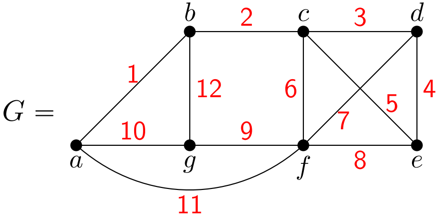
Warmup:

Let



(i) Verify that G is connected by giving an example of a walk from vertex a to each of the vertices b–g.

(ii) What is the shortest path from a to c? to e?

(iii) What is the longest path from a to c? to e?

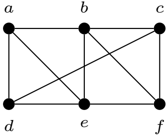
(iv) What is a longest path in G?

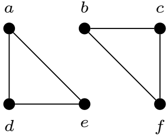
(v) Does G have any maximal paths that are shorter than the path in part (iv)?

(Recall a path is a walk with no repeated vertices or edges, and a maximal path is one that can't be extended in either direction.)

Recall from last time: A graph is connected if for every pair of vertices u and v, there is a walk from u to v.

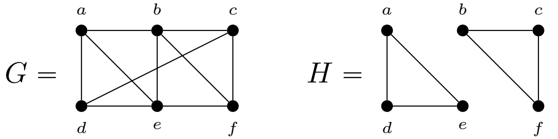
Recall from last time: A graph is connected if for every pair of vertices u and v, there is a walk from u to v. For example:

$G =$ 

$H =$ 

G is connected; H is not.

Recall from last time: A graph is connected if for every pair of vertices u and v, there is a walk from u to v. For example:



G is connected; H is not.

"Connected" is an equivalence relation on vertices: we say $u \sim v$ if there is a walk from u to v. A connected component of a graph is a maximally connected subgraph of G (H above has two connected components), i.e. the equivalence classes under the connectedness relation.

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite.

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.
4. Connected or not

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.
4. Connected or not
5. Paths or cycles of particular lengths

   Also: longest path or cycle length

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.
4. Connected or not
5. Paths or cycles of particular lengths

   Also: longest path or cycle length, maximal paths of certain lengths, ...

How connected?

Suppose G is connected – how can we evaluate how well connected it is? For example, if you're building a network of computers, can your network be disconnected if one computer or one line fails?

How connected?

Suppose G is connected – how can we evaluate how well connected it is? For example, if you're building a network of computers, can your network be disconnected if one computer or one line fails?

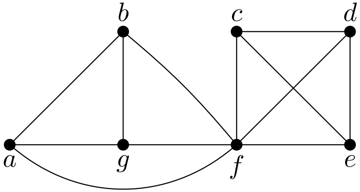
If the subgraph $G - v$ is not connected, we call v a cut vertex. Similarly, if $G - e$ is not connected, we call e a cut edge.

How connected?

Suppose G is connected – how can we evaluate how well connected it is? For example, if you're building a network of computers, can your network be disconnected if one computer or one line fails?

If the subgraph $G - v$ is not connected, we call v a cut vertex. Similarly, if $G - e$ is not connected, we call e a cut edge.

For example, in

$G =$ 

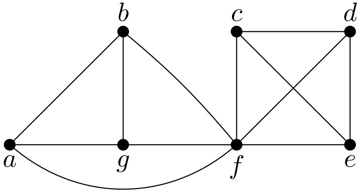
f is a cut vertex.

How connected?

Suppose G is connected – how can we evaluate how well connected it is? For example, if you're building a network of computers, can your network be disconnected if one computer or one line fails?

If the subgraph $G - v$ is not connected, we call v a cut vertex. Similarly, if $G - e$ is not connected, we call e a cut edge.

For example, in

$G =$ 

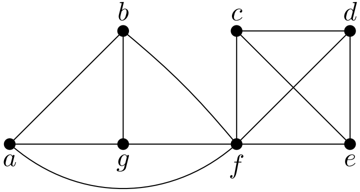
f is a cut vertex. G doesn't have any cut edges.

How connected?

If the subgraph $G - v$ is not connected, we call v a cut vertex.
Similarly, if $G - e$ is not connected, we call e a cut edge.

