Graph Theory Introduction

Written by Rachel Singh, last updated Sep 30, 2018

About

Now we’re starting a whole new topic – graph theory. We need to cover some terminology and notation first.

Topics

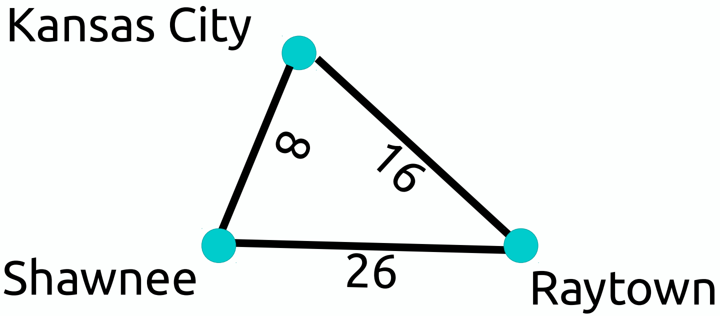
1. Intro to Graphs
2. Graph Terminology
3. Eulerian Graphs

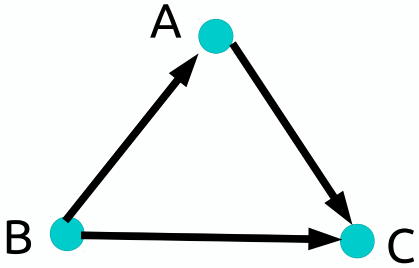
Intro to Graphs

1. Intro to Graphs

Graph Theory is a visual way we can represent relationships between objects.

In a graph, we have **nodes** (aka vertices) that represent some kind of data, and **edges** that join two nodes together. These edges can be directed or undirected. The edges may or may not have a weight associated with them.





Notes

Node/Vertex: A point in the graph, that acts as an end-point between edges.

Edge: A line that connects two vertices (or a single vertex to itself)

1. Intro to Graphs

In Computer Science, data may be represented with graphs as well. There are also Graph-based database systems like *Neo4j*, which is different from a more traditional *relational database system*.

Notes

Node/Vertex: A point in the graph, that acts as an end-point between edges.

Edge: A line that connects two vertices (or a single vertex to itself)

