Week 3

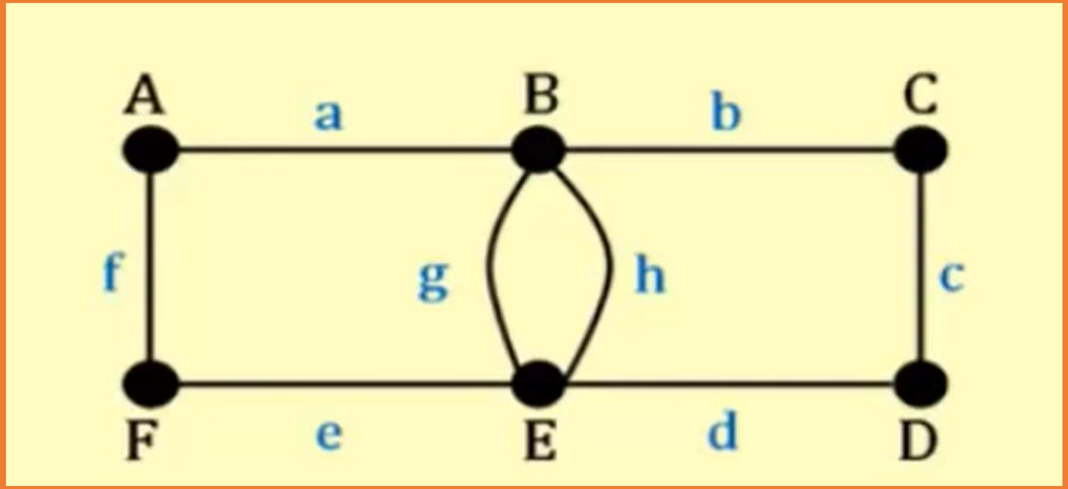
Graph Theory

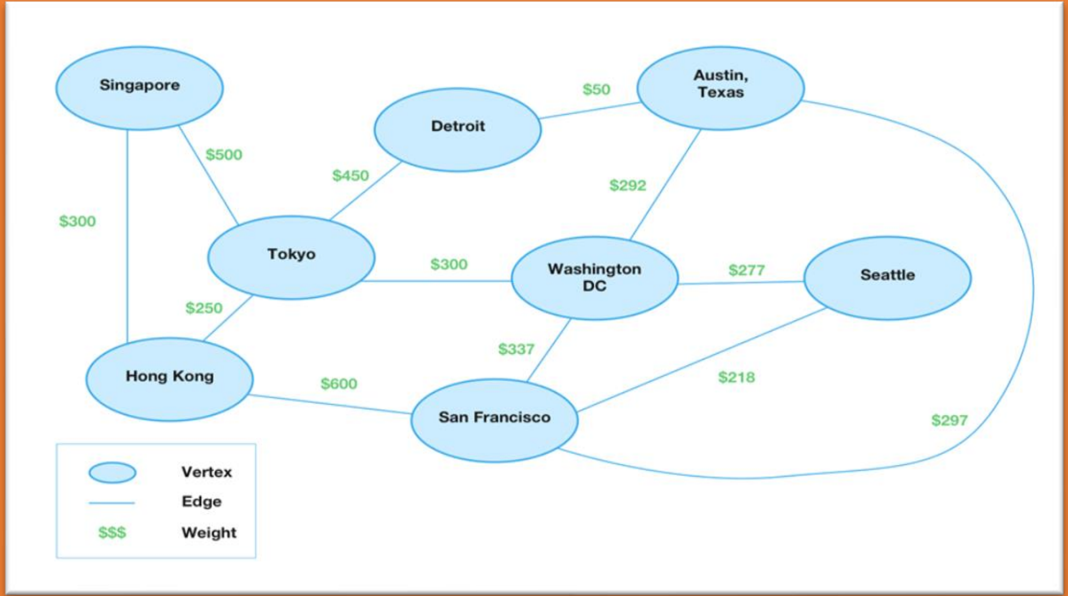
MA123: Mathematical Reasoning & Modeling
(Spring 2021)

Instructor: Emmanuel Thompson

Graph, Vertices and Edges

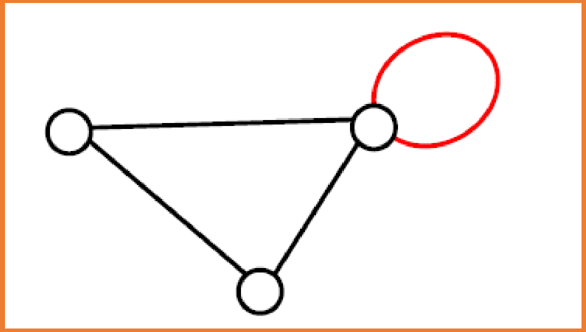
- Graph:
  - Consists of a set of dots, called **vertices**, and a set of **edges** connecting pairs of vertices.
- Vertex:
  - Is a dot in the graph.
  - It could represent an intersection of streets, a land mass, or a general location, like "work" or "school".
  - Vertices are often connected by edges.
- Edge:
  - Connect pairs of vertices (lower-case letters).
  - It can represent a connection between locations, like a street, or
  - Simply that a route connecting the two locations exists, like an airline flight.



