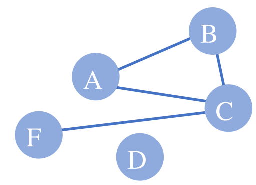
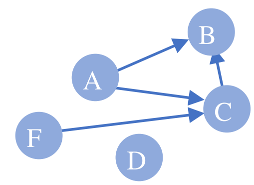
Graph Terminology



A *undirected graph* is a pair (`V`, `E`) where

1. `V` is a finite set of objects called *vertexes*, *vertices*, or *nodes*.
2. `E` is a set of pairs {`u`, `v`}, called *edges*, where `u` and `v` are nodes in `V`.

A *directed graph* is a pair (`V`, `E`) where



1. `V` is a finite set of objects called *vertexes*, *vertices*, or *nodes*.
2. `E` is a set of *ordered pairs* (`u`, `v`), called *edges*, where `u` and `v` are nodes in `V`. An edge (`u`, `v`) is depicted by an arrow leaving `u` and ending in `v`.

A directed graph is sometimes called a *digraph*.

Adjacency and degree

The nodes of an undirected edge {`u`, `v`} and a directed edge (`u`, `v`) are called the *endpoints* of the edge. For a directed edge (`u`, `v`), `u` is the *source* and `v` the *sink*.

Two nodes are *adjacent* if they are connected by an edge.

The *outdegree* of a vertex `u` in a directed graph is the number of edges for which `u` is the source.
The *indegree* of a vertex `u` in a directed graph is the number of edges for which `u` is the sink.
The *degree* of a node `u` in an undirected graph is the number of edges for which `u` is an endpoint.

Paths and cycles

A path is a sequence (v_0, v_1, …, v_p) of nodes such that for k, $0 \le k < p$,

If the graph is undirected, {v_k, v_{k+1}} is an edge,
If the graph is directed (v_k, v_{k+1}) is an edge.

The *length* of the path is the number of edges in it. This is 1 less than the number of nodes in the sequence!

Here are paths in the undirected and directed graphs shown above to the right:

(`C`) is a path of length 0 in both graphs.
(`F`, `C`, `B`) is a path of length 2 in both graphs.
(`F`, `C`, `A`) is a path of length 2 in the undirected graph. It is not a path in the directed graph.

A path is *simple* if all nodes in it are different. All three paths shown in the previous paragraph are simple. The path (`F`, `C`, `A`, `B`, `C`) in the undirected graph is not simple.

A path is a *cycle* if its length is ≥ 1 and its first and last nodes are the same. Paths (`C`, `A`, `B`, `C`) and (`C`, `A`, `C`, `A`, `C`) in the undirected graph are cycles. Path (`F`, `C`, `A`, `B`, `C`) in the undirected graph is not a cycle.

A cycle is *simple* if its only repeated nodes are its first and last nodes. Cycle (`C`, `A`, `B`, `C`) in the undirected graph is simple; cycle (`C`, `A`, `C`, `A`, `C`) in the undirected graph are is not simple.

Acyclic graphs and dags

A graph is *acyclic* if does not contain a cycle. The undirected graph shown above is not acyclic; the directed graph is acyclic.

A *d*irected *a*cyclic *g*raph is called a *DAG*. The directed graph shown above is a DAG.

Here is an example of a DAG. Let the nodes be the courses that can be taken to satisfy a computer science (or other) major. Draw a directed edge from course `c1` to course `c2` if `c1` is a prerequisite for `c2`. This graph better not have a cycle! Also, one can analyze the graph to find the longest prerequisite chain —it should not be more than 8, so students can graduate in eight semesters.

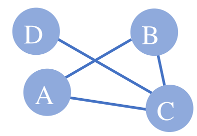
DAGs have many uses. In another pdf file on DAGs, we develop an algorithm called *topological sort*, which can determine whether a directed graph is a DAG.

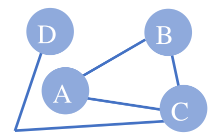
Graph coloring

A coloring of an undirected graph is an assignment of a color to each node such that no two adjacent nodes get the same color.

Here's one application of graph coloring. Consider the graph in which the nodes are tasks to be performed. There is an edge between two nodes/tasks if they require the same shared resource, so they cannot be executed simultaneously. Produce a graph coloring. Then, two nodes/tasks have different colors if they can be carried out simultaneously. Think of the colors as *time* slots in which to schedule the tasks. Hence, the minimum number of colors needed to color the graph is the minimum number of time slots needed to carry out all tasks.

Planarity



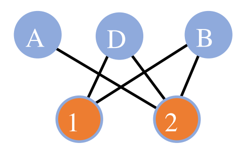


A graph is *planar* if it can be drawn in the plane without any edge crossing.

Be careful with this definition! The first graph on the right doesn't look planar because edges (`D`, `C`) and (`A`, `B`) cross. But edge (`D`, `C`) can be redrawn so that the edges don't cross, so this graph *is* planar.

Pdf file *PlanarGraphs* discusses planar graphss in more detail.

Bipartite graphs



A directed or undirected graph is *bipartite* if its nodes can be partitioned into two sets such that no edge connects two nodes in the same set. An example of a bipartite graph appears to the right, with the two sets being {`A`, `D`, `B` } and {`1`, `2`}.

The following three properties are equivalent.

1. Graph G is bipartite
2. Graph G is 2-colorable
3. Graph G has no cycles of odd length.