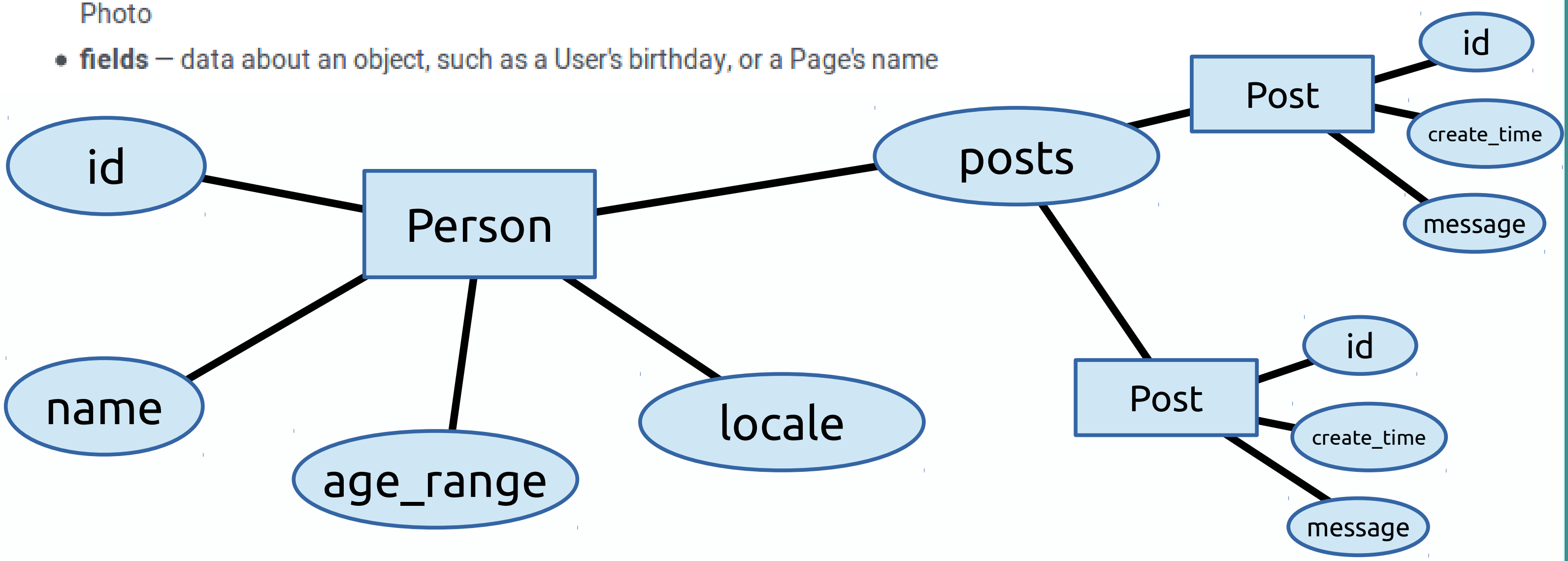
1. Intro to Graphs

Example: Facebook Graph API

https://developers.facebook.com/docs/graph-api/overview/

The Graph API is named after the idea of a "social graph" — a representation of the information on Facebook. It's composed of:

- **nodes** — basically individual objects, such as a User, a Photo, a Page, or a Comment
- **edges** — connections between a collection of objects and a single object, such as Photos on a Page or Comments on a Photo
- **fields** — data about an object, such as a User's birthday, or a Page's name



