We can use induction to prove things about graphs.

Example: Every path P_n is bipartite.

Proof: By induction on n.

Base Case ($n=1$): Color the vertex ~~[illegible]~~ red. Now all adjacent vertices have different colours.

Inductive Hypothesis: Assume that every path P_n with $n \leq k$ is bipartite.

Inductive Step: We need to show that P_{k+1} is bipartite. If we remove the latst vertex of P_{k+1}, v_{k+1}, we get P_k. By the IH, P_k is bipartite, so we can colour its vertices red and blue such that ~~no to~~ all edges ~~connect~~ have one red and one blue endpoint. ~~We~~ To turn this back into P_{k+1}, we need to add v_{k+1} and ~~add~~ the edge (v_k, v_{k+1}). To maintain the bipartite property, we must give v_{k+1} the opposite colour of v_k. This is a valid 2-colouring of P_{k+1}, so it is ~~bi~~ also bipartite.

Connectedness

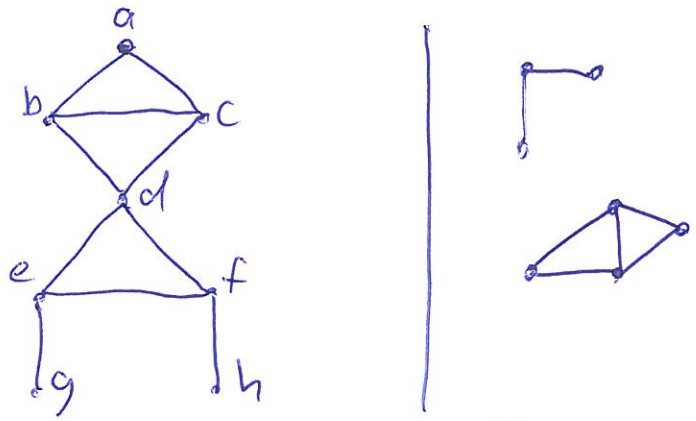
L12

A path in a graph $G=(V,E)$ is a sequence of vertices $v_0, v_1, v_2, \ldots, v_k$, such that $(v_i, v_{i+1}) \in E$ for all $0 \le i \le k-1$. This is a path between v_0 and v_k.

G is connected if there is a path between any two vertices.

A cycle in a graph is a path with $v_0 = v_k$.

Example:



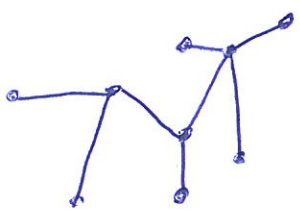
Connected

Disconnected

- a, b, d, c, a is a cycle.

A connected graph without cycles is called a tree.

Example:



Trees are used in many data structures (COMP 2402)

Claim: Every tree with $n \geq 2$ vertices has at least two leaves: vertices of degree 1.

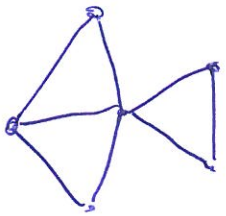
Proof: Suppose ~~one of~~ a ~~the~~ longest path in the tree is $v_0, v_1, \ldots, v_k$, which has length k. Now suppose that ~~v_0 or~~ v_k does not have degree 1. ~~They must have degree at least 1, so their d~~ Then it must have degree greater than 1, ~~Assume that it is v_k~~ (the argument for v_0 is similar). ~~Then v_k~~ so it has a neighbour u other than v_{k-1}. There are two cases. If u is on the path, ie. $u = v_j$ for some $0 \leq j \leq k-2$, then ~~the tree~~ $v_k, u, v_{j+1}, \ldots, v_k$ is a cycle. But a tree has no cycles, so ~~this is impossible. Then~~ u cannot be on the path. But then $v_0, \ldots, v_k, u$ is a longer path in the tree, which contradicts our assumption that $v_0, \ldots, v_k$ was a ~~the~~ longest path. Therefore u cannot exist - both v_0 and v_k ~~are~~ must be leaves.

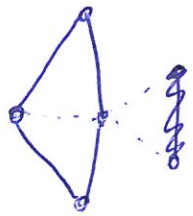
~~How can we test if a~~

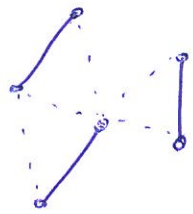
Try proving this by induction!

A subgraph of a graph $G=(V,E)$ is a graph $G'=(V',E')$ such that $V' \subseteq V$, $E' \subseteq E$, and for all $(u,v) \in E'$, it holds that $u \in V' \wedge v \in V'$.

Example:

$G =$ 

$G_1' =$ 

$G_2'' =$ 

A subgraph is spanning if $V'=V$.

