**Graph**

A **graph** (sometimes called an *undirected graph* to distinguish it from a [directed graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)#Directed_graph), or a *simple graph* to distinguish it from a [multigraph](https://en.wikipedia.org/wiki/Multigraph))[[4]](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)#cite_note-FOOTNOTEBenderWilliamson2010148-4)[[5]](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)#cite_note-5) is a [pair](https://en.wikipedia.org/wiki/Ordered_pair) *G* = (*V*, *E*), where *V* is a set whose elements are called *vertices* (singular: vertex), and *E* is a set of paired vertices, whose elements are called *edges* (sometimes *links* or *lines*).

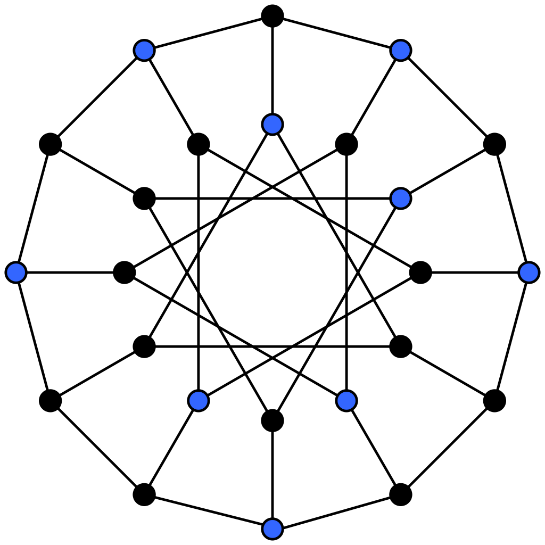
The vertices *x* and *y* of an edge {*x*, *y*} are called the *endpoints* of the edge. The edge is said to *join* *x* and *y* and to be *incident* on *x* and *y*. A vertex may belong to no edge, in which case it is not joined to any other vertex.

**Graph theory**

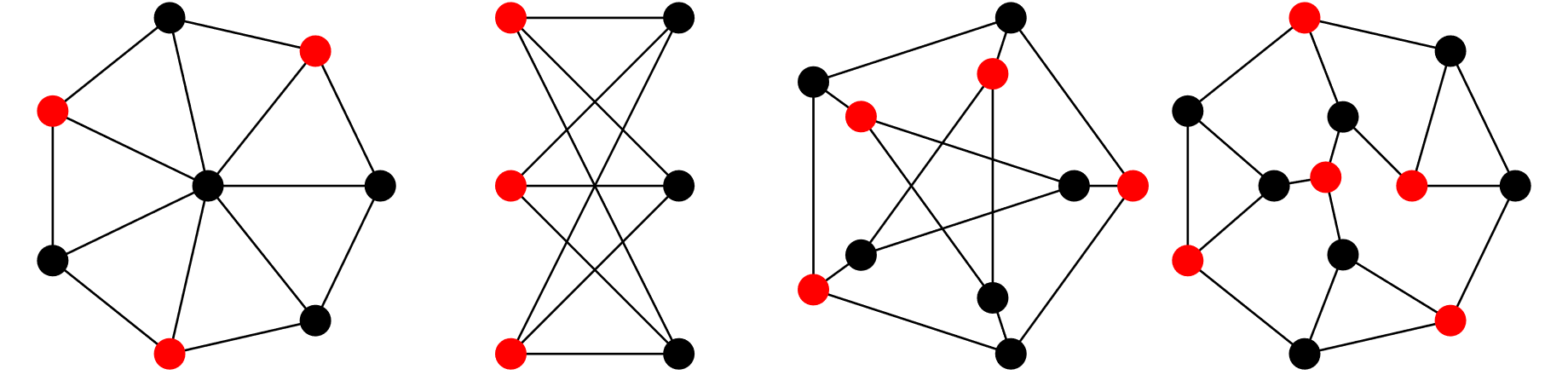
**Graph theory** is the study of [*graphs*](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)), which are mathematical structures used to model pairwise relations between objects.

**Independent set**

An **independent set**, **stable set**, **coclique** or **anticlique** is a set of [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) in a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)), no two of which are adjacent.



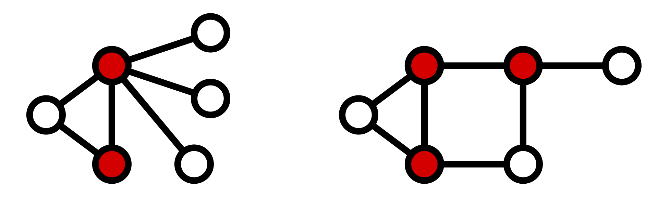
A **maximal independent set** (**MIS**) or **maximal stable set** is an [independent set](https://en.wikipedia.org/wiki/Independent_set_(graph_theory)) that is not a [subset](https://en.wikipedia.org/wiki/Subset) of any other independent set. In other words, there is no [vertex](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) outside the independent set that may join it because it is maximal with respect to the independent set property.