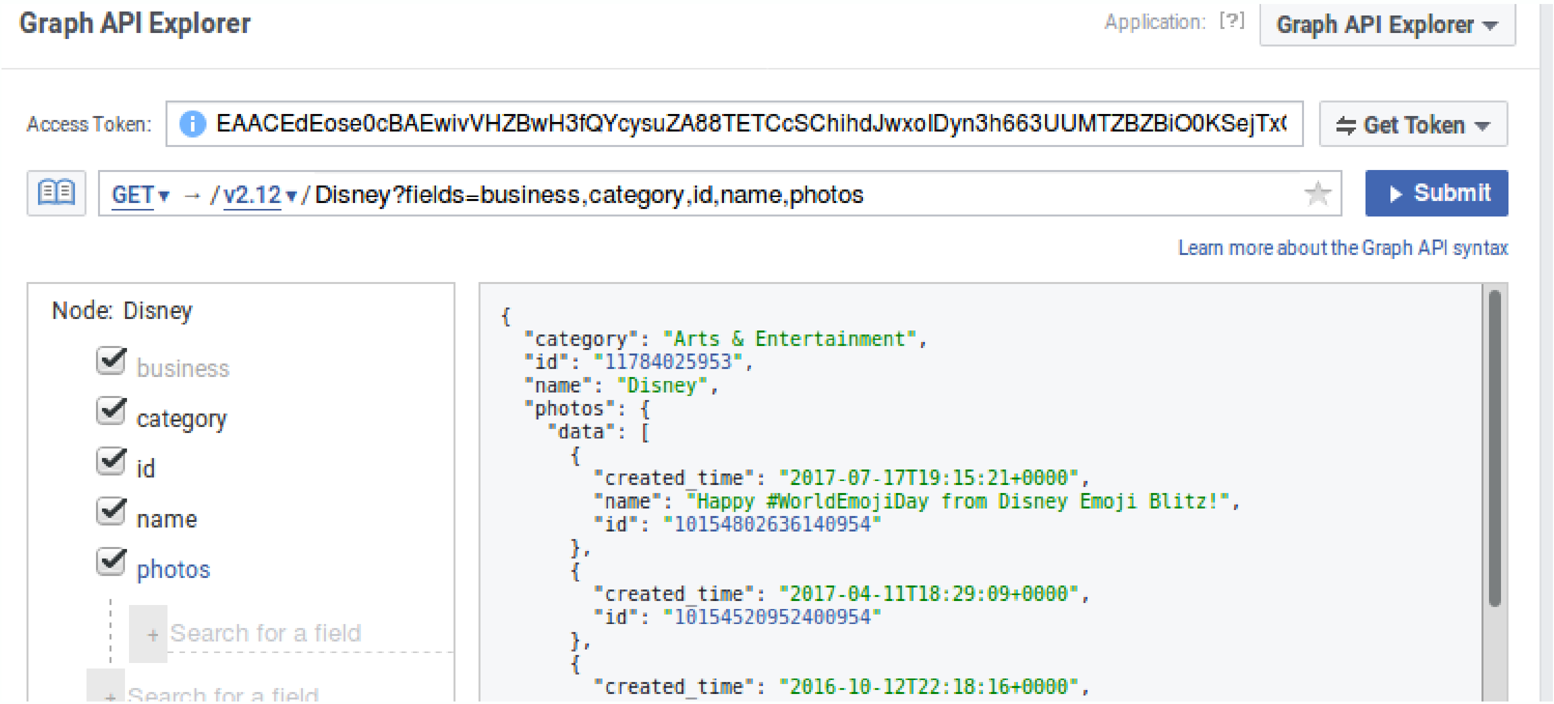
Notes

Node/Vertex: A point in the graph, that acts as an end-point between edges.

Edge: A line that connects two vertices (or a single vertex to itself)

1. Intro to Graphs

Example: Facebook Graph API
Explore the Graph API:
https://developers.facebook.com/tools/explorer/



Notes

Node/Vertex: A point in the graph, that acts as an end-point between edges.

Edge: A line that connects two vertices (or a single vertex to itself)

1. Intro to Graphs

Other uses of Graph Theory in Computer Science:

- Database relationships
- Data flow diagrams
- Representation of computer networks
- Data mining
- Image processing

Information from
http://research.ijcaonline.org/volume104/number1/pxc3899025.pdf

Notes

Node/Vertex: A point in the graph, that acts as an end-point between edges.

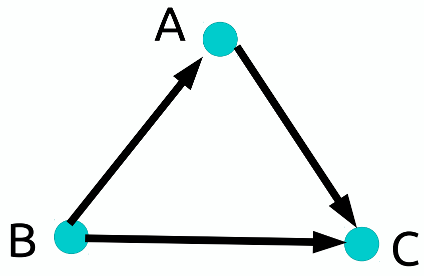
Edge: A line that connects two vertices (or a single vertex to itself)

Graph Terminology

2. Graph Terminology

A graph, at minimum, has **nodes/vertices** and **edges**.

We can think of a graph as having a set V of vertices, and a set E of edges.



V = { A, B, C }

E = { (B, A), (A, C), (B, C) }

B to A, A to C, and B to C

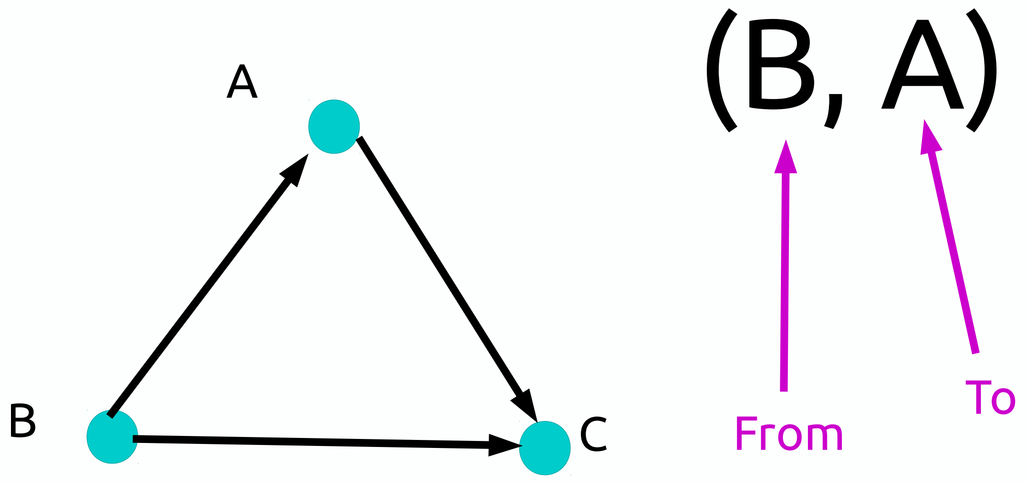
Notes

Node/Vertex: A point in the graph, that acts as an end-point between edges.

Edge: A line that connects two vertices (or a single vertex to itself)

2. Graph Terminology

For a **directed graph**, we use **ordered pairs** to specify the direction between two nodes.