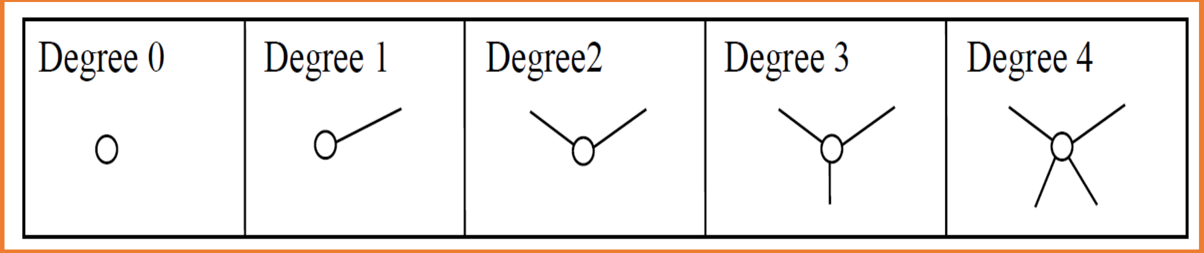


Loop, Degree of a Vertex, Path, Circuit, Connected, Weight

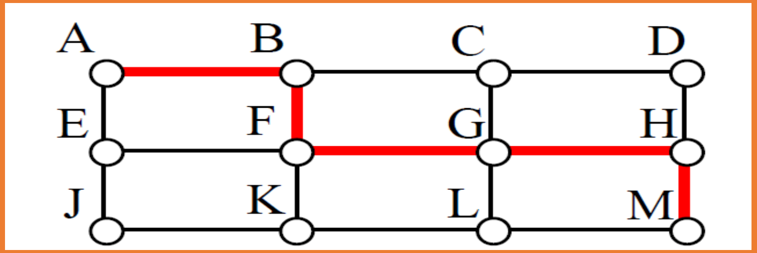
- Loop:
  - Is a special type of edge that connects a vertex to itself.



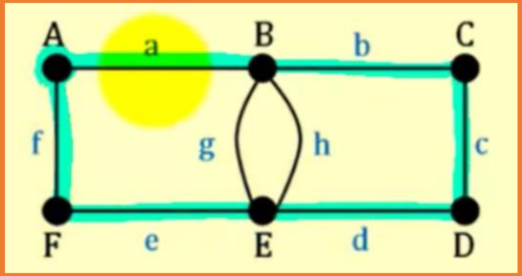
- Degree of a Vetex:
  - Is the **number** of **edges** meeting at that **vertex**.



- Path:
  - Is a sequence of vertices using the edges. Often, the interest is a path between two vertices.
  - Path from vertex A to vertex M (this is only one possibility).



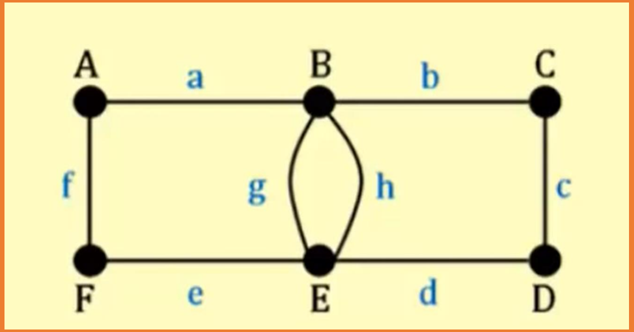
- Circuit:
- Is a path that begins and ends at the same vertex.



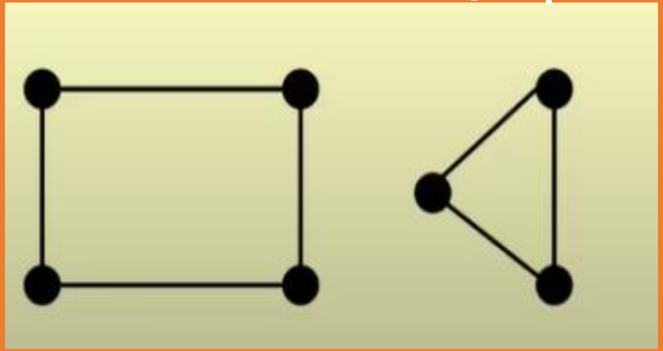
Loop, Degree of a Vertex, Path, Circuit, Connected, Weight

- Connected:
  - A graph is connected if there is a path from any vertex to any other vertex.

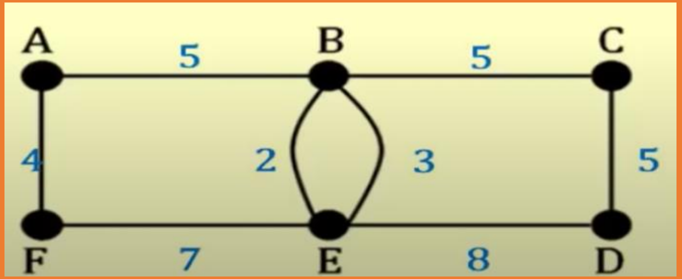
Connected

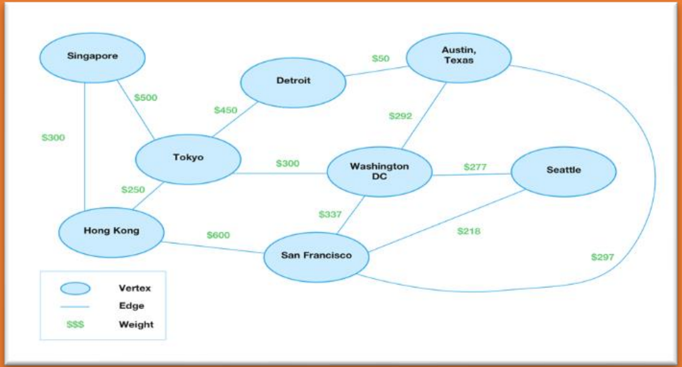


Not Connected



- Weight:
  - Sometimes weights are assigned to the edges.
  - Weight could be the distance between two locations, travel time, travel cost etc.

