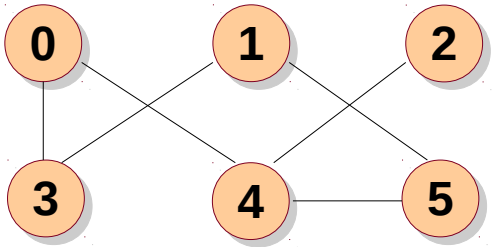
Graphs

- Most general data structure
  - list, stack, queue, tree, ... are special cases.
- Hence, less guarantees, more freedom
- Various types
  - (un)weighted, (un)directed, bipartite, (a)cyclic, ...
- A graph G consists of:
  - A set of nodes / vertices V
  - A set of edges E: V → V



Nomenclature

- Real graph may contain arbitrary data type (such as person name in a social network).
- It is often mapped to a unique integer while representing the graph in memory.
- A vertex may have attributes (such as email id)
- An edge may have attributes (such as road length). It may be weighted.
- An edge may be directed.
  - e.g., twitter's follow, guide-mentee relation.



Relationship

- Graphs are formed due to relationships
  - Guide-mentee relation forms academic geneology
  - Molecular interactions form a biological network
  - Planetary forces form a galactic network
  - Friendship relation forms facebook
  - Follow relation forms twitter
  - In-the-same-iit relation forms alumni network
  - Inherits-from relation forms class hierarchy
  - City-connected-to relation forms a road network
  - ...

Connection with Discrete Maths

- A symmetric relation can be modeled using undirected graphs.
- A transitive relation translates to a path.
- An equivalence relation leads to a connected component.
  - Every vertex has a path to every other (directly or indirectly) related vertex.
- An equivalence relation leads to forest of connected components.
- A poset can be modeled as a DAG (directed acyclic graph).

By Willie Nelson

I was married to a widow, who was pretty as can be

This widow had a grown-up daughter

Who had hair of red

My father fell in love with her, and soon they too were wed

This made my dad my son-in-law

And really changed my life

Now my daughter was my mother

Cause she was my father's wife

And to complicate the matter

Even though it brought me joy

I soon became the father of a bouncing baby boy

My little baby then became a brother-in-law to dad

And so became my uncle, though it made me very sad

For if, if he were my uncle, then that also made him brother

Of the widow's grown up daughter, who was of course, my stepmother

Uh huh

Father's wife then had a son who kept them on the run

And he became my grandchild, for he was my daughter's son

My wife is now my mother's mother, and it makes me blue

Because although she is my wife, she's my grandmother too

God

Now, if my wife is my grandmother, I am her grandchild, yeah

And every time I think of it, heh! Nearly drives me wild

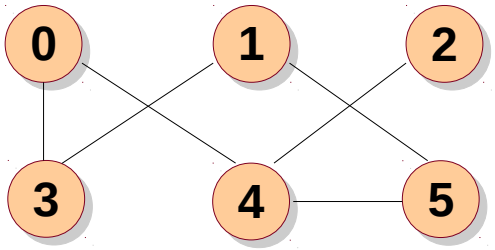
'Cause now I have become the strangest case you ever saw

As husband of my grandmother, I am my own grandpaw

- Republican Chronicle of Ithaca, New York on April 24, 1822

  “There was a widow and her daughter-in-law, and a man and his son. The widow married the son, and the daughter the old man; the widow was, therefore, mother to her husband's father, consequently grandmother to her own husband. They had a son, to whom she was great-grandmother; now, as the son of a great-grandmother must be either a grandfather or great-uncle, this boy was therefore his own grandfather.”

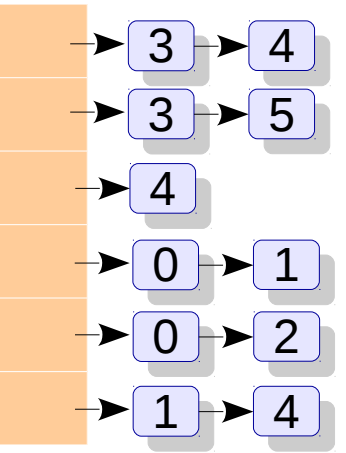
Only graphs can help us model such relationships!