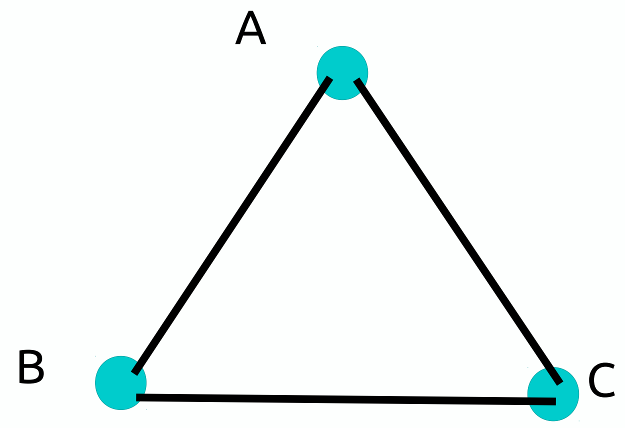
Notes

Node/Vertex: A point in the graph, that acts as an end-point between edges.

Edge: A line that connects two vertices (or a single vertex to itself)

2. Graph Terminology

For an **undirected graph**, the node order doesn't matter, and you should write it in square brackets.



[B, A]

[A, C]

[B, C]

Notes

Node/Vertex: A point in the graph, that acts as an end-point between edges.

Edge: A line that connects two vertices (or a single vertex to itself)

2. Graph Terminology

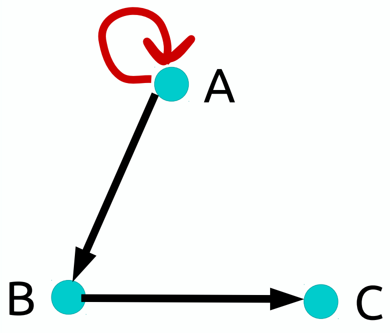
There's a lot of terminology to cover, so I'll try to cover it in an easy-to-look-up way…

Notes

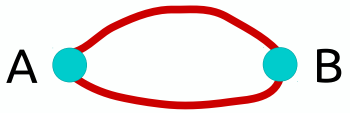
Node/Vertex: A point in the graph, that acts as an end-point between edges.

Edge: A line that connects two vertices (or a single vertex to itself)

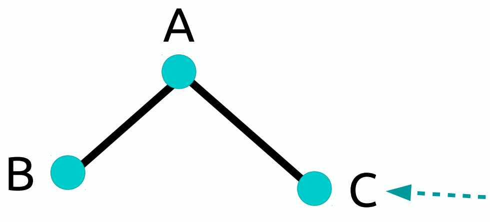
2. Graph Terminology



Loop: A vertex-edge-vertex grouping where the endpoints are the same vertex.



Parallel/Multiple Edges: Two edges that share the same two endpoints.



Adjacent Nodes (vertices): Two nodes that are joined by an edge.

A and B are adjacent, and A and C are adjacent.

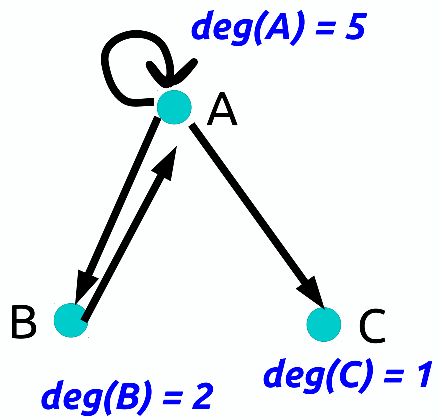
Notes

Loop: An edge where both its endpoints are the same vertex.

Parallel edges: Two edges that have the same two endpoints.

Adjacent nodes: Two nodes that are joined by an edge.

2. Graph Terminology



Degree: The degree of a vertex, *deg(v)*, is the number of times that *v* is an endpoint of an edge. Loops are counted twice.

Notes

Loop: An edge where both its endpoints are the same vertex.

Parallel edges: Two edges that have the same two endpoints.

Adjacent nodes: Two nodes that are joined by an edge.

2. Graph Terminology

Connected: A graph is connected if there is a walk between any two pair of nodes.

