BACKGROUND

A graph consists of a set of nodes, also known as vertices, (n_1, n_2, n_3, ...) and a set of edges (e_1, e_2, e_3, ...) where an edge connects two nodes that are in the graph. The node n_1 has children nodes if there exists an edge from n_1 to each child node. In Figure 1, Node "a" has children nodes "b" and "c".

There are two types of edges: directed and undirected. If the edge is directed, then the edge has a specific direction going from start to destination node. Graphs with directed edges are called directed graphs (or **digraph**).

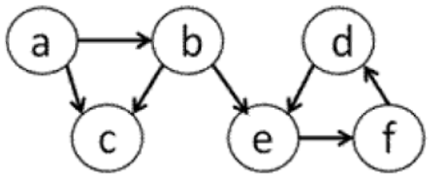


*Figure 1. Example of a **directed** graph where each edge has a specific direction.*

If the edge is undirected, also known as bidirectional, then it no longer matters which node is the start or destination node because you can traverse the edge from one node to the other in either direction. Essentially, a link in the graph can be represented by a directed edge going from Node "d" to Node "e" and a directed edge going in the reverse direction.

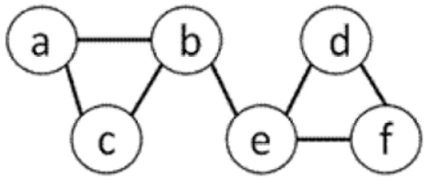


*Figure 2. Example of an **undirected** graph where each edge is bidirectional.*

An edge can also have a weight. If every edge is associated with a real number (edge weight), then we have weighted graph.

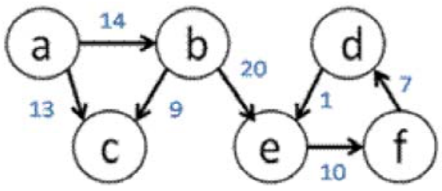
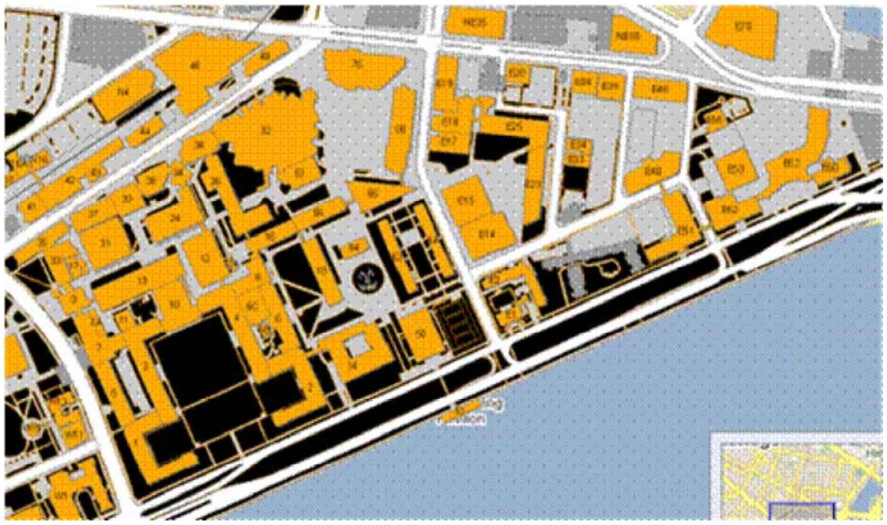


*Figure 3. Example of an **weighted** graph where each edge has a weight associated with it.*

In a graph theory problem, the **objective function** is the function to be minimized (or maximized). For example, choosing the shortest path for airplane flights is an optimization problem where the objective function is to minimize the distance traveled. The nodes are the destination airports and edges are the presence of airplane routes between airports. We can add additional **constraints** on the problem that must be satisfied such as requiring that the plane only make at most 2 stops along the way from start to end destination. Then the shortest path is only valid if it satisfies the constraint.

THE MIT MAP



Here is the map of the MIT campus. From the text input file, `mit_map.txt`, you will build a representation of this map in Python using the graph-related data structures that we provide.

Each line in `mit_map.txt` has 4 pieces of data in it in the following order separated by a single space (space-delimited): the start building, the destination building, the distance in meters between the two buildings, and the distance in meters between the two buildings that must be spent outdoors. For example, suppose the map text file contained the following line:

```
10      32      200      40
```

This means that the map contains an edge from building 10 (start location) to building 32 (end location) that is 200 meters long, where 40 of those 200 meters are spent outside.

To make the problem interesting, we will say that not every route between a pair of buildings is bi-directional. For example, it may be possible to get from building 54 (Green building) to building 56, but not the other way around, because the wind that blows away from the Green building is too strong.