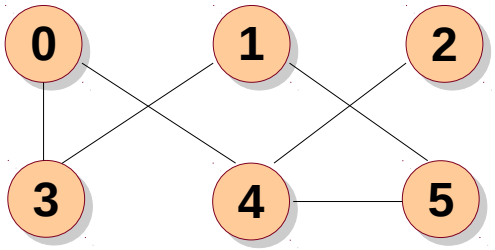


Graph Representation

			1	1	
			1		1
				1	
1	1				
1		1			
	1			1	

1. Adjacency matrix
   - |V| x |V| matrix
   - Each entry [i, j] denotes if edge (i,j) is present in G
   - Useful for **dense** graph
   - Finding neighbors is O(|V|)
2. Adjacency list
   - |V| + |E| size
   - Each vertex i has a list of its neighbors
   - Useful for **sparse** graphs
   - Finding neighbors is O(max. degree)

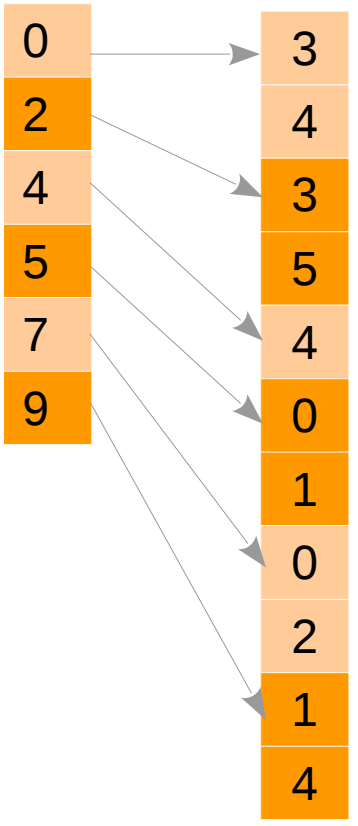




Graph Representation

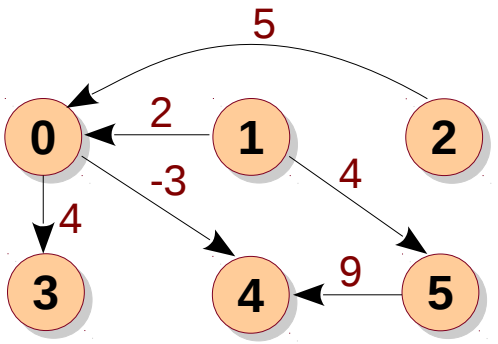
0	3
0	4
1	3
1	5
2	4
3	0
3	1
4	2
5	1
5	4

3. Edge list / Coordinate list (COO)
   - $|\mathbf{E}|$ pairs
   - Useful for edge-based algorithms
   - Typically sorted on vertex id
4. Compressed sparse row (CSR)
   - Concatenated adjacency lists
   - Useful for **sparse** graphs
   - Useful for data transfer



Classwork

- Represent the graph using
  - Adjacency matrix
  - Adjacency list
  - COO format
  - CSR format



Primitive Operations

Classwork: Implement read-only operations in C/C++ for our graph representations.

Classwork: How would you implement dynamic graphs using our graph representations?

- isEdge(e) or hasNeighbor(u, v)
- getDegree(u)
- findNeighbors(u)

Read-only

- updateAttribute(e, attr)
- updateAttribute(u, attr)

Read-Write

- addVertex(u)
- addEdge(u, v)

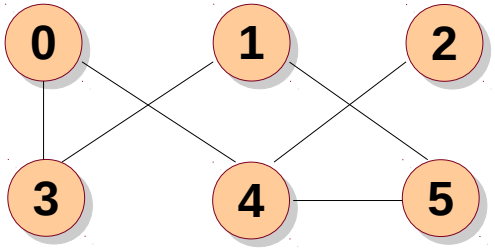
Dynamic graphs

High-level Operations

- Use primitives to:
  - Find if a directed path exists from u to v
  - Find if all the vertices of an undirected graph are connected
  - Find if (u, v, w) form a cycle
  - Check if a graph is bipartite given two partitions of vertices P1 and P2
  - Count all the triangles in the graph

Traversals

- Repeated application of findNeighbors()
- Strategies:
  - Level-by-level: breadth-first search (BFS)
  - Explore a neighbor completely before moving to the other: depth-first search (DFS)
  - Cost based exploration: A* (used in AI / games)
  - ...