**Vertex cover**

A **vertex cover** (sometimes **node cover**) of a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) is a set of [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) that includes at least one endpoint of every [edge](https://en.wikipedia.org/wiki/Edge_(graph_theory)) of the graph.

The **minimum vertex cover problem** is the [optimization problem](https://en.wikipedia.org/wiki/Optimization_problem) of finding a smallest vertex cover in a given graph.



Complement of Vertex Cover is an Idependent set.  
Hence, finding a Maximum Independent Set (MIS) equivalent to finding a Minimum Vertex Cover

Diagram

Description automatically generated

**Matching**

A **matching** (or **independent edge set)** in an undirected [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) is a set of [edges](https://en.wikipedia.org/wiki/Edge_(graph_theory)) without common [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)).[[1]](https://en.wikipedia.org/wiki/Matching_(graph_theory)#cite_note-NetworkX_2.8.2_documentation-1) In other words, a subset of the edges is a matching if each vertex appears in at most one edge of that matching.

A **maximum matching** (also known as **maximum-cardinality matching**[[2]](https://en.wikipedia.org/wiki/Matching_(graph_theory)#cite_note-2)) is a matching that contains the largest possible number of edges. There may be many maximum matchings.

A picture containing application

Description automatically generated

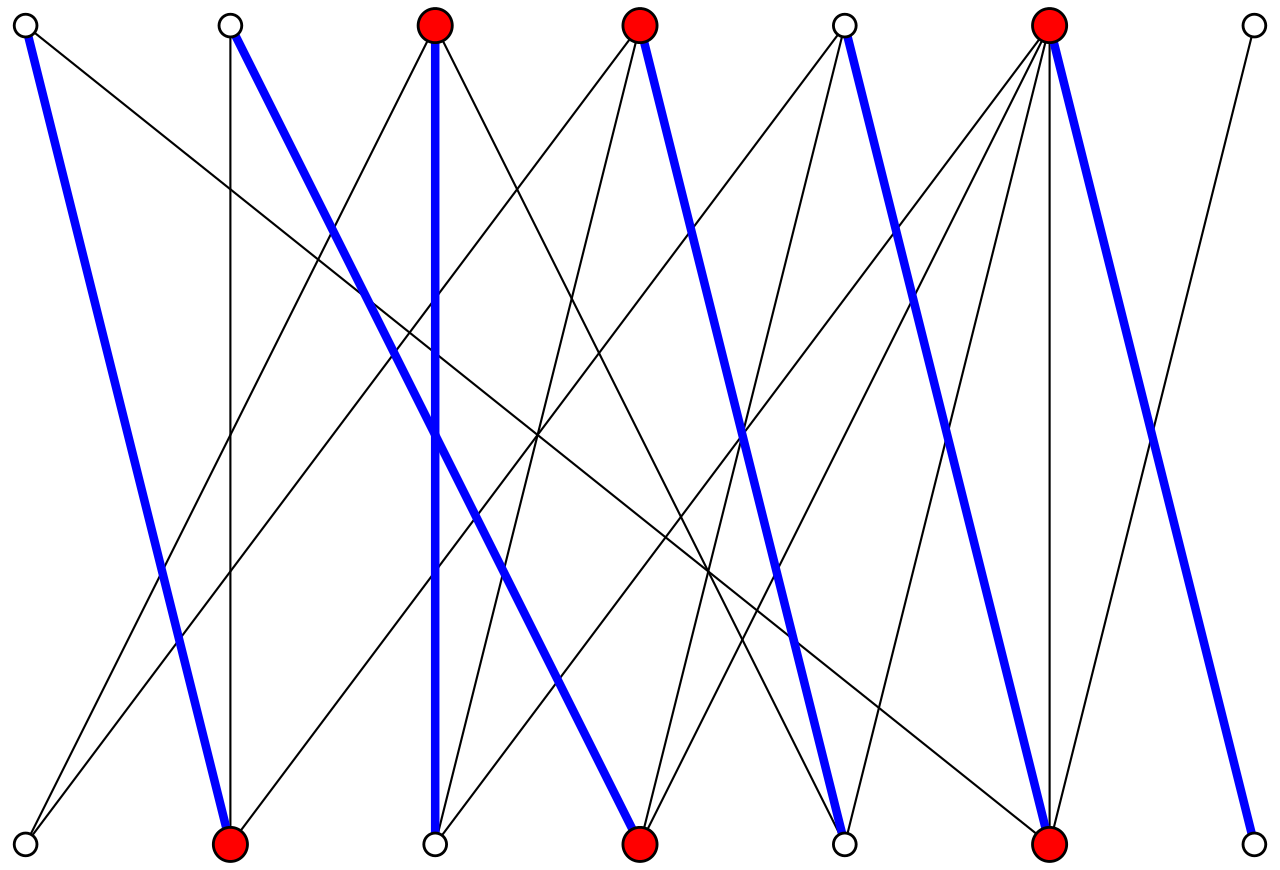
**Bipartite graph**

A **bipartite graph** (or **bigraph**) is a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) whose [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) can be divided into two [disjoint](https://en.wikipedia.org/wiki/Disjoint_sets) and [independent sets](https://en.wikipedia.org/wiki/Independent_set_(graph_theory)) ***U*** �and ***V***�, that is every [edge](https://en.wikipedia.org/wiki/Edge_(graph_theory)) connects a [vertex](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) ***U*** in �to one in ***V***.�

Graphical user interface, application

Description automatically generated

In bipartite graphs, the size of [minimum vertex cover](https://en.wikipedia.org/wiki/Minimum_vertex_cover) is equal to the size of the [maximum matching](https://en.wikipedia.org/wiki/Maximum_matching); this is [Kőnig's theorem](https://en.wikipedia.org/wiki/K%C5%91nig%27s_theorem_(graph_theory)).



**Hopcroft-Karp algorithm**

The **Hopcroft–Karp algorithm** (sometimes more accurately called the **Hopcroft–Karp–Karzanov algorithm**)[[1]](https://en.wikipedia.org/wiki/Hopcroft%E2%80%93Karp_algorithm#cite_note-1) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) that takes a [bipartite graph](https://en.wikipedia.org/wiki/Bipartite_graph) as input and produces a [maximum-cardinality matching](https://en.wikipedia.org/wiki/Maximum-cardinality_matching) as output.

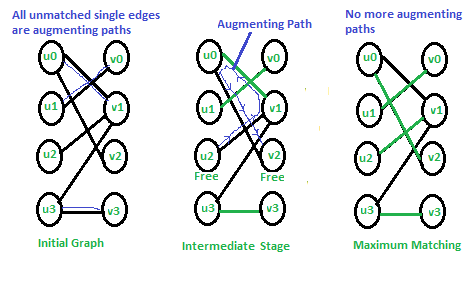
Let us define few terms before we discuss the algorithm