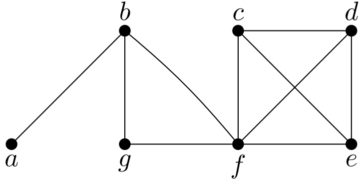
For example, in

$G =$



f is a cut vertex. G doesn't have any cut edges. In

$H =$

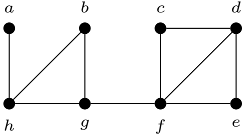


the cut vertices are b and f, and edge $a{-}b$ is the only cut edge.

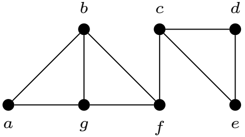
You try:

Identify the cut edges and vertices (if any) of the following graphs:

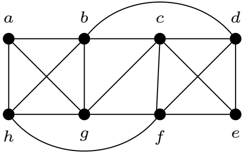
$G =$



$H =$



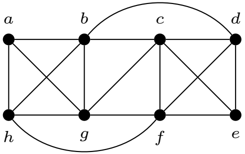
$K =$



If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.

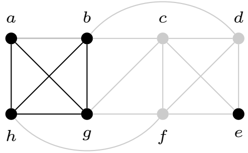
If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.

For example, in

$K =$ 

one example of a vertex cut is $\{c, d, f\}$.

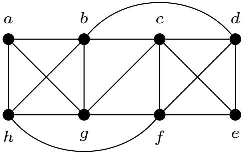
If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.
For example, in

$K =$ 

one example of a vertex cut is $\{c, d, f\}$.

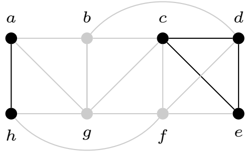
If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.
For example, in

$K =$



one example of a vertex cut is $\{c, d, f\}$.
Another vertex cut is $\{b, f, g\}$.

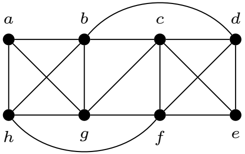
If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.
For example, in

$K =$ 

one example of a vertex cut is $\{c, d, f\}$.
Another vertex cut is $\{b, f, g\}$.

If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.
For example, in

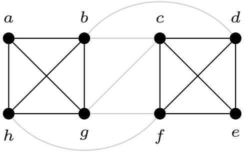
$K =$



one example of a vertex cut is $\{c, d, f\}$.
Another vertex cut is $\{b, f, g\}$.

One example of an edge cut is $\{b{-}c, b{-}d, c{-}g, f{-}g, f{-}h\}$.

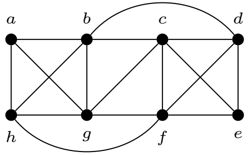
If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.
For example, in

$K =$ 

one example of a vertex cut is $\{c, d, f\}$.
Another vertex cut is $\{b, f, g\}$.

One example of an edge cut is $\{b{-}c, b{-}d, c{-}g, f{-}g, f{-}h\}$.

If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.
For example, in

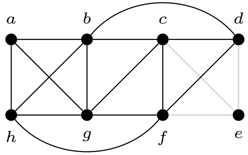
$K =$ 

one example of a vertex cut is $\{c, d, f\}$.
Another vertex cut is $\{b, f, g\}$.

One example of an edge cut is $\{b{-}c, b{-}d, c{-}g, f{-}g, f{-}h\}$.
Another edge cut is $\{c{-}e, d{-}e, e{-}f\}$.

If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.
For example, in

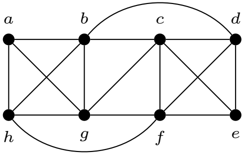
$K =$



one example of a vertex cut is $\{c, d, f\}$.
Another vertex cut is $\{b, f, g\}$.

One example of an edge cut is $\{b{-}c, b{-}d, c{-}g, f{-}g, f{-}h\}$.
Another edge cut is $\{c{-}e, d{-}e, e{-}f\}$.