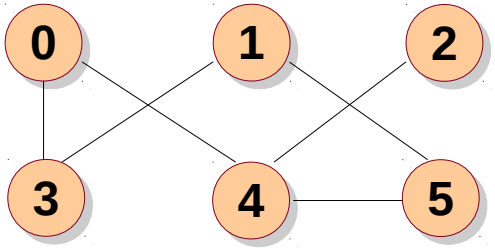


BFS

- Start from a given vertex
  - Then visit each neighbor of the vertex (without exploring the neighbor's neighbors)
  - Then visit the first neighbor's neighbors
  - Followed by second neighbor's neighbors
  - And so on.
- **Classwork**
  - Show the order of BFS traversal from 5 on the example.
  - What edges are present in the BFS?
  - What data structure does it form?
  - What is the complexity of BFS?

BFS Implementation

- **Classwork**: Let's implement BFS.
  - Source: bfs.cpp
  - Implement BFS without a queue. What is the complexity?
- Applications of BFS:
  - Finding connectivity
  - Finding cycles
  - Finding strongly connected components (forward-backward method)
  - Checking bipartiteness
  - ...



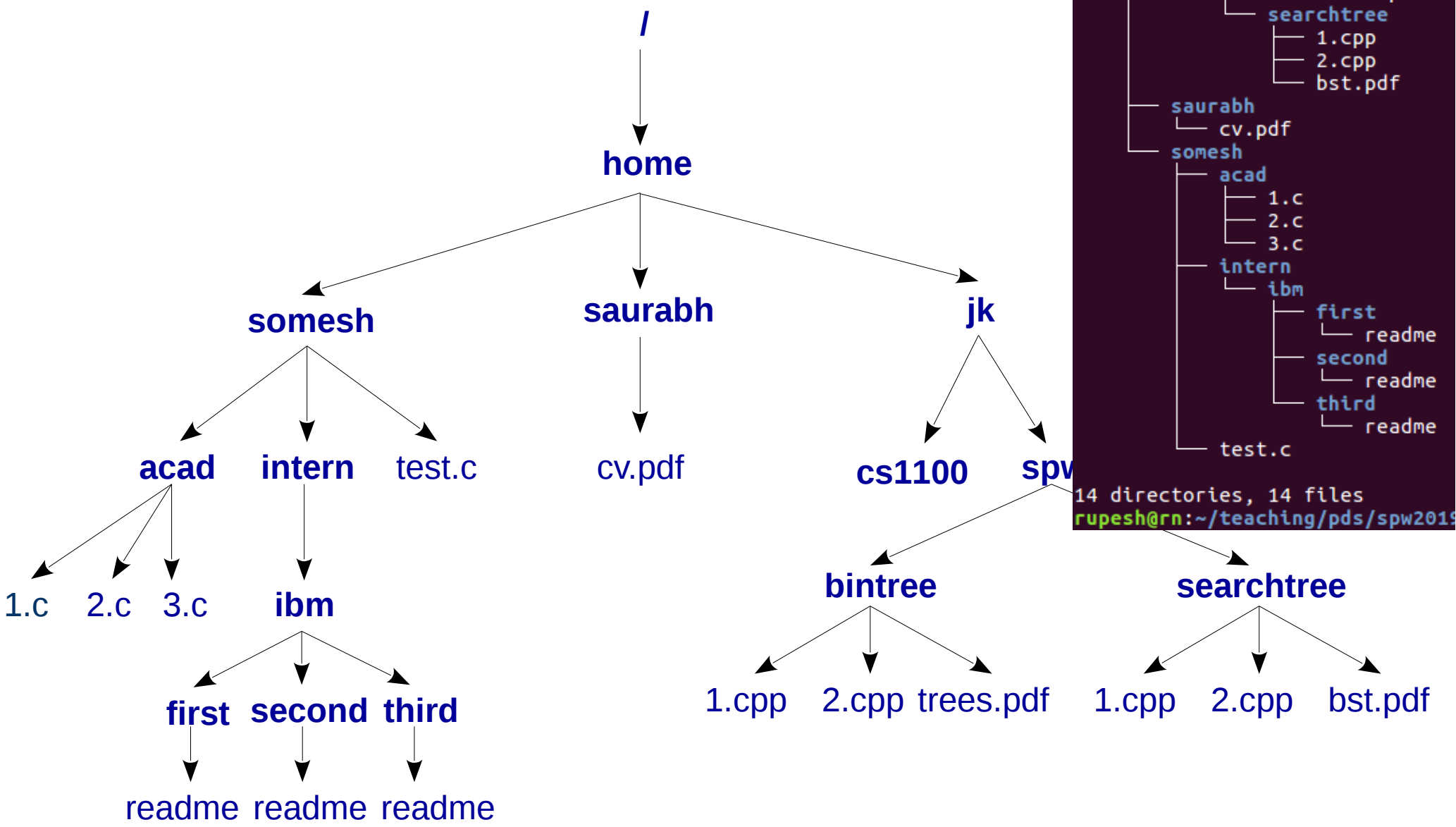
DFS

- Start from a given vertex
  - Then visit the first neighbor of the vertex
  - Then visit the first neighbor's neighbors, without exploring the second neighbor. Go deep.
  - Followed by second neighbor's neighbors. Go deep.
  - And so on.
- **Classwork**
  - Show the order of DFS traversal from node 1.
  - What edges are present in the DFS?
  - What data structure does it form?