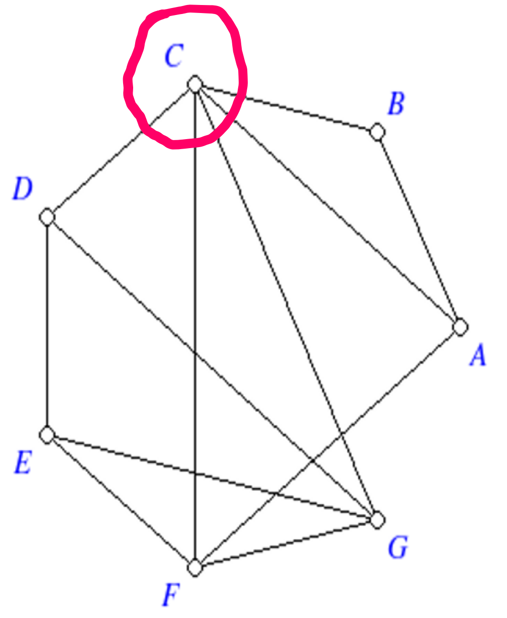




Loop, Degree of a Vertex, Path, Circuit, Connected, Weight

Try This

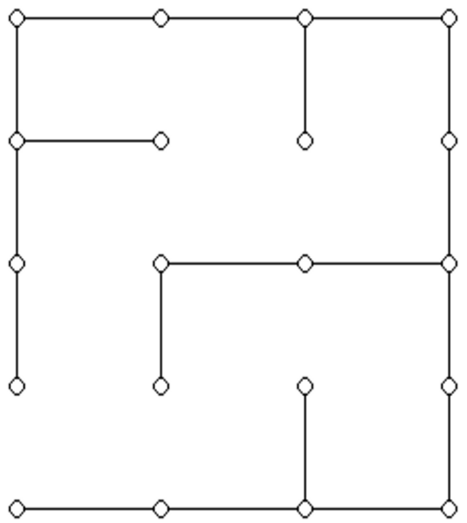
1. What is the degree of vertex C



5

Try This

2. Is this graph connected?



__Not Connected

[✓ __Connected]

Dijkstra's Algorithm (Finding the Shortest Path)

Dijkstra's Algorithm

1) Mark the ending vertex with a distance of zero. Designate this vertex as current.

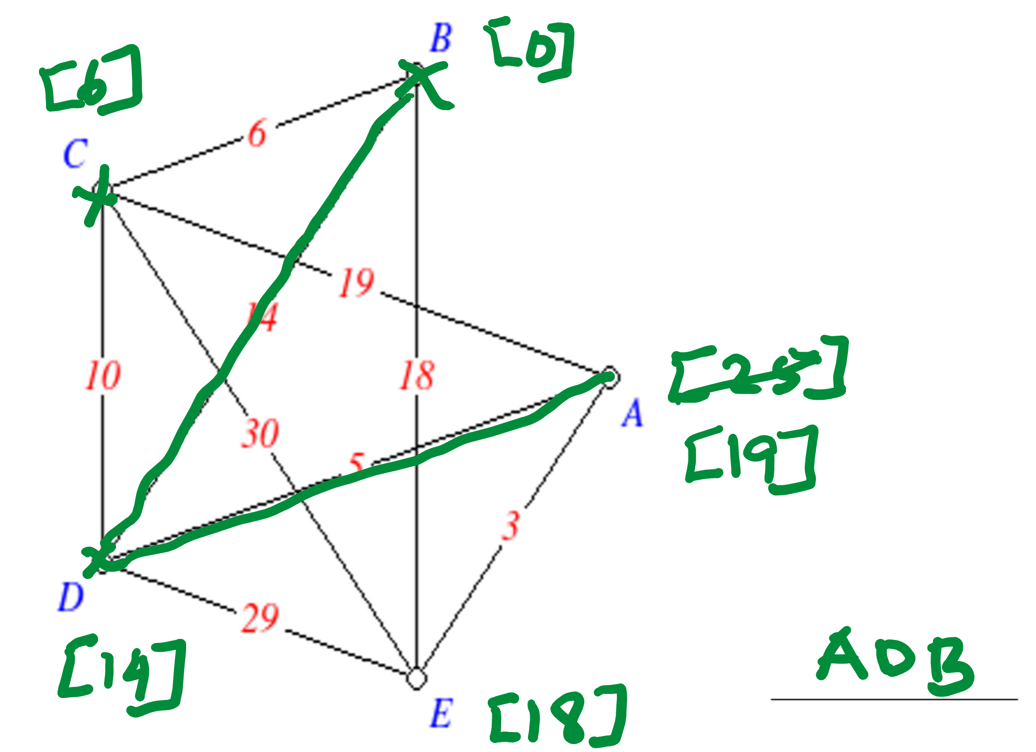
2) Find all vertices leading to the current vertex. Calculate their distances to the end. Since we already know the distance the current vertex is from the end, this will just require adding the most recent edge. Don't record this distance if it is longer than a previously recorded distance.

3) Mark the current vertex as visited. We will never look at this vertex again.

4) Mark the vertex with the smallest distance as current and repeat from step 2.

Try This

Find the shortest path from vertex A to vertex B. Give your answer as a sequence of vertexes, like ACB



ADB

Dijkstra's Algorithm (Finding the Shortest Path)

Dijkstra's Algorithm

1) Mark the ending vertex with a distance of zero. Designate this vertex as current.

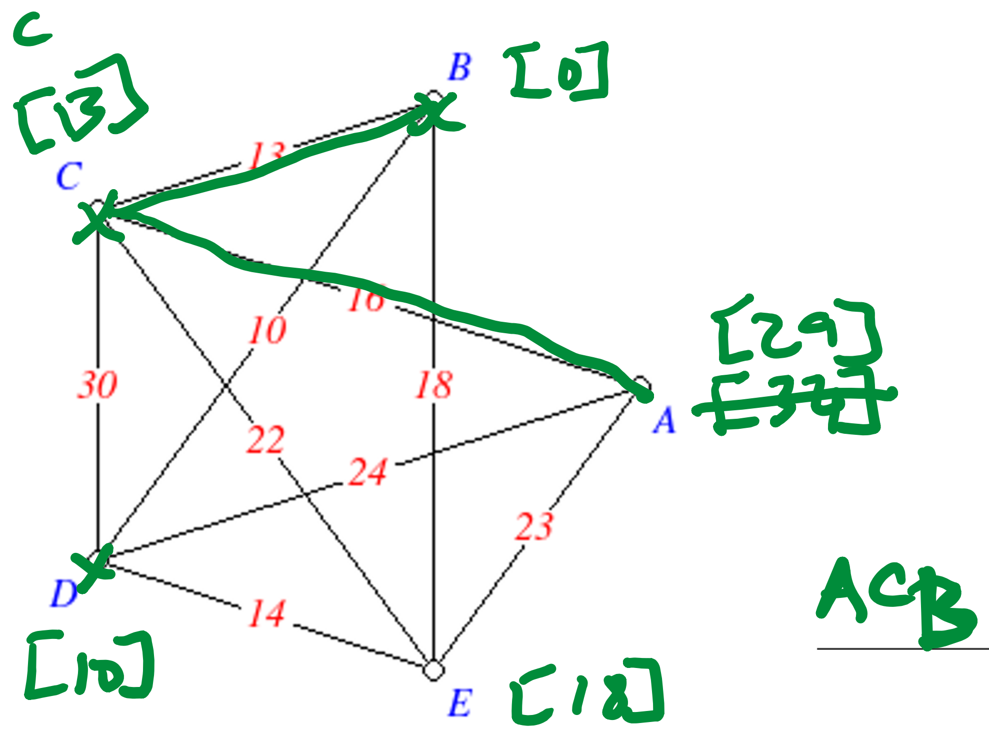
2) Find all vertices leading to the current vertex. Calculate their distances to the end. Since we already know the distance the current vertex is from the end, this will just require adding the most recent edge. Don't record this distance if it is longer than a previously recorded distance.