* ***Free Node or Vertex:*** Given a matching M, a node that is not part of matching is called free node.
* ***Matching and Not-Matching edges:*** Given a matching M, edges that are part of matching are called Matching edges and edges that are not part of M (or connect free nodes) are called Not-Matching edges.
* ***Alternating Paths:*** Given a matching M, an alternating path is a path in which the edges belong alternatively to the matching and not matching. All single edges paths are alternating paths.
* ***Augmenting path:*** Given a matching M, an augmenting path is an alternating path that starts from and ends on free vertices. All single edge paths that start and end with free vertices are augmenting paths. In below diagram, augmenting paths are highlighted with blue color. Note that the augmenting path always has one extra matching edge. The Hopcroft Karp algorithm is based on below concept. *A matching M is not maximum if there exists an augmenting path. It is also true other way, i.e, a matching is maximum if no augmenting path exists.* So the idea is to one by one look for augmenting paths. And add the found paths to current matching.

**Hopcroft Karp Algorithm:**

Text, letter

Description automatically generated



There are few important things to note before we start implementation.

1. We need to **find an augmenting path** (A path that alternates between matching and not matching edges, and has free vertices as starting and ending points).
2. Once we find alternating path, we need to **add the found path to existing Matching**. Here adding path means, making previous matching edges on this path as not-matching and previous not-matching edges as matching.

The idea is to use **BFS (Breadth First Search)** to find augmenting paths. Since BFS traverses level by level, it is used to divide the graph in layers of matching and not matching edges. A dummy vertex NIL is added that is connected to all vertices on left side and all vertices on right side. Following arrays are used to find augmenting path. Distance to NIL is initialized as INF (infinite). If we start from dummy vertex and come back to it using alternating path of distinct vertices, then there is an augmenting path.

1. *pairU[]:* An array of size m+1 where m is number of vertices on left side of Bipartite Graph. pairU[u] stores pair of u on right side if u is matched and NIL otherwise.
2. pairV[]: An array of size n+1 where n is number of vertices on right side of Bipartite Graph. pairV[v] stores pair of v on left side if v is matched and NIL otherwise.
3. *dist[]*: An array of size m+1 where m is number of vertices on left side of Bipartite Graph. dist[u] is initialized as 0 if u is not matching and INF (infinite) otherwise. dist[] of NIL is also initialized as INF

Once an augmenting path is found, **DFS (Depth First Search)** is used to add augmenting paths to current matching. *DFS* simply follows the distance array setup by *BFS*. It fills values in *pairU[u]* and *pairV[v]* if *v* is next to *u* in BFS.