Notes

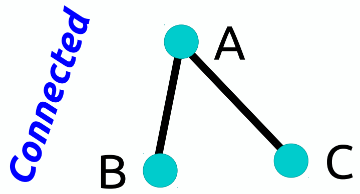
Loop: An edge where both its endpoints are the same vertex.

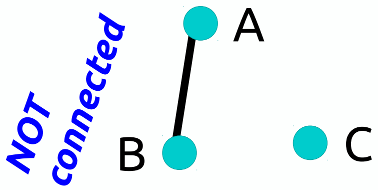
Parallel edges: Two edges that have the same two endpoints.

Adjacent nodes: Two nodes that are joined by an edge.

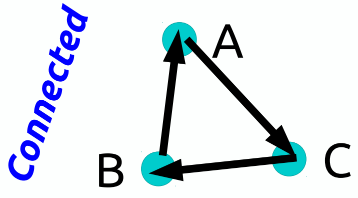
2. Graph Terminology

Connected: A graph is connected if there is a walk between any two pair of nodes.





For a **directed graph**, there must be a walk between any two pair of nodes as well.



Notes

Loop: An edge where both its endpoints are the same vertex.

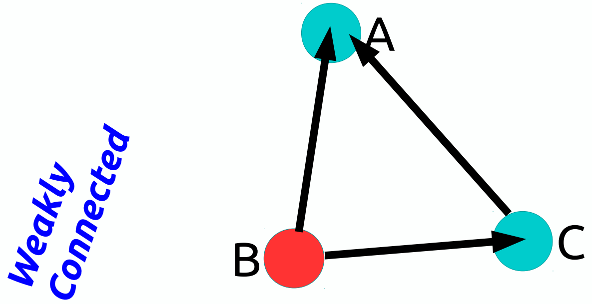
Parallel edges: Two edges that have the same two endpoints.

Adjacent nodes: Two nodes that are joined by an edge.

Connected graph: If there is a walk between any two pairs of nodes.

2. Graph Terminology

A **directed graph** is only **weakly connected** if, after turning all directed edges into undirected edges it is a connected graph.



B only has outgoing edges, so it cannot be reached via a walk.

Notes

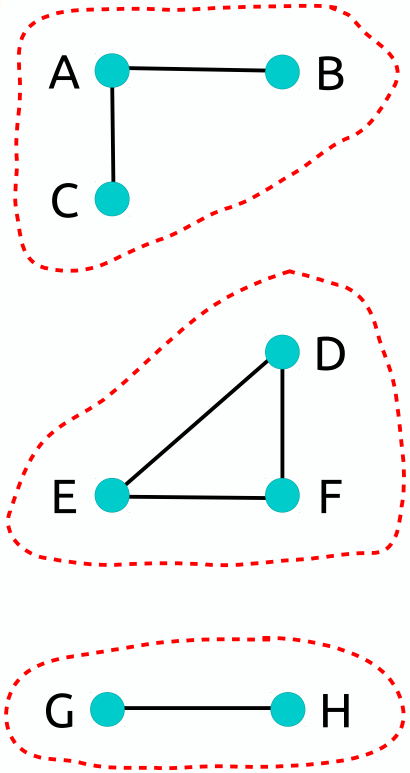
Loop: An edge where both its endpoints are the same vertex.

Parallel edges: Two edges that have the same two endpoints.

Adjacent nodes: Two nodes that are joined by an edge.

Connected graph: If there is a walk between any two pairs of nodes.

2. Graph Terminology



One graph, three connected components

Connected components: The different groupings of connected subgraphs in a full graph.

Notes

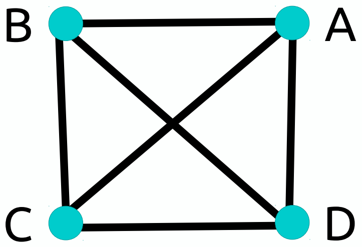
Loop: An edge where both its endpoints are the same vertex.

Parallel edges: Two edges that have the same two endpoints.

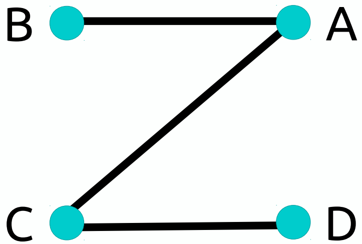
Adjacent nodes: Two nodes that are joined by an edge.

Connected graph: If there is a walk between any two pairs of nodes.

