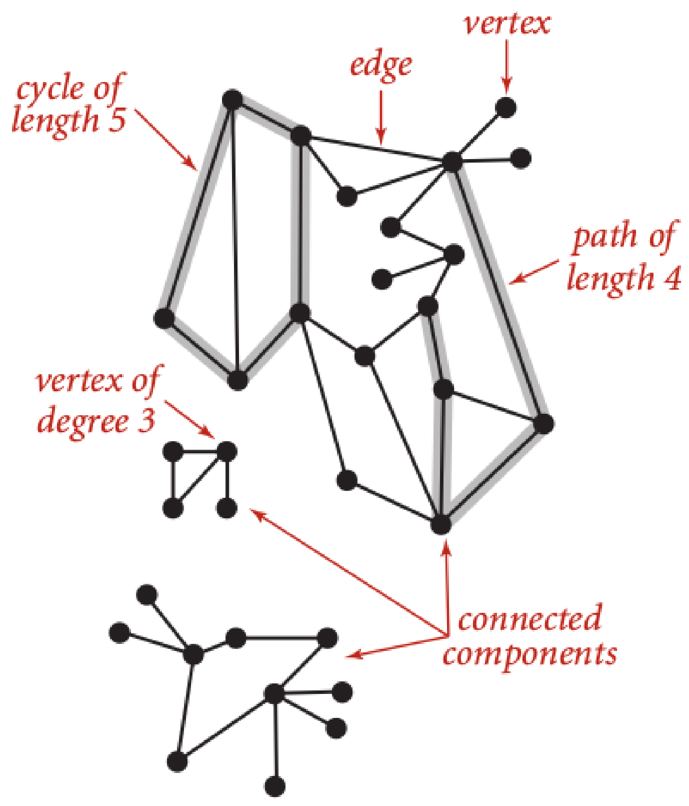
4. Graphs

Graphs is abstract mathematical objects to model complex problems into simple representation. Graph theory became a major branch of mathematics that has been studied intensively for hundreds of years since Euler introduced this idea to solve “seven bridge of konigsberg” at 1735. There are four important type of graphs:

1. Graphs that have simple connections.
2. Digraphs that the direction of each connection is significant.
3. Edge-wighted graphs that each connection has an associated weight.
4. Edge-weighted digraphs that each connection has both direction and a weight.

4.1 Undirected Graph

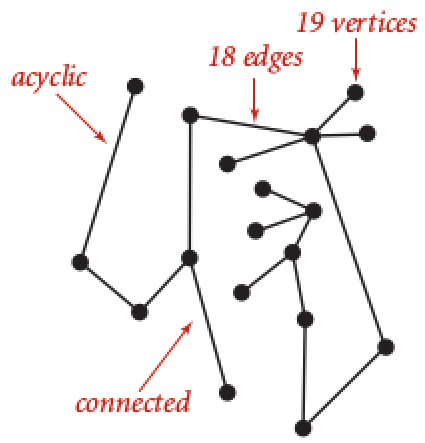
Undirected graph is simplest model contains set of **vertices** and a collection of **edges** that each connect a pair of vertices. Each vertex has **degree** which contains how many adjacent vertices connected to it or also called as neighbors. A graph may has many **path** that sequence of vertices connected by edges. A path called as simple path if has no repeated vertices.



A graph my has **cycle** that a path has a same vertex at first at last. A cycle called as **simple cycle** if has no repeated edges or vertices between first an last.

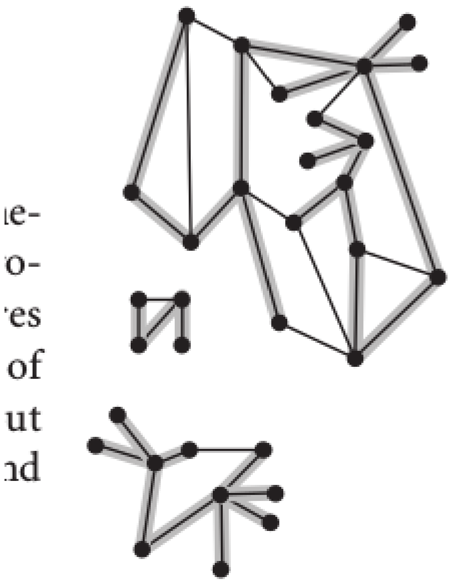
The length of a path and cycle defined by number of edges.