  - What is the complexity of DFS?

DFS Implementation

- **Classwork**: Let's implement DFS.
  - Source: dfs.cpp
- Applications of DFS:
  - Finding connectivity
  - Finding cycles
  - Finding strongly connected components (forward-backward method)
  - Checking bipartiteness
  - ...

Directory Listing




```
rupesh@rn:~/teaching/pds/spw2019
.
└── home
    ├── jk
    │   ├── cs1100
    │   └── spw
    │       ├── bintree
    │       │   ├── 1.cpp
    │       │   ├── 2.cpp
    │       │   └── trees.pdf
    │       └── searchtree
    │           ├── 1.cpp
    │           ├── 2.cpp
    │           └── bst.pdf
    ├── saurabh
    │   └── cv.pdf
    └── somesh
        ├── acad
        │   ├── 1.c
        │   ├── 2.c
        │   └── 3.c
        ├── intern
        │   └── ibm
        │       ├── first
        │       │   └── readme
        │       ├── second
        │       │   └── readme
        │       └── third
        │           └── readme
        └── test.c

14 directories, 14 files
rupesh@rn:~/teaching/pds/spw2019
```

Topological Sort

Source: topo.cpp
Uses Kahn's algorithm.

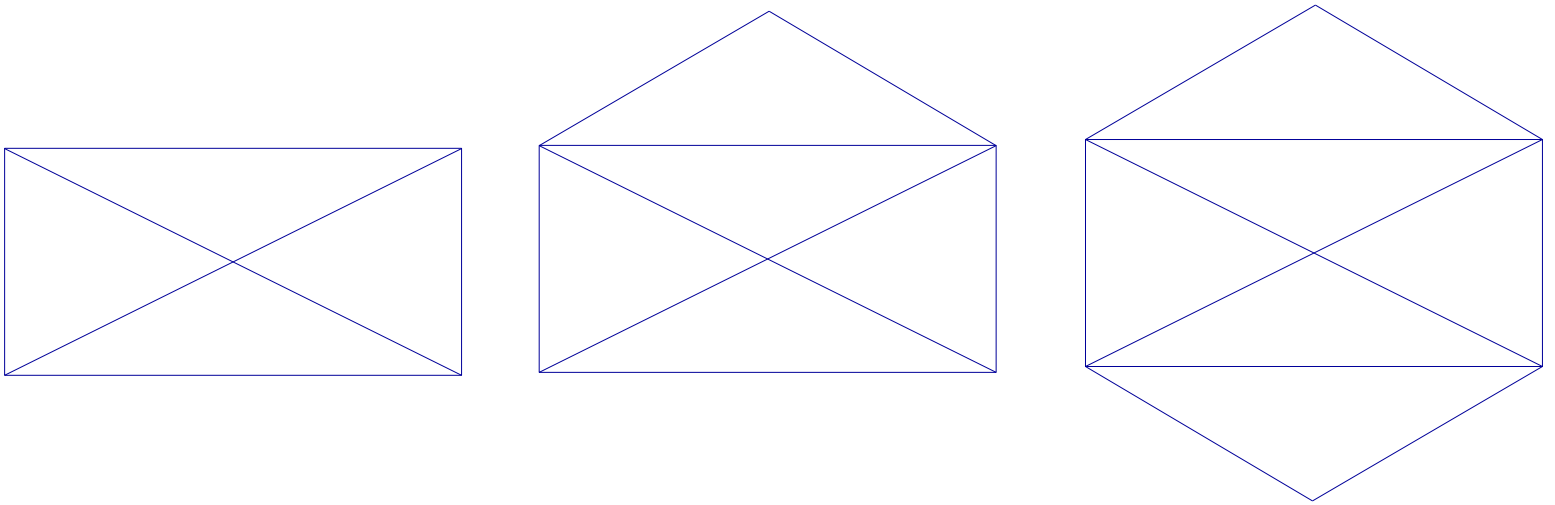
- Given an arbitrary, directed graph,

  collapse cycles.