2. Graph Terminology



Graph A



Graph B

Subgraph: A graph B is a subgraph of A if all nodes in B are also in A, and all edges in B are also in A.

You can think of this as, you can build a subgraph B by using only nodes and edges available in the original graph A.

Notes

Loop: An edge where both its endpoints are the same vertex.

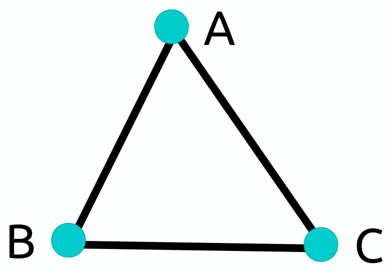
Parallel edges: Two edges that have the same two endpoints.

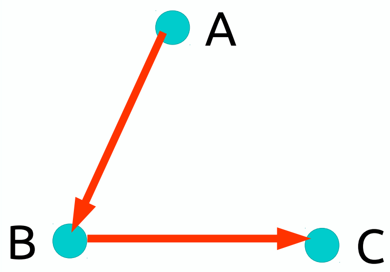
Adjacent nodes: Two nodes that are joined by an edge.

Connected graph: If there is a walk between any two pairs of nodes.

Subgraph: You can build a subgraph using the building blocks of another graph – and no additional items.

2. Graph Terminology





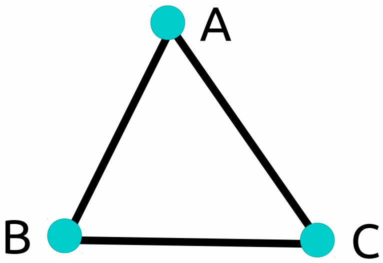
Walk: A sequence of alternating vertices/edges, beginning and ending with vertices.

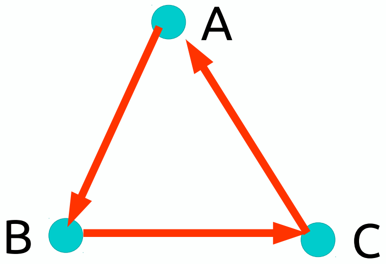
For this graph, a walk could be
A → B → C

Notes

Walk: A series of alternating vertices/edges.

2. Graph Terminology





Walk: A sequence of alternating vertices/edges, beginning and ending with vertices.

Closed walk: A walk that begins and ends at the same vertex.

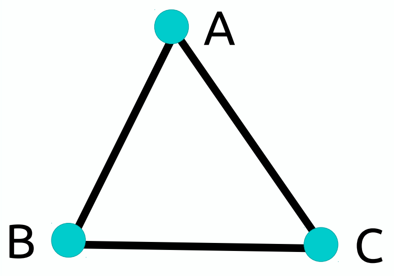
The example here is
A → B → C → A

Notes

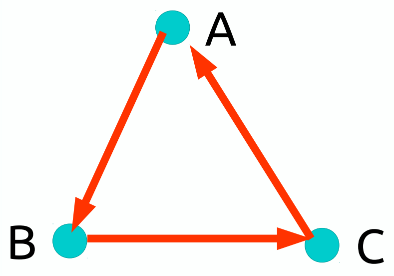
Walk: A series of alternating vertices/edges.

Closed walk: A walk that begins & ends at the same vertex.

2. Graph Terminology



Walk: A sequence of alternating vertices/edges, beginning and ending with vertices.



Closed walk: A walk that begins and ends at the same vertex.

Length of a walk: The amount of edges in the walk.

For A → B → C → A, the length is 3.

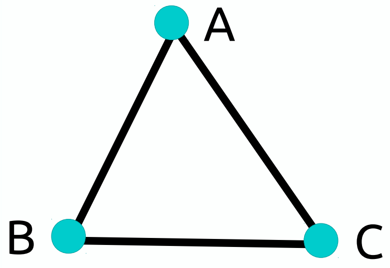
Notes

Walk: A series of alternating vertices/edges.

Closed walk: A walk that begins & ends at the same vertex.

Walk length: The amount of edges in a walk.

2. Graph Terminology



Walk: A sequence of alternating vertices/edges, beginning and ending with vertices.

Trivial: A walk of length 0 (no edges) is a trivial walk.