If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.
For example, in

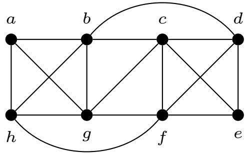
$K =$ 

one example of a vertex cut is $\{c, d, f\}$.
Another vertex cut is $\{b, f, g\}$.

One example of an edge cut is $\{b{-}c, b{-}d, c{-}g, f{-}g, f{-}h\}$.
Another edge cut is $\{c{-}e, d{-}e, e{-}f\}$.

CAREFUL! In "cut vertex", "vertex" is the noun and "cut" is the adjective; in "vertex cut", "cut" is the noun and "vertex" is the adjective.

If $W \subset V$ has the property that $G[V - W]$ is not connected, we call W a vertex cut. If $F \subset E$ has the property that $G - E$ is not connected, we say F is an edge cut.
For example, in

$K =$ 

one example of a vertex cut is $\{c, d, f\}$.
Another vertex cut is $\{b, f, g\}$.

One example of an edge cut is $\{b{-}c, b{-}d, c{-}g, f{-}g, f{-}h\}$.
Another edge cut is $\{c{-}e, d{-}e, e{-}f\}$.

CAREFUL! In "cut vertex", "vertex" is the noun and "cut" is the adjective; in "vertex cut", "cut" is the noun and "vertex" is the adjective. Same thing in "cut edge" versus "edge cut".

Vertex connectivity

Let $\kappa(G)$ be the fewest number of vertices needed to disconnect a graph (or to whittle it down to a single vertex, whichever is fewer).

Vertex connectivity

Let $\kappa(G)$ be the fewest number of vertices needed to disconnect a graph (or to whittle it down to a single vertex, whichever is fewer). We call $\kappa(G)$ the (vertex) connectivity of G.

Vertex connectivity

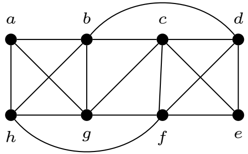
Let $\kappa(G)$ be the fewest number of vertices needed to disconnect a graph (or to whittle it down to a single vertex, whichever is fewer). We call $\kappa(G)$ the (vertex) connectivity of G.

How to compute: To show $\kappa(G) = k$, you have to give a vertex cut of size k **and** show that removing all possible subsets of size $< k$ leaves a connected graphs.

Vertex connectivity

Let $\kappa(G)$ be the fewest number of vertices needed to disconnect a graph (or to whittle it down to a single vertex, whichever is fewer). We call $\kappa(G)$ the (vertex) connectivity of G.

How to compute: To show $\kappa(G) = k$, you have to give a vertex cut of size k **and** show that removing all possible subsets of size $< k$ leaves a connected graphs.

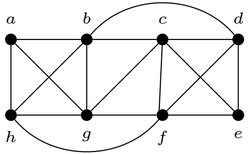
$G =$ 

Vertex connectivity

Let $\kappa(G)$ be the fewest number of vertices needed to disconnect a graph (or to whittle it down to a single vertex, whichever is fewer). We call $\kappa(G)$ the (vertex) connectivity of G.

How to compute: To show $\kappa(G) = k$, you have to give a vertex cut of size k **and** show that removing all possible subsets of size $< k$ leaves a connected graphs.

$G =$

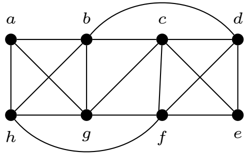


$$\boxed{\kappa(G) = 3}$$

Vertex connectivity

Let $\kappa(G)$ be the fewest number of vertices needed to disconnect a graph (or to whittle it down to a single vertex, whichever is fewer). We call $\kappa(G)$ the (vertex) connectivity of G.

How to compute: To show $\kappa(G) = k$, you have to give a vertex cut of size k **and** show that removing all possible subsets of size $< k$ leaves a connected graphs.

$G =$ 

$\kappa(G) = 3$

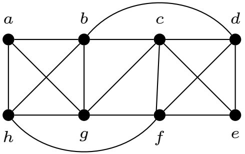
Note that the complete graph has no vertex cut, so we define $\kappa(K_n) = n - 1$. Thus

$$0 \leqslant \kappa(G) \leqslant |V| - 1.$$

Vertex connectivity

Let $\kappa(G)$ be the fewest number of vertices needed to disconnect a graph (or to whittle it down to a single vertex, whichever is fewer). We call $\kappa(G)$ the (vertex) connectivity of G.