  This leads to a DAG (directed acyclic graph).
- Each DAG exhibits a topological ordering.
  - All ancestors of a node precede the node.
- This is what we did in the last assignment.
- Applications:
  - OS task dependences
  - Course prerequisites
  - Job scheduling (in companies, check gantt charts)
  - ...

Euler Tour

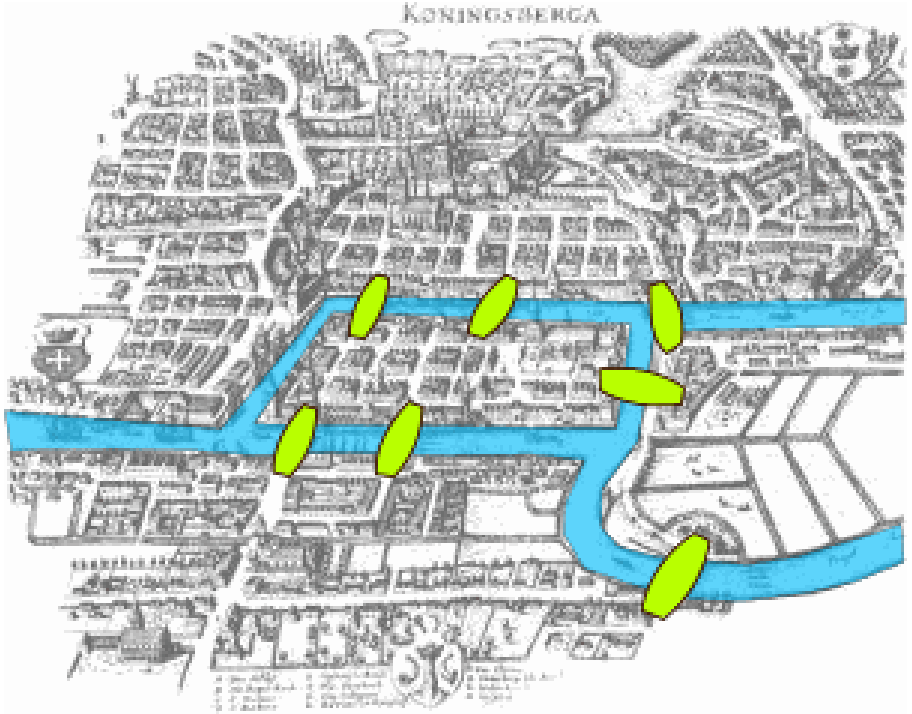
- Draw the following diagrams in your copies without retracing and without lifting the pen.
  - Can you come back to the original point?



- Once we enter a vertex via one edge, we need to leave the vertex via another edge.
  - Hence, each node degree must be even.

Euler Tour

- It is proven that even degree is both necessary and sufficient criterion for Euler tour.
  - Also called Eulerian circuit
  - Considered to be start of Graph Theory



The city of Königsberg in Prussia (now Kaliningrad, Russia)