3) Mark the current vertex as visited. We will never look at this vertex again.

4) Mark the vertex with the smallest distance as current and repeat from step 2.

Try This

Find the length of the shortest path from vertex A to vertex B.



ACB

The length of the shortest path is 29

Dijkstra's Algorithm (Finding the Shortest Path)

Dijkstra's Algorithm

1) Mark the ending vertex with a distance of zero. Designate this vertex as current.

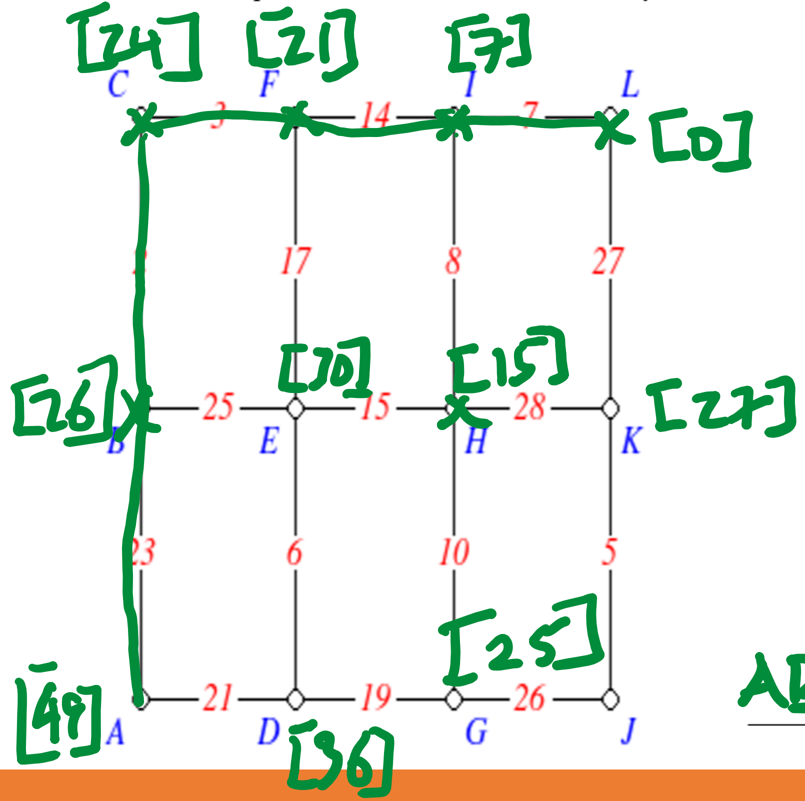
2) Find all vertices leading to the current vertex. Calculate their distances to the end. Since we already know the distance the current vertex is from the end, this will just require adding the most recent edge. Don't record this distance if it is longer than a previously recorded distance.

3) Mark the current vertex as visited. We will never look at this vertex again.

4) Mark the vertex with the smallest distance as current and repeat from step 2.

Try This

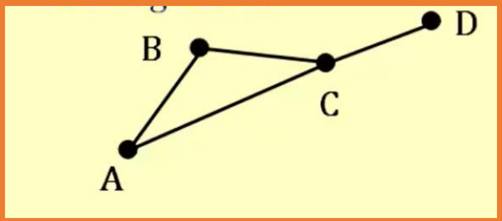
Find the shortest path from vertex A to vertex L. Give your answer as a sequence of vertexes, like ABCFIL

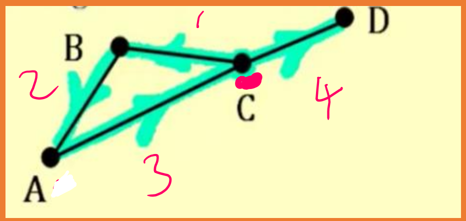


ABCFIL

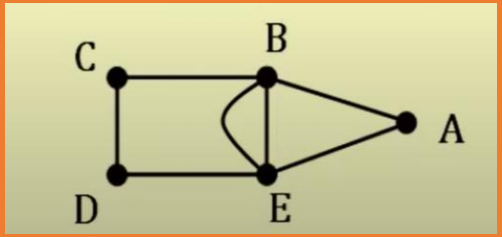
Euler Circuits and Paths

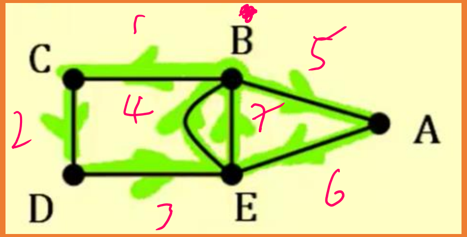
- Euler Path:
  - Path that uses every edge in a graph with no reaptes.
    - Being a path, it does not have to return to the starting vertex.





- Euler Circuit:
  - Circuit that uses every edge in a graph with no repeats.
    - Being a circuit, it must start and end at the same vertex.

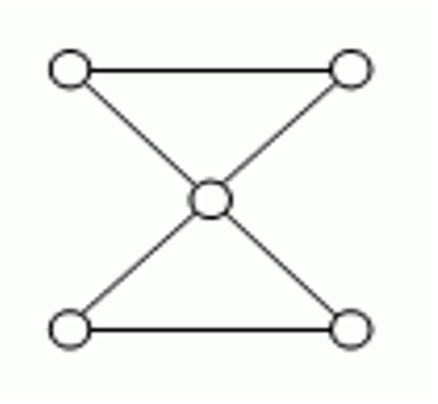




- Euler's Path and Circuit Theorems
  - A graph will contain an Euler path if it contains at most two vertices of odd degree.
  - A graph will contan an Euler circuit if all vertices have even degree.
- Fleury's Algorithm
  1. Start at any vertex if finding an Euler circuit. If finding an Euler path, start at one of the two vertices with odd degree.
  2. Choose any edge leaving your current vertex, provided deleting that edge will not separate the graph into two disconnected sets of edges.
  3. Add the edge to your circuit and delete it from your graph.
  4. Continue until you are done.