How to compute: To show $\kappa(G) = k$, you have to give a vertex cut of size k **and** show that removing all possible subsets of size $< k$ leaves a connected graphs.

$G =$ 

$\kappa(G) = 3$

Note that the complete graph has no vertex cut, so we define $\kappa(K_n) = n - 1$. Thus

$$0 \leqslant \kappa(G) \leqslant |V| - 1.$$

The larger the κ, the more connected the graph. We say G is k-connected if $\kappa(G) \geqslant k$.

Edge connectivity

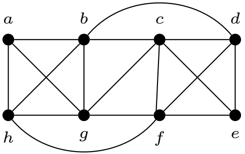
Let $\lambda(G)$ be the fewest number of edges needed to disconnect a graph. We call $\lambda(G)$ the edge connectivity of G.

Edge connectivity

Let $\lambda(G)$ be the fewest number of edges needed to disconnect a graph. We call $\lambda(G)$ the edge connectivity of G.
How to compute: To show $\lambda(G) = \ell$, you have to give an edge cut of size ℓ **and** show that removing all possible subsets of size $< \ell$ leaves a connected graphs.

Edge connectivity

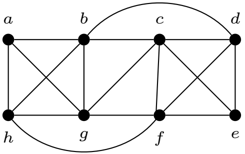
Let $\lambda(G)$ be the fewest number of edges needed to disconnect a graph. We call $\lambda(G)$ the edge connectivity of G.
How to compute: To show $\lambda(G) = \ell$, you have to give an edge cut of size ℓ **and** show that removing all possible subsets of size $< \ell$ leaves a connected graphs.

$G =$ 

Edge connectivity

Let $\lambda(G)$ be the fewest number of edges needed to disconnect a graph. We call $\lambda(G)$ the edge connectivity of G.
How to compute: To show $\lambda(G) = \ell$, you have to give an edge cut of size ℓ **and** show that removing all possible subsets of size $< \ell$ leaves a connected graphs.

$G =$



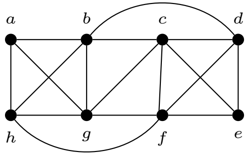
$\boxed{\lambda(G) = 3}$

Edge connectivity

Let $\lambda(G)$ be the fewest number of edges needed to disconnect a graph. We call $\lambda(G)$ the edge connectivity of G.

How to compute: To show $\lambda(G) = \ell$, you have to give an edge cut of size ℓ **and** show that removing all possible subsets of size $< \ell$ leaves a connected graphs.

$G =$

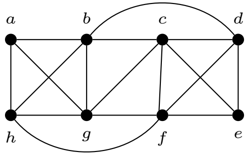


$$\boxed{\lambda(G) = 3}$$

Note that we can disconnect a graph by removing all the edges around a single vertex.

Edge connectivity

Let $\lambda(G)$ be the fewest number of edges needed to disconnect a graph. We call $\lambda(G)$ the edge connectivity of G.
How to compute: To show $\lambda(G) = \ell$, you have to give an edge cut of size ℓ **and** show that removing all possible subsets of size $< \ell$ leaves a connected graphs.

$G =$ 

$\boxed{\lambda(G) = 3}$

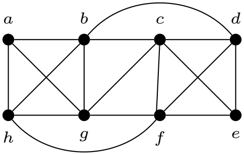
Note that we can disconnect a graph by removing all the edges around a single vertex. So

$$\lambda(G) \leqslant \min_{v \in V} \deg(v).$$

Moreover, if we remove all the vertices adjacent to v, then v is isolated.

Edge connectivity

Let $\lambda(G)$ be the fewest number of edges needed to disconnect a graph. We call $\lambda(G)$ the edge connectivity of G.
How to compute: To show $\lambda(G) = \ell$, you have to give an edge cut of size ℓ **and** show that removing all possible subsets of size $< \ell$ leaves a connected graphs.