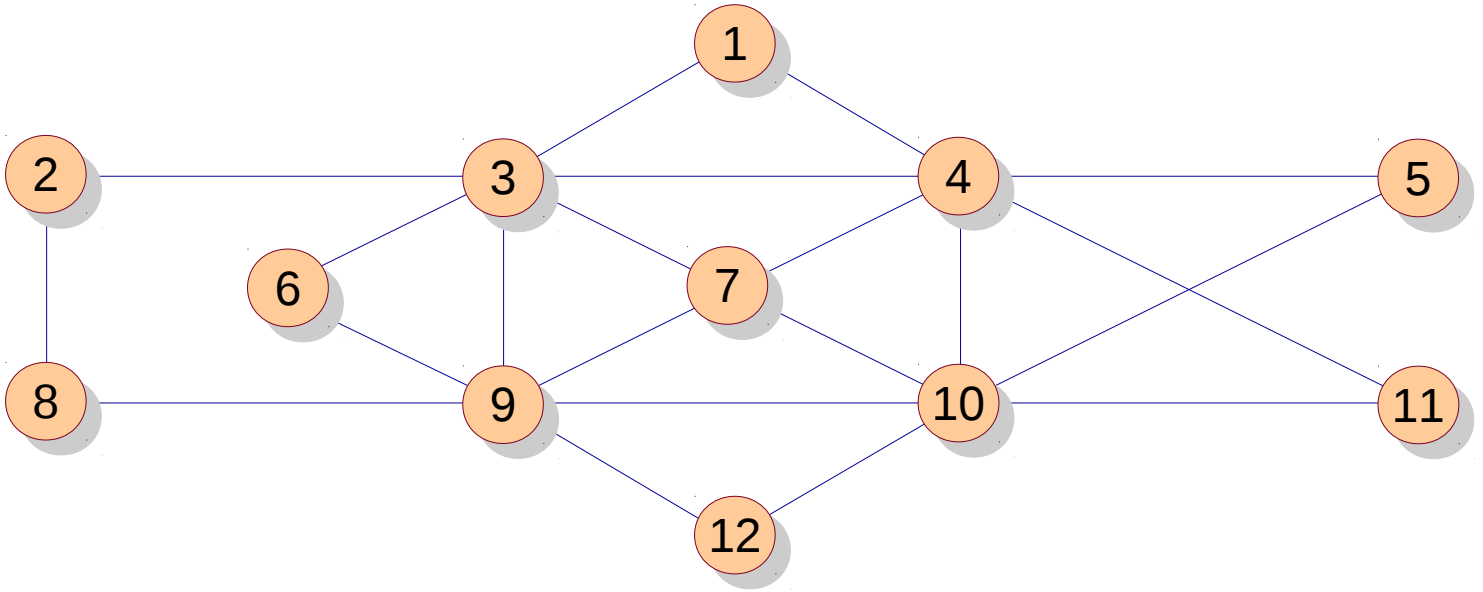
The problem was to devise a walk through the city that would cross each of the seven bridges exactly once.

In 1736, Euler proved that it was not possible.
Considered to be the first theorem of graph theory, and first true proof in the theory of networks.

Currently, there are five bridges (and Euler path is possible, not tour).