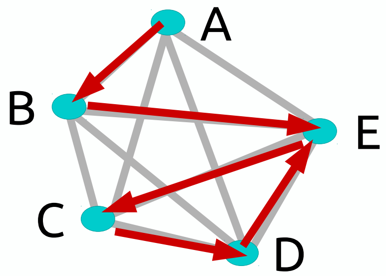
Notes

Walk: A series of alternating vertices/edges.

Closed walk: A walk that begins & ends at the same vertex.

Walk length: The amount of edges in a walk.

2. Graph Terminology



Trail: A trail is a walk with no repeated edges.

For example:

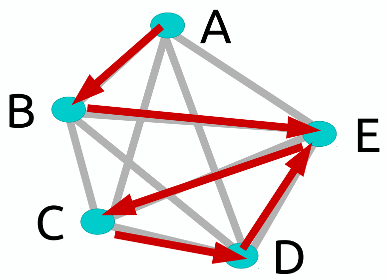
A → B → E → C → D → E

Notes

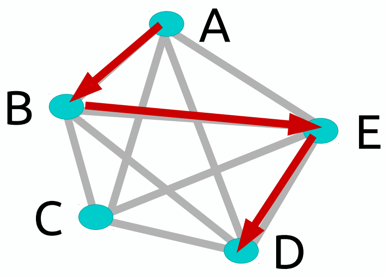
Walk: A series of alternating vertices/edges.

Trail: A walk with no repeated edges.

2. Graph Terminology



Trail: A trail is a walk with no repeated edges.



Path: A walk with no repeated vertices (and therefore no repeated edges).

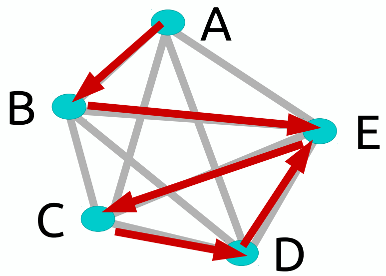
Notes

Walk: A series of alternating vertices/edges.

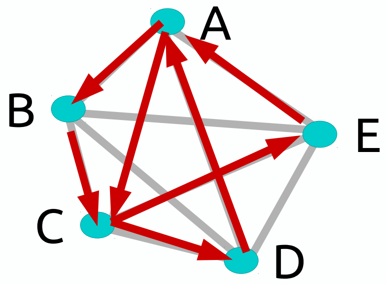
Trail: A walk with no repeated edges.

Path: A walk with no repeated vertices.

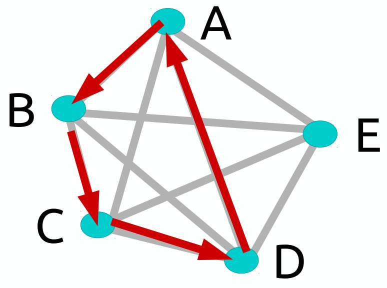
2. Graph Terminology



Trail: A trail is a walk with no repeated edges.



Circuit: A closed trail – the walk begins and ends at the same vertex.



Cycle: A nontrivial circuit where the only repeated node is the begin/end.

Notes

Walk: A series of alternating vertices/edges.

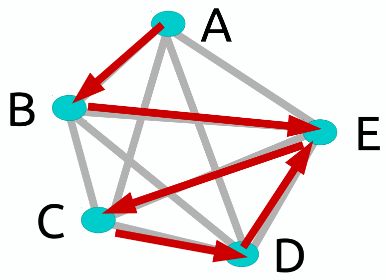
Trail: A walk with no repeated edges.

Path: A walk with no repeated vertices.

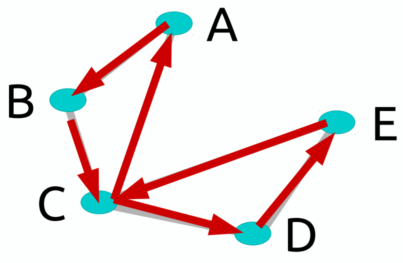
Circuit: A closed trail.

Cycle: A nontrivial circuit where the only repeated nodes are the first/last ones.

2. Graph Terminology



Trail: A trail is a walk with no repeated edges.



Eulerian Trail: A trail where every edge is traversed.

Note: This graph is not the same as above; I had to change it to access all edges.