$G =$ 

$\lambda(G) = 3$

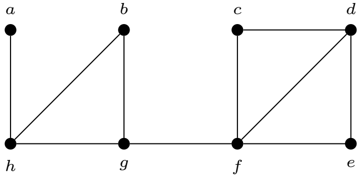
Note that we can disconnect a graph by removing all the edges around a single vertex. So

$$\lambda(G) \leqslant \min_{v \in V} \deg(v).$$

Moreover, if we remove all the vertices adjacent to v, then v is isolated. So

$$\kappa(G) \leqslant \lambda(G) \leqslant \min_{v \in V}(\deg(v)).$$

$G =$

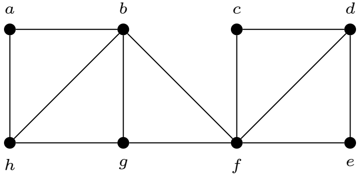


$\kappa(G) :$

$\lambda(G) :$

$\min_{v \in V}(\deg(v)) :$

$H =$



$\kappa(G) :$

$\lambda(G) :$

$\min_{v \in V}(\deg(v)) :$

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.
4. Paths or cycles of particular lengths

   Also: longest path or cycle length, maximal paths of certain lengths, ...
5. Edge and vertex connectivity

Graph invariants

Recall, a graph invariant is a statistic about a graph that is preserved under isomorphisms (relabeling of the vertices). Namely, if you don't need the labels to calculate the statistic, then it's probably a graph invariant.

1. $|V|$, $|E|$
2. Degree sequence

   Also: Minimum degree, maximum degree, vertex of degree d_1 adjacent to vertex of degree d_2, ...
3. Bipartite or not

   If any subgraph is not bipartite, then G is not bipartite. A graph is bipartite if and only if it has no odd cycles as subgraphs.
4. Paths or cycles of particular lengths

   Also: longest path or cycle length, maximal paths of certain lengths, ...
5. Edge and vertex connectivity

You try: Exercise 52.

Aside: necessary and sufficient conditions

A necessary condition is a condition that must be present for an event to occur.

Aside: necessary and sufficient conditions

A necessary condition is a condition that must be present for an event to occur.
Some examples:

- Event: Passing 365; NC: take all three exams.
- Event: Staying alive; NC: breathing.
- Event: $x^2 = 1$; NC: $x \in \mathbb{Z}$.

Aside: necessary and sufficient conditions

A necessary condition is a condition that must be present for an event to occur.
Some examples:

- Event: Passing 365; NC: take all three exams.
- Event: Staying alive; NC: breathing.
- Event: $x^2 = 1$; NC: $x \in \mathbb{Z}$.

A sufficient condition is a condition or set of conditions that will produce the event.

Aside: necessary and sufficient conditions

A necessary condition is a condition that must be present for an event to occur.
Some examples:

- Event: Passing 365; NC: take all three exams.
- Event: Staying alive; NC: breathing.
- Event: $x^2 = 1$; NC: $x \in \mathbb{Z}$.

A sufficient condition is a condition or set of conditions that will produce the event.
Some examples:

- Event: Passing 365;
  SC: do all of the homework and get 100% on all exams and quizzes.
- Event: Being a parent; SC: having a daughter.
- Event: $x^2 = 1$; SC: $x = -1$.

Sufficient conditions	imply	Event	implies	Necessary conditions

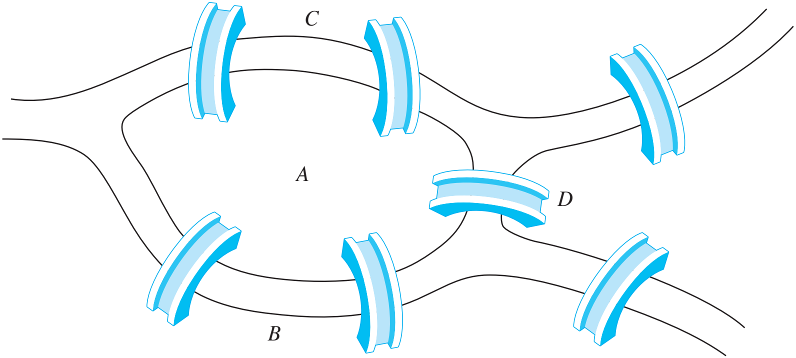
Eulerian trails and circuits

Suppose you're trying to design a maximally efficient route for postal delivery, or street cleaning. You want walk on the city streets that visits every street exactly once.