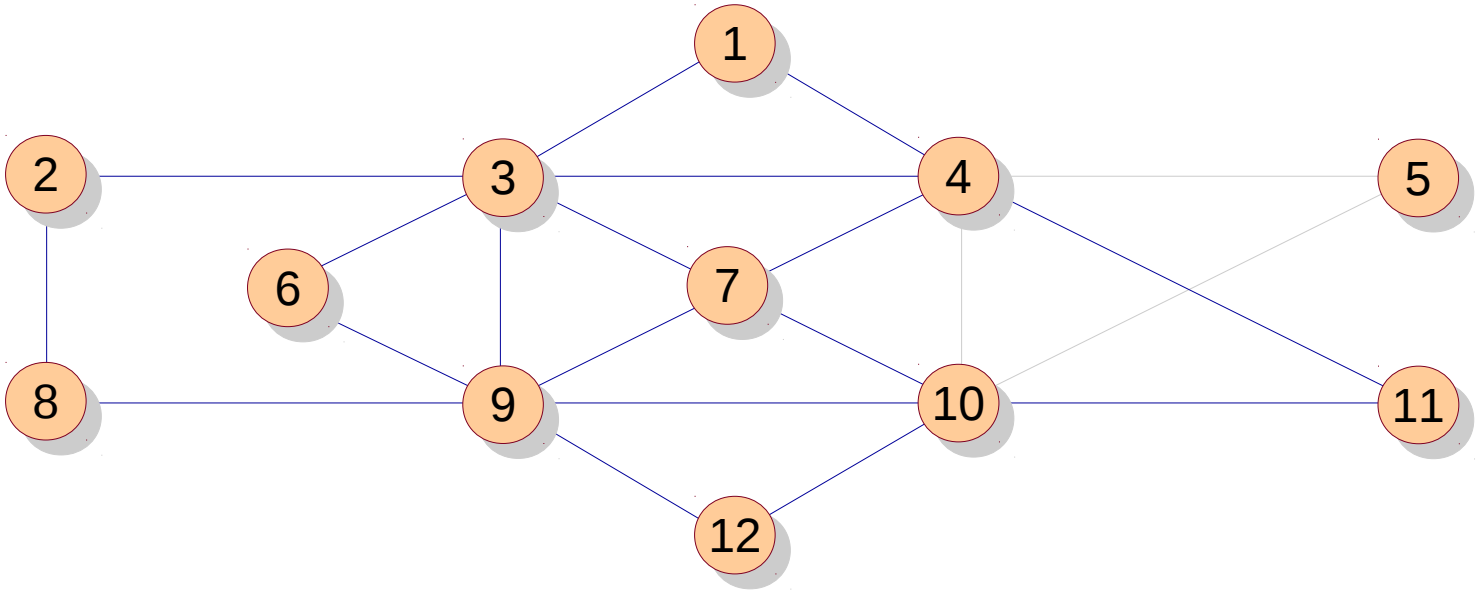
Source: wikipedia

Finding Euler Tour



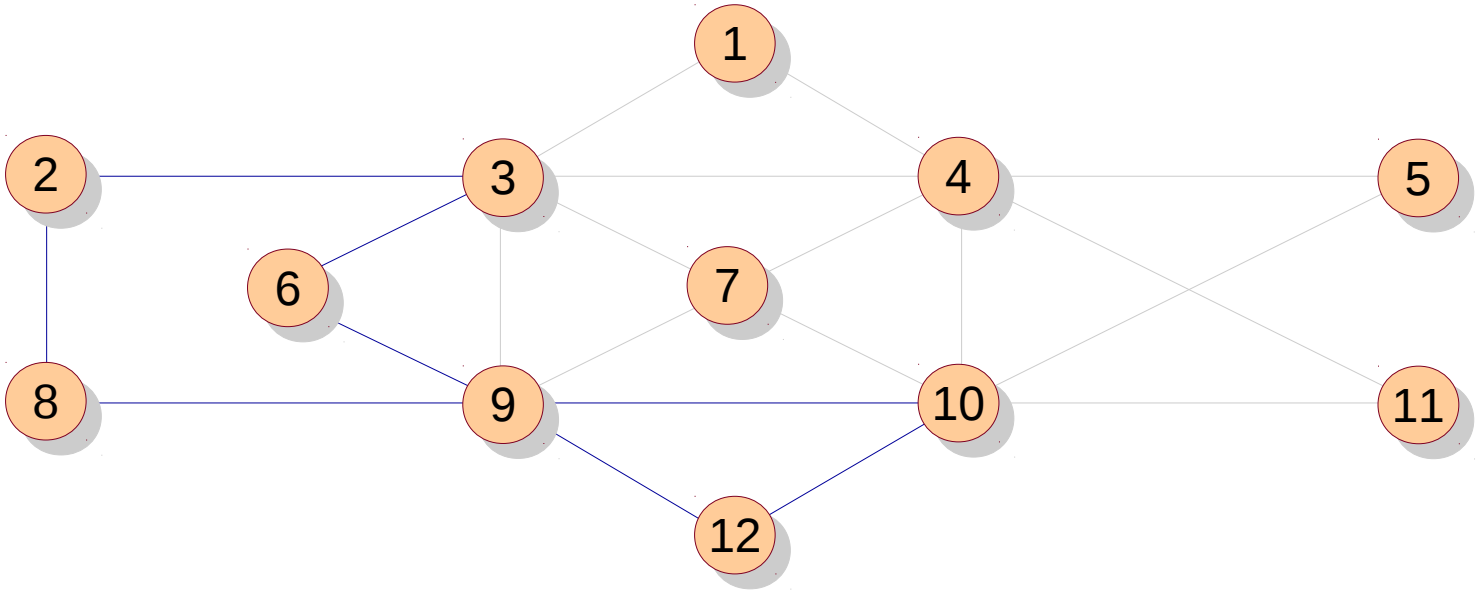
- Find a cycle, remove it. Splice the path.
- Continue with the vertex having unvisited edge in the path.
  - The remaining graph must be Eulerian.