Euler Circuits and Paths

Try This

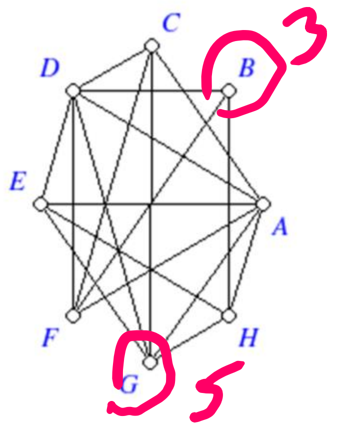


__This graph does not have an Euler Circuit

✓ __This graph has an Euler Circuit

All vertices are of even degree

Determine whether the graph contains an Euler path or an Euler circuit. *Select the one best response.*



At most 2 vertices are odd degree!

✓ __The graph contains at least one Euler path, but no Euler circuit.

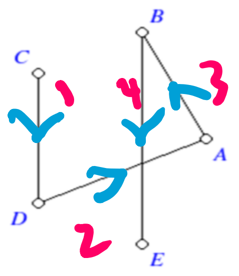
B of degree 3
G of degree 5

__The graph contains at least one Euler circuit (which is also an Euler path).

__The graph does not contain any Euler paths nor Euler circuits.

Try This

Find an Euler path for the graph. Enter your response as a sequence of vertices in the order they are visited, for example, ABCDEA.



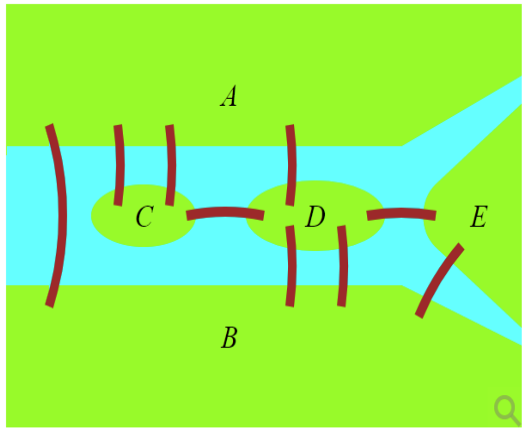
Start with vertex of odd degree

CDABE

Euler Circuits and Paths

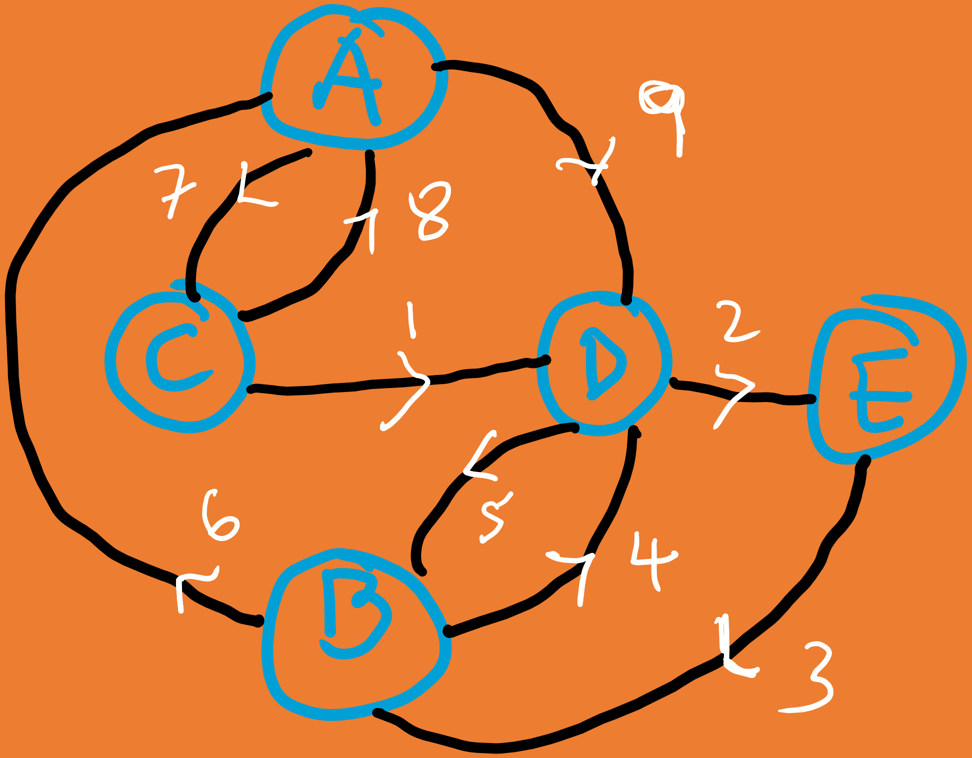
Try This

A city is built on the banks of a river and some islands in the river. The map below shows the bridges connecting the various land masses. Draw a graph that models the connecting relationships in the map below. The vertices represent the land masses and the edges represent bridges connecting them.



Is it possible to find a path through the city that uses each bridge once? If so, enter the sequence of land masses(vertices) visited, for example ABDEA. If it is not possible, enter DNE.

