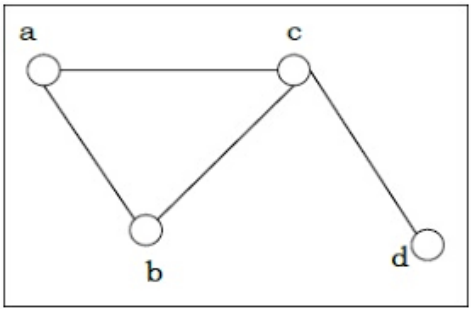
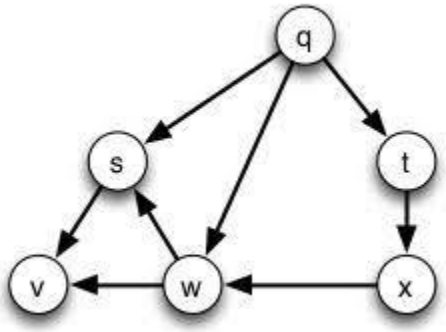
Graph

A graph, **G**, consists of a set of vertices (nodes), **V**, along with a set of edges (connections), **E.**



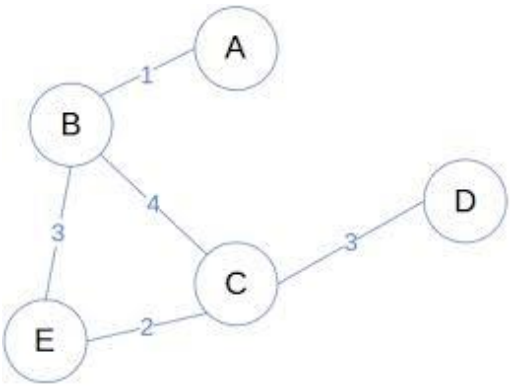
In this graph, **V**={a, b, c, d}, and **E**={{a, b}, {a, c}, {b, c}, {c,d}}.

Digraph (Directed Graph)

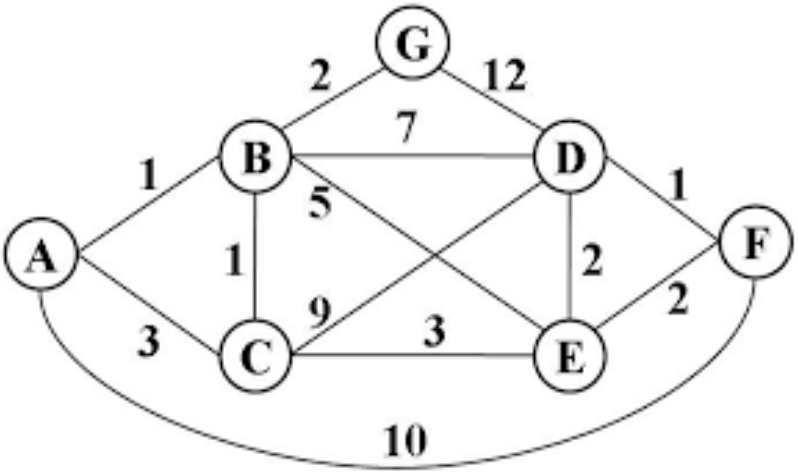


V = {q, s, t, v, w, x}, and **E** = {(q, s), (q, t), (q, w), (s, v), (t, x), (w, s), (w, v), (x, w)}.

Weighted Graphs



Shortest Paths and Dijkstra's Algorithm



Goal: Find the shortest path from A to D

Dijkstra's Algorithm gives us an efficient method to find shortest paths from a given start vertex to every other vertex.

In this algorithm, we divide the vertex set **V** into two subsets: the labeled vertices, **L**, and the unlabeled vertices.

A label is an ordered pair that consists of the total length of the shortest path to that vertex from the starting vertex, and the previous vertex.

When the algorithm begins, the set **L** consists of only the starting vertex, and **U** is all the rest of the vertices.

At every stage, we take one or more vertices from **U** and put them into **L.** We always take vertices that are in **U** but connected to vertices in **L** with an edge. In particular, we pick only those vertices in **U**, connected to a vertex in **L** and of all such vertices, they have the shortest total distance from the starting vertex.

The algorithm ends when we either run out of vertices, or the destination vertex gets put into the **L** set.

Once you finish, follow the labels backwards from the end to the start.

How can we apply the same ideas to exploring mazes?

Two main ideas:

1. Break up the maze states into 2 sets. At first, one of the set consists only of the starting point. Slowly add more of the states to that first set until you get to your ending point.

2. Use the labels at each point to find your way backwards from the end to the start.