**Graph ADT**

A graph is a set of objects, called vertices, together with a collection of pairwise connections between them, called edges.

Directed graph

* edge (u, v) (u and v are vertices(objects)) is directed from u to v if the pair (u and v) are in order, meaning u is before v
* a directed graph, also called a digraph, is a graph whose edges are all directed.
* The outgoing edges of a vertex are the directed edges whose origin is that vertex.
* The incoming edges of a vertex are the directed edges whose destination is that vertex.
* If an edge is directed, its first endpoint is its origin and the other is the destination of the edge

Undirected graph

* edge (u, v) is undirected if the pair (u and v are unordered)
* we show this with notation that in the undirected case (u,v) is the same as (v,u)
* If all the edges in a graph are undirected, then we say the graph is an undirected graph.
* Two vertices joined by an edge are end vertices
* Adjacent vertices are vertices that have an edge connecting them
* If a vertex is an edges end point, then the edge is incident to a vertex
* Indegree of a vertex are the number of incoming edges of the vertex
* Out-degree of a vertex are the number of outgoing edges of the vertex
* A path is a sequenced of alternating vertices and edges that start and end at a vertex
* A cycle is a path that starts and ends at the same vertex
* A cycle is simple when a every vertex in the cycle is unique, except for the first and last one since they are the same
* Forest is a graph without cycles
* A tree is a connected forest, that is, a connected graph without cycles
* A spanning tree of a graph is a spanning subgraph that is a tree.
* A directed path is a path such that all edges are directed and are traversed along their direction.
* V reaches U if the path from V to U is direct
* In an undirected graph, reachability is symmetric, meaning is V reaches U that mean U reaches V
* A graph is connected if, for any two vertices, there is a path between them.
* A directed graph G is strongly connected if for any two vertices u and v of G, u reaches v and v reaches u.
* A subgraph of a graph G is a graph H whose vertices and edges are subsets of the vertices and edges of G, respectively, (a graph within a graph)
* A spanning subgraph of G is a subgraph of G that contains all the vertices of the graph G (a subgraph that is the only subgraph)

Data structures for graphs

* Purpose is to maintain a collection to store the vertices of a graph
* Each representation is different and can affect the graph implementation

1. In an edge list, we maintain an unordered list of all edges. This is the simplest way, but there is no efficient way to locate a particular edge (u,v), or the set of all edge’s incident to a vertex v.
2. In an adjacency list, for each vertex, we maintain a separate list containing those edges that are incident to the vertex. This organization allows us to more efficiently find all edges incident to a given vertex.
3. An adjacency map is similar to an adjacency list, but the secondary container of all edges incident to a vertex is organized as a map, rather than as a list, with the adjacent vertex serving as a key. This allows more efficient access to a specific edge (u,v), for example, in O(1) expected time with hashing.
4. An adjacency matrix provides worst-case O(1) access to a specific edge (u,v) by maintaining an n × n matrix, for a graph with n vertices. Each slot is dedicated to storing a reference to the edge (u,v) for a particular pair of vertices u and v; if no such edge exists, the slot will store null.