We called a graph is **connected** if every vertex connected. Furthermore, a set of connected graph called as set of **connected components**.



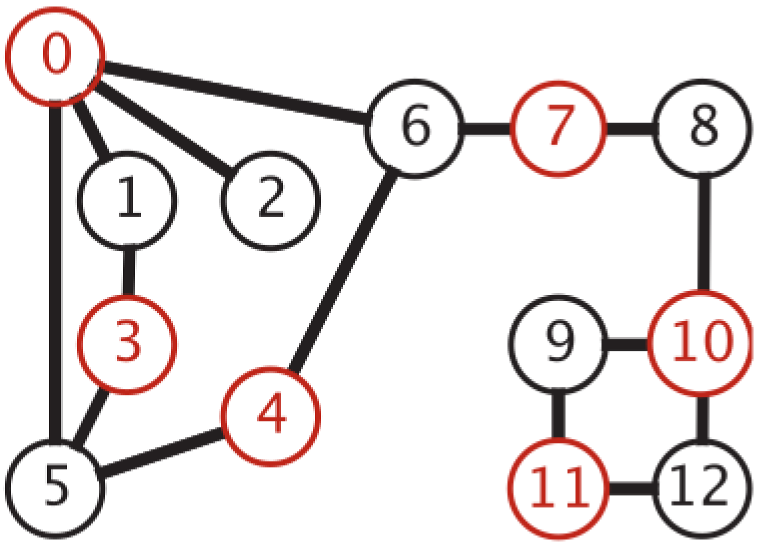
A tree

A graph called as **acyclic** if has no cycles. If we can found an acyclic graph as subgraph of connected graph, then we called that as **tree**. A disjoint set of trees is called as **forest**. A single tree called as **spanning tree** if contains all vertices in the graph. A **spanning tree** is union of spanning trees of its connected components.



A spanning forest

A graph called as **bipartite graph** if we can divide it into two set of graph.

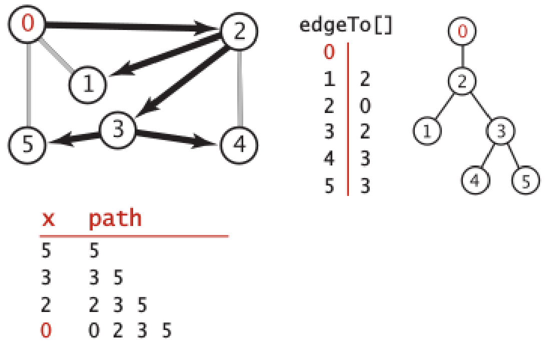


A bipartite graph

Depth-First Search

DFS is oldest graph processing algorithm. Furthermore, we can say that DFS is representation of the nature of the recursion. Image below represent that the trace of recursive process can be drawn as a binary tree.

```
private void dfs(Graph G, int v)
{
   marked[v] = true;
   for (int w : G.adj(v))
      if (!marked[w])
      {
         edgeTo[w] = v;
         dfs(G, w);
      }
}
```



```
edgeTo[]
0 |
1 | 2
2 | 0
3 | 2
4 | 3
5 | 3
```

```
x   path
5   5
3   3 5
2   2 3 5
0   0 2 3 5
```

Each call will process current vertex at its first neighbor until we can not found a next first neighbor, then going up to explore next first neighbor of upper vertex. That is, we use stack to store each call step.

DFS can be used to solve **single source connectivity** problem and **single source path**.

Breath-First Search

BFS implementing Queue to store each call process, so we can iterate to all neighbor of current vertex first, then go to iterate all neighbors of next vertex. Intuitively, the method implemented by BFS is likely same as the way we try to learn best path at unknown places.