City = Vertex
Bridge = Edge

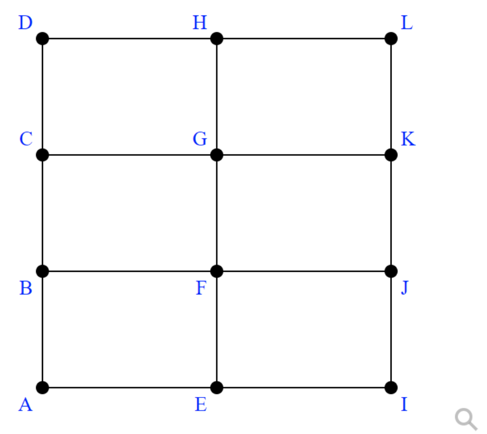


CDEBDBACAD

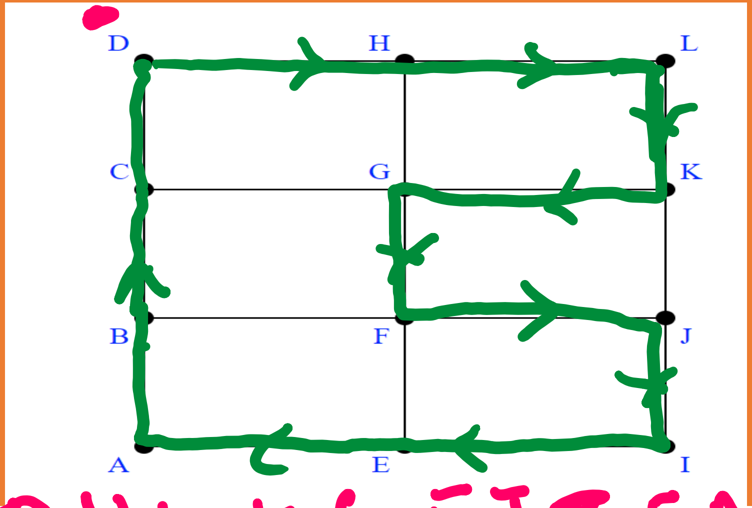
Hamiltonian Circuits and Paths

- Hamiltonian Circuit:
  - Circuit that vists every vertex once with no repeats. Being a circuit, it must start and end at the same vertex.
- Hamiltonian Path
  - Path that visits every vertex once with no repeats but does not have to start and end at the same vertex.
- Brute Force Algorithm (a.k.a Exhaustive Search)
  1. List all possible Hamiltonian circuits.
  2. Find the length of each circuit by adding the edge weights.
  3. Select the circuit with minimal total weight.

Try This



Find any Hamiltonian circuit on the graph above. Give your answer as a list of vertices, starting and ending at the same vertex. Example: ABCA



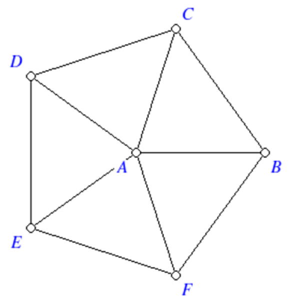
D H L K G F J I E A B C D

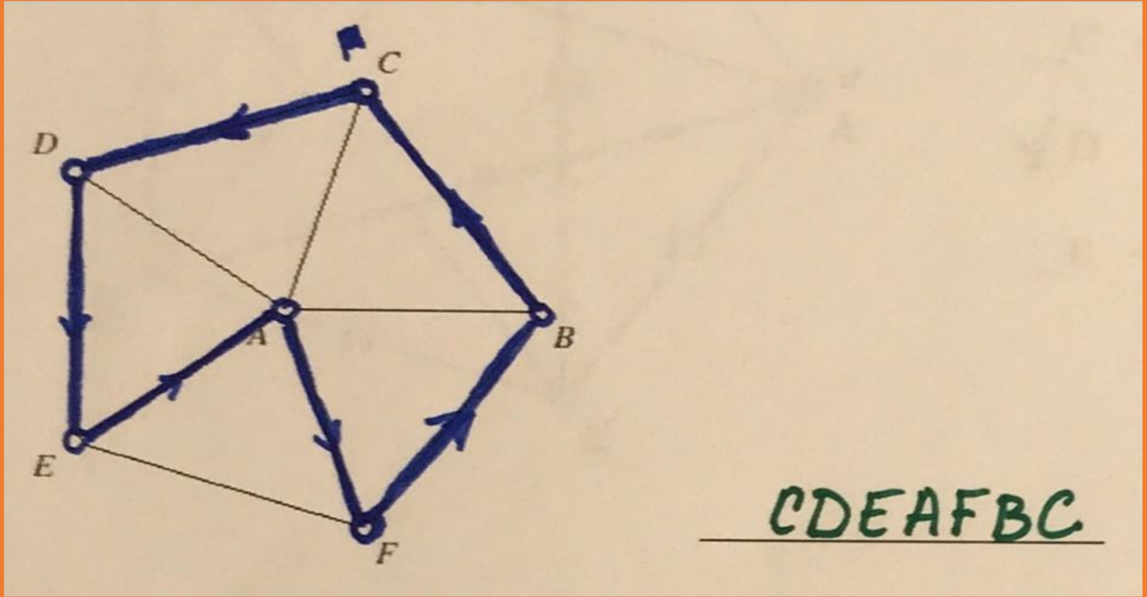
Hamiltonian Circuits and Paths

- Hamiltonian Circuit:
  - Circuit that vists every vertex once with no repeats. Being a circuit, it must start and end at the same vertex.
- Hamiltonian Path
  - Path that visits every vertex once with no repeats but does not have to start and end at the same vertex.
- Brute Force Algorithm (a.k.a Exhaustive Search)
  - List all possible Hamiltonian circuits.
  - Find the length of each circuit by adding the edge weights.
  - Select the circuit with minimal total weight.

Try This

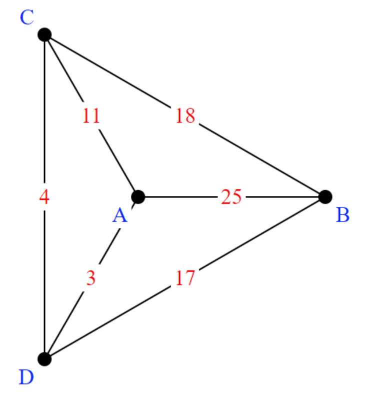
Find any Hamiltonian circuit on the graph above. Give your answer as a list of vertices, starting and ending at the same vertex. Example: ABCA



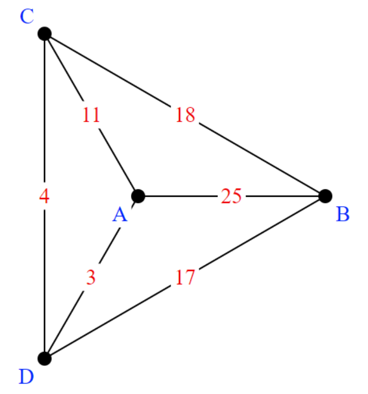


Nearest Neighbors Algorithm (NNA)

- NNA
  1. Select a starting point.
  2. Move to the nearest unvisted vertex (the edge with smallest weight).
  3. Repeat until the circuit is complete.



