To specify an interface we do not need to know how the programmer will implement the structure underneath. We only need to specify the methods that must be available for the user of the graph. Here are the typical operations for a graph:

Graphical user interface

Description automatically generated with low confidence

Notice that the interface doesn't specify which of the four graph types this interface defines. That is because the types all involve how the operations behave, not which operations exist. Here is the hierarchy for the four graph types:

Diagram, timeline

Description automatically generated

Since the only time we need to be aware of which type of graph we are working with is during adds. It is during adds that we must determine if an edge can go in (does it violate the rules. For example if add\_edge(x,y) is invoked and the graph is undirected, then nothing should happen if an edge already exists (cycles are not a consideration because, by definition, if a-->b, then b-->a which is a cycle). If it doesn't, two entries must be made into the sparce matrix, indicating the edge exists between x-->y and y-->x. On the other hand, if add\_edge(x,y) is invoked on a directed graph, then only one entry is updated: x-->y. In a cyclic graph, the add-edge(x,y) would need to determine if there exists a path already, from y-->x. If there is then the edge x-->y cannot be added as it would produce a cycle. No such requirement is placed on the cyclic graph so the edge could go in as needed.