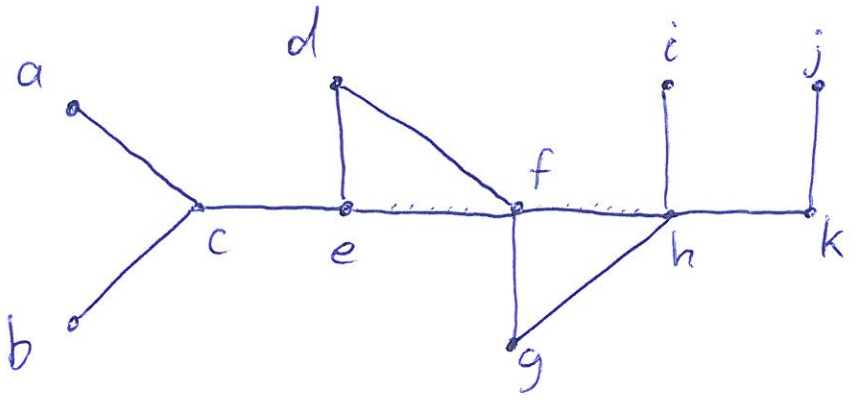
Depth-first search

How do you check if a graph is connected?

~~Input: Graph $G=(V,E)$~~
~~Output:~~

DFS
1. Mark all vertices as unvisited
2. ~~sta~~Pick a starting vertex
3. Go to an unvisited neighbour
4. Repeat until ~~all neighbours of~~ you get to a vertex that has no unvisited neighbours
5. Go back.
6. If you visited all vertices → Connected
   Otherwise → Disconnected

Example:



(start at f)

We visited the vertices in order f, g, h, k, j, i, d, e, c, a, b

The edges we followed form a spanning tree: the DFS-tree.
(Why no cycles?)

We can use the DFS-tree T to find out if G is bipartite:

- 2-colour T (all trees are bipartite)
- check if any edge connects vertices with the same colour.

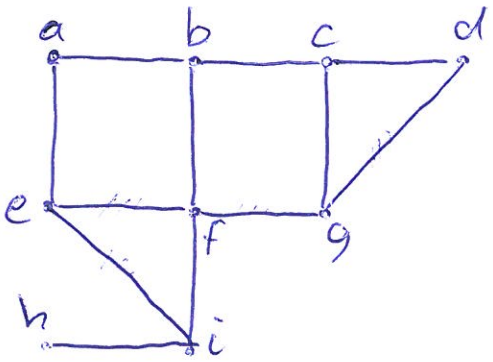
It also finds a path between any two nodes, but the path might be longer than necessary.

How can we find the shortest path?

Breadth-first search

1. Mark all vertices as unvisited, except for v.
2. $L \leftarrow [v]$
3. ~~while~~ Take the first vertex from L. ~~out~~, ~~and~~ add all its unvisited neighbours to the end of L. ~~if they're not already there~~ and mark them as visited.
4. Repeat until L is empty.

Example:



We visited the vertices in order: b, a, c, f, e, d, g, i, h

The edges we followed form a spanning tree: the BFS-tree.

(Start at b)

L = [b]
L = [a, c, f]
L = [c, f, e]
L = [f, e, d, g]
L = [e, d, g, i]
L = [d, g, i]
L = [g, i]
L = [i]
L = [h]
L = []

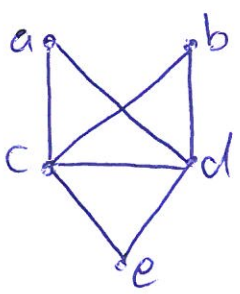
The BFS-tree contains a shortest path to any vertex from b!

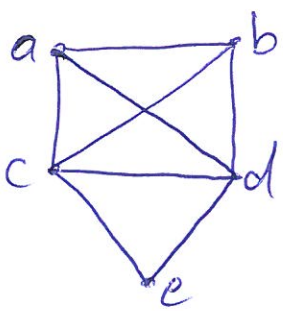
Special Cycles

A cycle $v_0, v_1, \ldots, v_0$ that uses every edge exactly once is an Euler cycle.

A cycle ~~that~~ that uses every ~~ed~~ vertex exactly once is a Hamilton cycle.

Example:





- c, a, d, b, c, e, d, c is an Euler cycle
- No Hamilton cycle

- a, b, d, e, c, a is a Hamilton cycle
- No Euler cycle

Claim: A connected graph G has an Euler cycle iff ~~G is connected and~~ every vertex has even degree.

Proof Sketch: ~~Induction on $|E|$. Base case ▽~~

$\Rightarrow$: We always pass through vertices without reusing edges: ✗✗, thus even degree.

~~Torah~~ Lucas

$\Leftarrow$: Induction on $|E|$. Base case: ▽ ✓

Step: start at v and walk. Even degree means we can't get stuck, so eventually we get back to v. Remove all edges we took. ~~Now~~ Now multiple pieces, but degrees are still even. Apply IH on pieces and stich cycles together.