Apply the nearest neighbor algorithm to the graph above starting at vertex A. Give your answer as a list of vertices, starting and ending at vertex A. Example: ABCDA



ADC BA

$A \to D \to C \to B \to A$

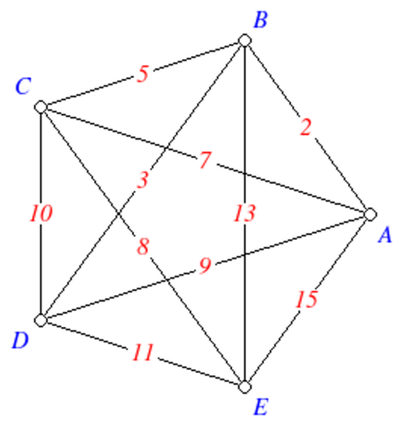
Apply the nearest neighbor algorithm to the graph above starting at vertex A. Give your answer as a list of vertices, starting and ending at vertex A. Example: ABCDA

Videos:
Nearest Neighbor Algorithm (RNNA)
https://www.youtube.com/watch?v=zPgsNsOfxQ8&t=313s

Repeat Nearest Neighbors Algorithm (RNNA)

- RNNA
  1. Do the Nearest Neighbor Algorithm starting at each vertex.
  2. Choose the circuit produced with the minimal weight.

Apply the repeated nearest neighbor algorithm to the graph above. Starting at which vertex or vertices produces the circuit of lowest cost?



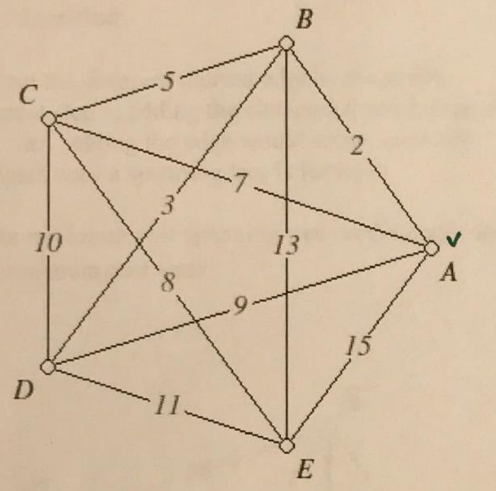
__A

__B

__C

__D

__E



__A → B → D → C → E → A
2 + 3 + 10 + 8 + 15 = 38

[✓ B] → A → C → E → D → B
2 + 7 + 8 + 11 + 3 = $\boxed{31}$

__C → B → A → D → E → C
5 + 2 + 9 + 11 + 8 = 35

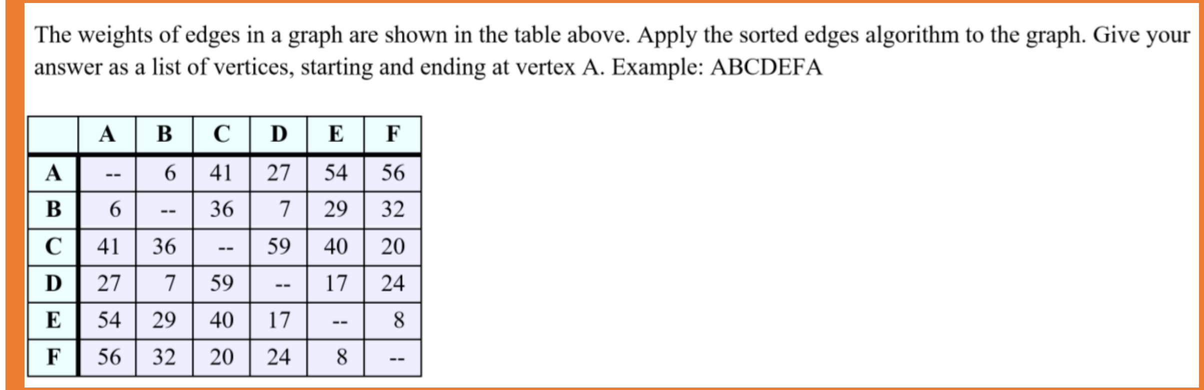
[✓ D] → B → A → C → E → D
3 + 2 + 7 + 8 + 11 = $\boxed{31}$

__E → C → B → A → D → E
8 + 5 + 2 + 9 + 10 = 34

Videos:
Repeated Nearest Neighbor Algorithm (RNNA)
https://www.youtube.com/watch?v=dcsiU3xtK2

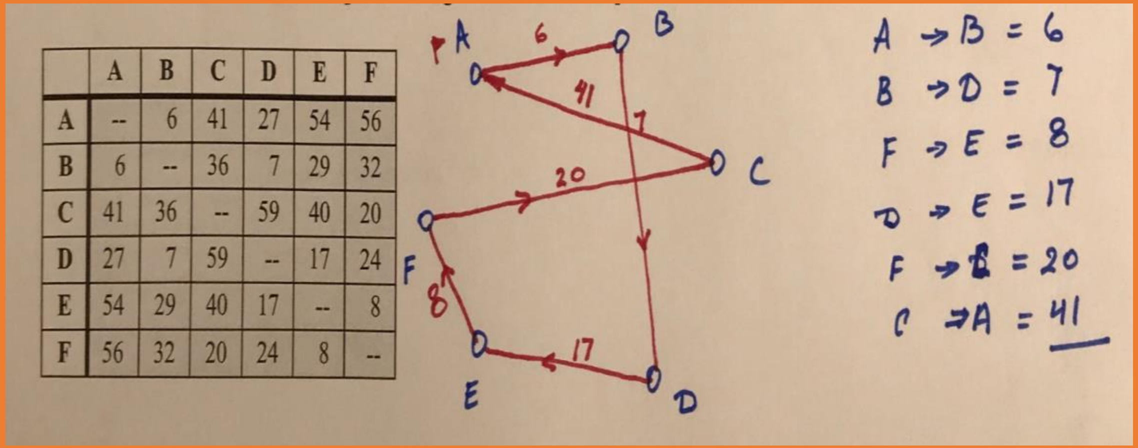
Sorted Edges Algorithm (a.k.a. Cheapest Link Algorithm)

- Sorted Edges Algorithm
  1. Select the cheapest unused edge in the graph.
  2. Repeat step 1, adding the cheapest unused edge to the circuit, unless:
     a) adding the edge would create a circuit that does not contain all vertices or
     b) adding the edge would give a vertex of degree 3.
  3. Repeat until a circuit containing all vertices is formed.