Eulerian trails and circuits

Suppose you're trying to design a maximally efficient route for postal delivery, or street cleaning. You want walk on the city streets that visits every street exactly once.

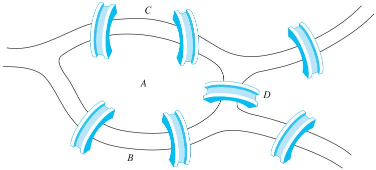
"The Seven Bridges of Königsberg", Leonhard Euler (1736)

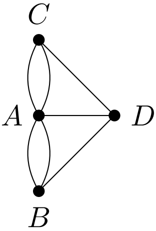


Question: is it possible to start at some location in the town, travel across all the bridges once without crossing any bridge twice?

Eulerian trails and circuits

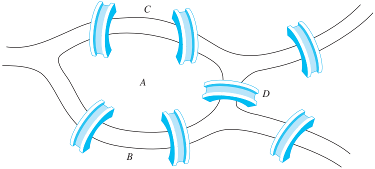
What Euler did was model the problem as the multigraph

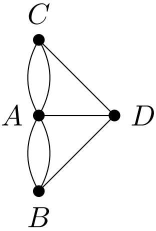




Eulerian trails and circuits

What Euler did was model the problem as the multigraph

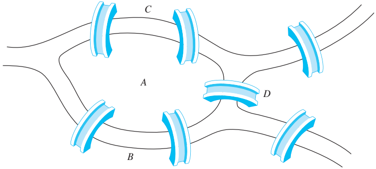


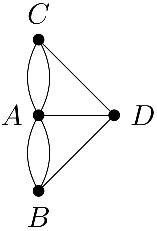


In honor of his contribution, we say that an Eulerian trail in a graph G is a trail (no repeated edges) that passes through every edge of G (exactly once).

Eulerian trails and circuits

What Euler did was model the problem as the multigraph

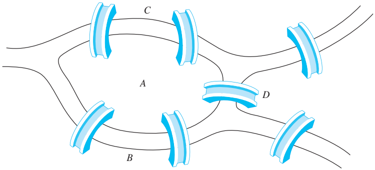


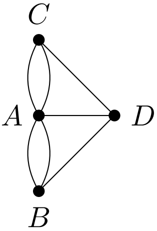


In honor of his contribution, we say that an Eulerian trail in a graph G is a trail (no repeated edges) that passes through every edge of G (exactly once). An Eulerian circuit is an Eulerian trail that ends where it started.

Eulerian trails and circuits

What Euler did was model the problem as the multigraph



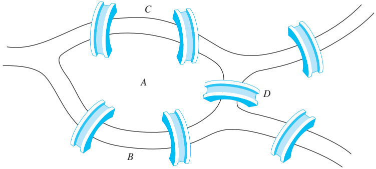


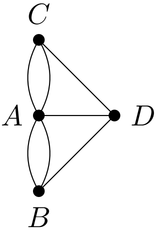
In honor of his contribution, we say that an Eulerian trail in a graph G is a trail (no repeated edges) that passes through every edge of G (exactly once). An Eulerian circuit is an Eulerian trail that ends where it started.

Necessary: Connected

Eulerian trails and circuits

What Euler did was model the problem as the multigraph



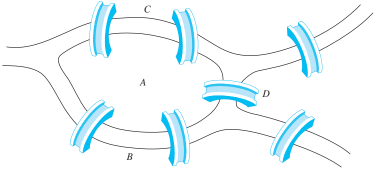


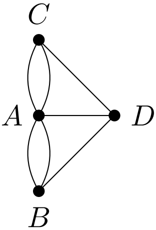
In honor of his contribution, we say that an Eulerian trail in a graph G is a trail (no repeated edges) that passes through every edge of G (exactly once). An Eulerian circuit is an Eulerian trail that ends where it started.