Finding Euler Tour



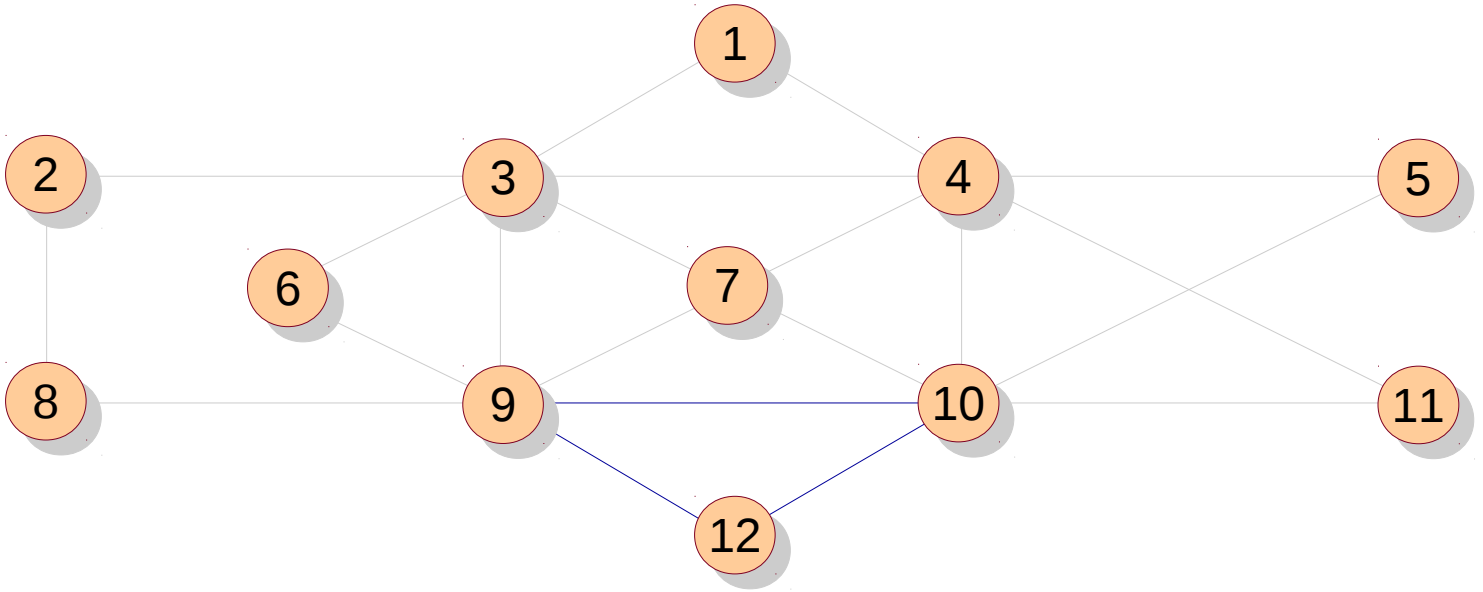
- Cycle: 5 4 10 5
- Path = 5 4 10 5
- First vertex with unvisited edges = 4
- Cycle: 4 1 3 7 4 11 10 7 9 3 4
- Path: 5 4 1 3 7 4 11 10 7 9 3 4 10 5

Finding Euler Tour



- Path: 5 4 1 3 7 4 11 10 7 9 3 4 10 5
- First vertex with unvisited edges = 3
- Cycle: 3 2 8 9 6 3
- Path: 5 4 1 3 2 8 9 6 3 7 4 11 10 7 9 3 4 10 5

Finding Euler Tour



- Path: 5 4 1 3 2 8 9 6 3 7 4 11 10 7 9 3 4 10 5
- First vertex with unvisited edges = 9
- Cycle: 9 12 10 9
- Path: 5 4 1 3 2 8 9 12 10 9 6 3 7 4 11 10 7 9 3 4 10 5
- This is the Euler tour. Runs in O(|E| + |V|) time.

Hamiltonian Cycle

- Find a tour such that each node is visited exactly once, and all the nodes are covered.
  - Have we seen such a problem before?
  - Unlike Euler tour, this problem is hard on general graphs.

Practice Questions

- Given an unweighted graph and two vertices u, v, there could be multiple paths between u and v. What is the shortest distance between them?
- Given a weighted graph with weight values 1 or 2, find the shortest distance between two vertices.
- Find if a node is a part of a 5-clique.
- Find if a node is a part of a 5-cycle.
- Find an edge, deleting which partitions the graph into two disconnected subgraphs, if it exists.
- If A is an adjacency matrix, what does A^k mean?

Some Graph Problems

- Given a node, find another node having the same neighborhood, if one exists.
  - Approximate similarity
  - Bipartite graphs (document-terms, course preferences)
- Find if a graph is uniform or skewed.
  - Road networks, social networks
- Simulate nslookup.
  - Network of routers; store tables.
- Graph coloring

Learning Outcomes

- Represent graphs in memory
- Traverse graphs
- Solve simple graph problems