Reachability/connections/subgraphs

- given vertices u and v of a directed graph G, we say that u <u>reaches</u> v, and v is <u>reachable</u> from u, if G has a directed path from u to v
- in an undirected graph, the notion of <u>reachability</u> is symmetric, u reaches v if ~~only~~ and only if v reaches u
- a graph is <u>connected</u> if for any 2 vertices, there is a path btwn. them
- a directed graph is <u>strongly connected</u> if for any 2 vertices u and v of $\vec{G}$, u reaches v and v reaches u

- a <u>subgraph</u> of a graph G is a graph H whose vertices and edges of G
- a <u>spanning subgraph</u> of G is a subgraph of G that contains all vertices of graph G
- if a graph G is not connected its maximal connected subgraphs are called <u>connected components</u> of G
- a <u>forest</u> is a graph without cycles
- a <u>tree</u> is a connected forest (a connected graph w/o cycles)
- a <u>spanning tree</u> of a graph is a spanning subgraph that is a tree

Data Structures for Graphs

- edge list: we maintain an unordered list of all edges. Minimally suffices but there is no efficient way to locate a particular edge (u,v), or the set of all edges incident to a vertex v
- adjacency list: maintain, for each vertex, a separate list containing those edges which are incident to the vertex
- adjacency map: very similar to adjacency list but secondary container of all edges is organized as a map rather than list with adjacent vertex serving as key
- adjacency matrix: provides worst-case access to a specific edge (u,v) by maintaining an n×n matrix, for a graph with n vertices

Edge List Structure

- Simplest representation of a graph G. Vertex objects are stored in unordered list V, all edge objects are stored in an unordered list E.