Necessary: Connected; at most two vertices of odd degree.

Eulerian trails and circuits

What Euler did was model the problem as the multigraph





In honor of his contribution, we say that an Eulerian trail in a graph G is a trail (no repeated edges) that passes through every edge of G (exactly once). An Eulerian circuit is an Eulerian trail that ends where it started.

Necessary: Connected; at most two vertices of odd degree. This is also a sufficient condition. Why? ...

Eulerian trails and circuits

An Eulerian trail in a graph G is a trail (no repeated edges) that passes through every edge of G (exactly once). An Eulerian circuit is an Eulerian trail that ends where it started.

Necessary: Connected; at most two vertices of odd degree. This is also a sufficient condition. Why? ...

Eulerian trails and circuits

An Eulerian trail in a graph G is a trail (no repeated edges) that passes through every edge of G (exactly once). An Eulerian circuit is an Eulerian trail that ends where it started.

Necessary: Connected; at most two vertices of odd degree. This is also a sufficient condition. Why? ...

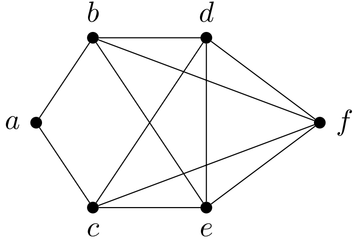
Algorithm for finding an Eulerian circuit in any graph with all even degree vertices: Start anywhere and go until you get stuck – you'll be back where you started. Somewhere in the middle, you have a vertex where you didn't exhaust the edges incident. Go back and start from there and go until you get stuck. Repeat.

Eulerian trails and circuits

An Eulerian trail in a graph G is a trail (no repeated edges) that passes through every edge of G (exactly once). An Eulerian circuit is an Eulerian trail that ends where it started.

Necessary: Connected; at most two vertices of odd degree. This is also a sufficient condition. Why? ...

Algorithm for finding an Eulerian circuit in any graph with all even degree vertices: Start anywhere and go until you get stuck – you'll be back where you started. Somewhere in the middle, you have a vertex where you didn't exhaust the edges incident. Go back and start from there and go until you get stuck. Repeat.

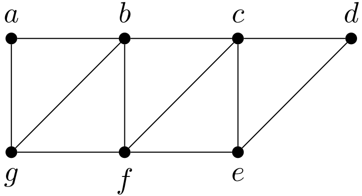


Eulerian trails and circuits

An Eulerian trail in a graph G is a trail (no repeated edges) that passes through every edge of G (exactly once). An Eulerian circuit is an Eulerian trail that ends where it started.

Necessary: Connected; at most two vertices of odd degree. This is also a sufficient condition. Why? ...

Algorithm for finding an Eulerian trail in any graph with all but two even degree vertices: Start at an odd-degree vertex and go until you get stuck. Somewhere in the middle, you have a vertex where you didn't exhaust the edges incident. Go back and start from there and go until you get stuck. Repeat.



Eulerian trails and circuits

Theorem

A graph has an Eulerian trail if and only if it is connected and has at most two vertices of odd degree. Further, a connected graph has an Eulerian circuit if and only if every vertex is of even degree.

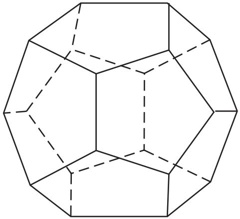
Hamilton paths and cycles

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

Hamilton paths and cycles

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

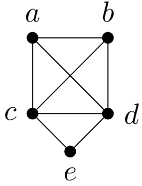
Famous puzzle: start with a dodecahedron, and imagine the vertices are cities around the world, and the edges are routes from one city to the next. The goal is to visit every city exactly once.