Notes

Walk: A series of alternating vertices/edges.

Trail: A walk with no repeated edges.

Path: A walk with no repeated vertices.

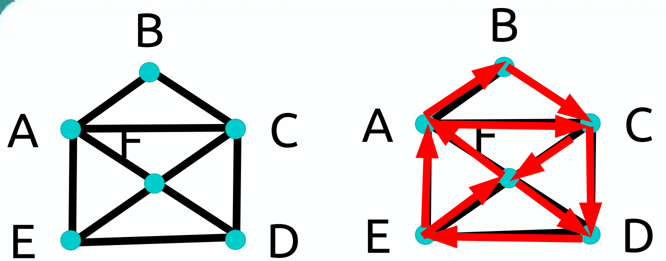
Circuit: A closed trail.

Cycle: A nontrivial circuit where the only repeated nodes are the first/last ones.

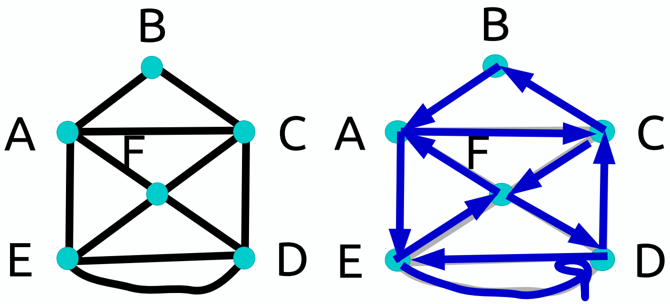
Eulerian Trail: A trail where every edge is traversed.

Eulerian Graphs

3. Eulerian Graphs



Eulerian Trail: A trail where every edge is traversed exactly once. Doesn't matter where we begin/end.



Eulerian Circuit: A circuit where every edge is traversed exactly once. We must begin and end at the same vertex.

Notes

Walk: A series of alternating vertices/edges.

Trail: A walk with no repeated edges.

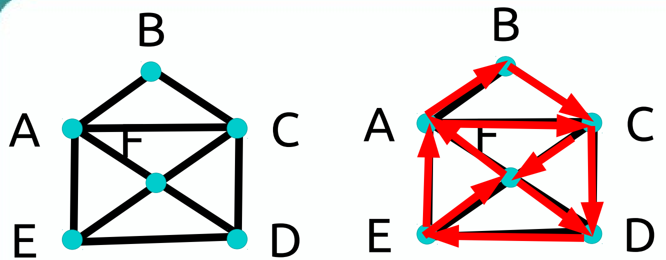
Path: A walk with no repeated vertices.

Circuit: A closed trail.

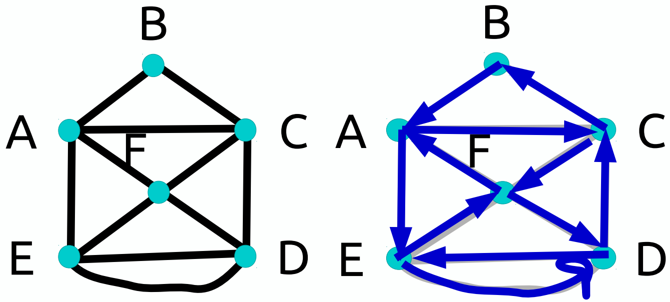
Cycle: A nontrivial circuit where the only repeated nodes are the first/last ones.

Eulerian Trail: A trail where every edge is traversed.

3. Eulerian Graphs



Non-Eulerian Graph with a Eulerian Trail



Eulerian Graph with a Eulerian Circuit

Eulerian Graph: A graph is Eulerian if it contains a <u>Eulerian Circuit</u> – that is, a circuit that traverses each edge exactly once, and starts and ends at the same vertex.

Note: A graph can be Non-Eulerian and contain a Eulerian Trail.

Notes

Walk: A series of alternating vertices/edges.

Trail: A walk with no repeated edges.

Path: A walk with no repeated vertices.

Circuit: A closed trail.

Cycle: A nontrivial circuit where the only repeated nodes are the first/last ones.

Eulerian Trail: A trail where every edge is traversed.

Conclusion

Make sure you keep a reference of the different terminology as you’re working through these concepts.

Next time we will cover more terminology and look at trees.