BFS can be sued to solve **single source shortest path.**

4.2 Directed Graph

A directed graph (also called as digraph) is set of vertices that each vertices may be connected with directed edge, so the connection may became ordered defined by arrow of edges.

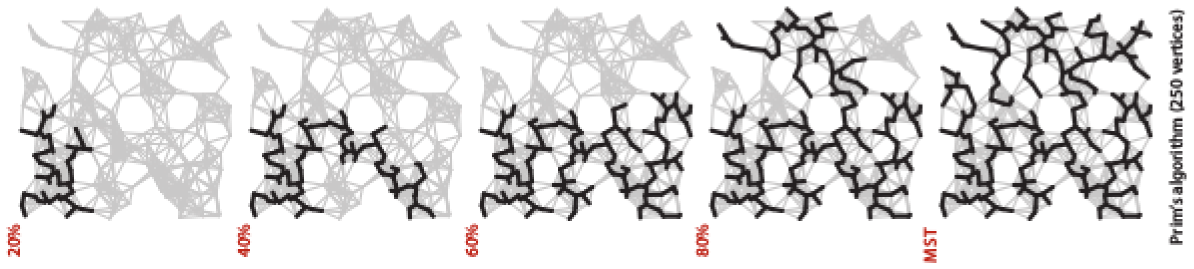
Let **v** → **w** is one of vertex in digraph, then we say that **v is tail** or parent and **w is head** or child. Remember, that a vertex v may has many childs. The **outdegree** is how many child that vertex v has, while **indegree** is how many child that vertex v has.

There are many application of digraph, some of them are **topological sort** for scheduling purpose using **Depth First Order algorithm**, finding **strong connected components** for identify social network or electronic circuit using **kosajaru algorithm**. Another interesting application is building the **WordNet** which is rooted DAG (Directed Acyclic Graph). We can determine semantic relationship between two words or more by implementing **Breath-First** to find **Shortest Ancestral Path** and **Ancestral Parent**.

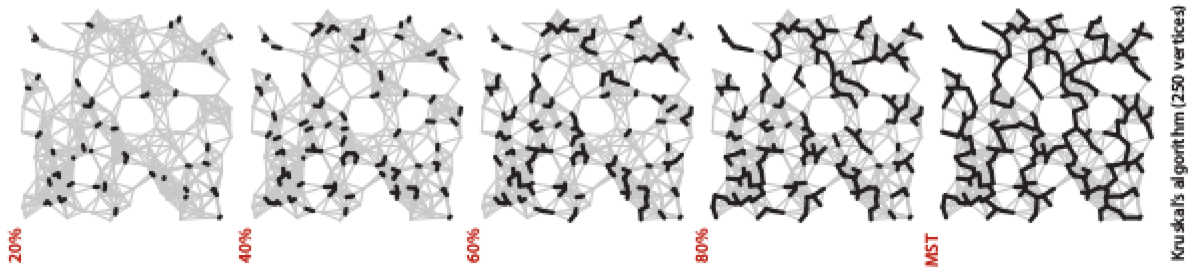
4.3 Minimum Spanning Tree

Given an **edge-weighted graph**, **spanning tree** is a path that connected all vertex which is some of vertices could be grouped as acyclic subgraph. So then, **minimum spanning tree** (MST) is a spanning tree whose has lowest sum of weight of its edge.

There are two classical and most used algorithms to find minimum spanning tree:



Prim's algorithm is an algorithm work like 'growing tree' to construct MST. This algorithm used Priority Queue to promoting next edge. Meanwhile, there are two approach that we can used to maintain priority queue; **Lazy approach** which still keeping ineligible edges in priority queue (space = E, time = E log E) and **Eager approach** which is keeping only eligible edges in priority queue (space = V, time = E log V).



Kruskal algorithm is an algorithm that work in 'ordered manner' to construct MST. This algorithm needed preprocessing that sorted list of edges, so then in processing we can easily choose best edge first until find longest possible edge (space = E, time = E log E).