The weights of edges in a graph are shown in the table above. Apply the sorted edges algorithm to the graph. Give your answer as a list of vertices, starting and ending at vertex A. Example: ABCDEFA

	A	B	C	D	E	F
A	--	6	41	27	54	56
B	6	--	36	7	29	32
C	41	36	--	59	40	20
D	27	7	59	--	17	24
E	54	29	40	17	--	8
F	56	32	20	24	8	--



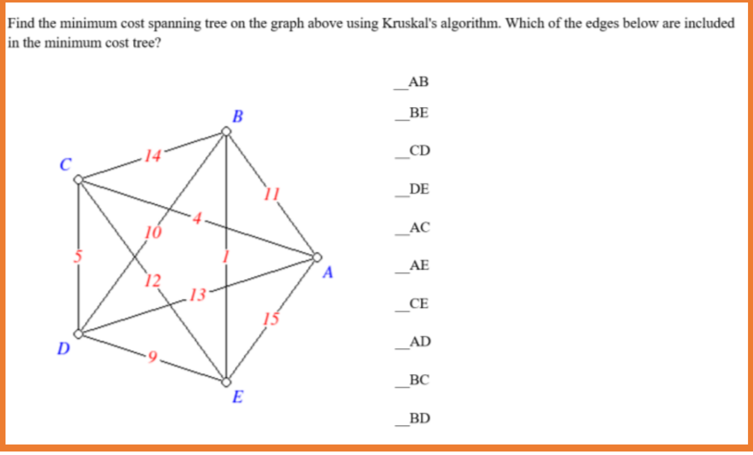
	A	B	C	D	E	F
A	--	6	41	27	54	56
B	6	--	36	7	29	32
C	41	36	--	59	40	20
D	27	7	59	--	17	24
E	54	29	40	17	--	8
F	56	32	20	24	8	--

$A \to B = 6$
$B \to D = 7$
$F \to E = 8$
$D \to E = 17$
$F \to C = 20$
$C \to A = 41$

Videos:
Sorted Edges Algorithm (Cheapest Link Algorithm)
https://www.youtube.com/watch?v=WUMxRp3xei0

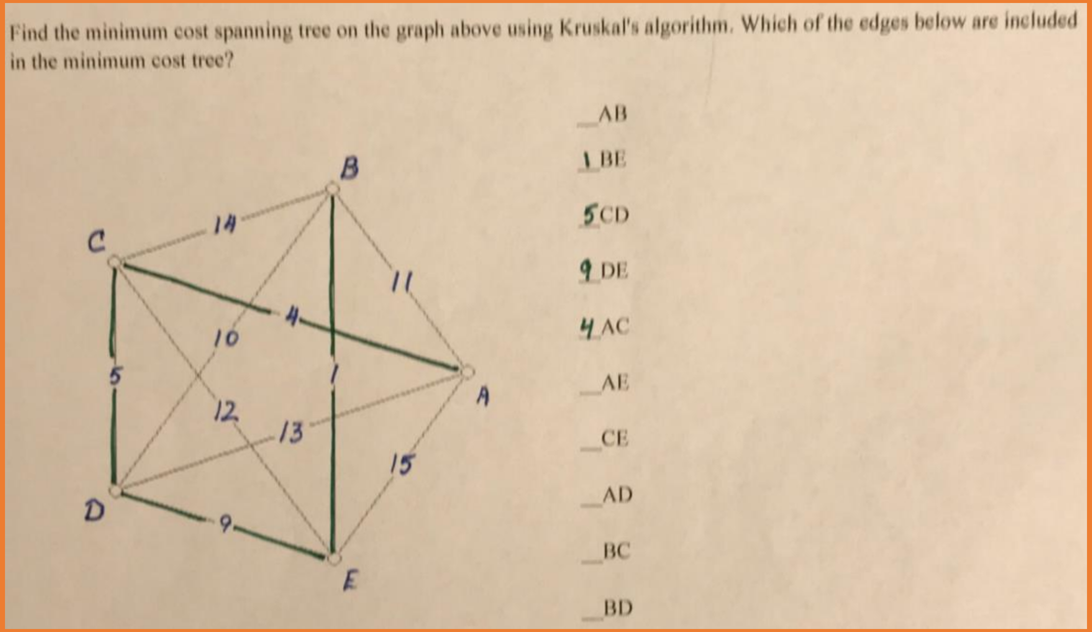
Spanning Tree

- Spanning Tree
  - Connected graph using all vertices in which there are no circuits. In other words, there is a path from any vertex to any other vertex, but no circuits.
- Minimum Cost Spanning Tree (MCST)
  - Spanning tree with the smallest total edge weight.
- Kruskal's Algorithm
  1. Select the cheapest unused edge in the graph.
  2. Repeat step 1, adding the cheapest unused edge, unless
     a) adding the edge would create a circuit.
  3. Repeat until a spanning tree is formed.



Find the minimum cost spanning tree on the graph above using Kruskal's algorithm. Which of the edges below are included in the minimum cost tree?

__AB
__BE
__CD
__DE
__AC
__AE
__CE
__AD
__BC
__BD



Find the minimum cost spanning tree on the graph above using Kruskal's algorithm. Which of the edges below are included in the minimum cost tree?

__AB
1 BE
5 CD
9 DE
4 AC
__AE
__CE
__AD
__BC
__BD

Videos:
Spanning Trees
https://www.youtube.com/watch?v=0ljjRM8hWjU
Graph Theory: Kruskal's Algorithm
https://www.youtube.com/watch?v=d4BEgzK08JE