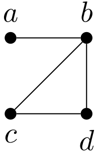


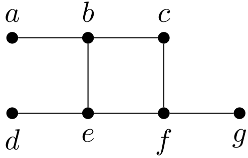
Hamilton paths and cycles

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

Some simpler examples:



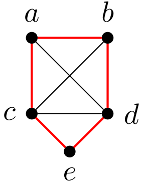


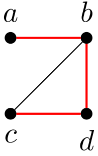


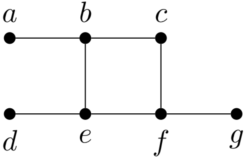
Hamilton paths and cycles

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

Some simpler examples:





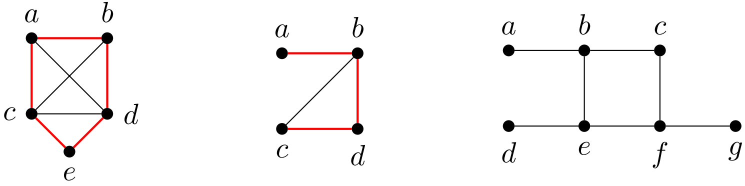


The first has a Hamilton cycle, the second has a Hamilton path, the third has neither (you'd get stuck at a, d, or g without hitting at least one of those three).

Hamilton paths and cycles

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

Some simpler examples:



The first has a Hamilton cycle, the second has a Hamilton path, the third has neither (you'd get stuck at a, d, or g without hitting at least one of those three).

See also: traveling salesman problem
"What is the shortest route a traveling salesperson should take to visit a set of cities?"

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

In contrast to Eulerian trails, **there are no simple necessary and sufficient conditions for the existence of Hamilton paths and cycles**.

Necessary conditions

- Paths: no more than two vertices of degree 1.

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

In contrast to Eulerian trails, **there are no simple necessary and sufficient conditions for the existence of Hamilton paths and cycles**.

Necessary conditions

- Paths: no more than two vertices of degree 1.
- Cycles:
  - No vertices of degree 1.

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

In contrast to Eulerian trails, **there are no simple necessary and sufficient conditions for the existence of Hamilton paths and cycles**.

Necessary conditions

- Paths: no more than two vertices of degree 1.
- Cycles:
  - No vertices of degree 1.
  - If a vertex has degree 2, you know both edges incident must be in the cycle.

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

In contrast to Eulerian trails, **there are no simple necessary and sufficient conditions for the existence of Hamilton paths and cycles**.

Necessary conditions

- Paths: no more than two vertices of degree 1.
- Cycles:
  - No vertices of degree 1.
  - If a vertex has degree 2, you know both edges incident must be in the cycle.
  - No cut vertices or edges.

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

In contrast to Eulerian trails, **there are no simple necessary and sufficient conditions for the existence of Hamilton paths and cycles**.

Sufficient conditions

Note: the more edges a graph has, the more likely it is that there's a Hamilton cycle.

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

In contrast to Eulerian trails, **there are no simple necessary and sufficient conditions for the existence of Hamilton paths and cycles**.

Sufficient conditions

Note: the more edges a graph has, the more likely it is that there's a Hamilton cycle.

Dirac's Theorem

If G is a simple connected graph with $n \geqslant 3$ vertices, such that

$$\min_{v \in V} \deg(v) \geqslant n/2,$$

then G has a Hamilton circuit.

A Hamilton path is a path in G that visits every vertex exactly once. A Hamilton cycle is a cycle that visits every vertex in G.

In contrast to Eulerian trails, **there are no simple necessary and sufficient conditions for the existence of Hamilton paths and cycles**.

Sufficient conditions

Note: the more edges a graph has, the more likely it is that there's a Hamilton cycle.

Dirac's Theorem

If G is a simple connected graph with $n \geqslant 3$ vertices, such that

$$\min_{v \in V} \deg(v) \geqslant n/2,$$

then G has a Hamilton circuit.

Ore's Theorem

If G is a simple connected graph with $n \geqslant 3$ vertices such that

$$\deg(u) + \deg(v) \geqslant n$$

for every pair of non-adjacent vertices u and v, then G has a Hamilton circuit.

(Note that Ore's theorem implies Dirac's theorem.)