Other Graph Algorithms You've Seen

Finding a Topological Ordering (for a DAG)

Finding the SCCs (of a directed graph)
And finding the "condensation" graph (G^{SCC}) – we'll talk about that in a moment.

Finding the connected components (of an undirected graph)

Shortest Paths
Dijkstra's handles weighted graphs, if all weights are positive.

Minimum Spanning Tree
Prim's and Kruskal's both work

Full list on the webpage (look under resources, the list of common DS & algs).

Graph Modeling Process

1. What are your fundamental objects?
Those will probably become your vertices.

2. How are those objects related?
Represent those relationships with edges.

3. How is what I'm looking for encoded in the graph?
Do I need a path from s to t? The shortest path from s to t? A minimum spanning tree? Something else?

4. Do I know how to find what I'm looking for?
Then run that algorithm/combination of algorithms
Otherwise go back to step 1 and try again.

Scenario #1

You've made a new social networking app, Convrs. Users on Convrs can have "asymmetric" following (I can follow you, without you following me). You decide to allow people to form multi-user direct messages, but only if people are probably in similar social circles (to avoid spamming).

You'll allow a messaging channel to form only if for every pair of users a,b in the channel: a must follow b or follow someone who follows b or follow someone who follows someone who follows b, or …
And the same for b to a.

You'd like to be able to quickly check for any new proposed channel whether it meets this condition.

What are the vertices?

What are the edges?

What are we looking for?

What do we run?

Scenario #2

Sports fans often use the "transitive law" to predict sports outcomes -- . In general, if you think A is better than B, and B is also better than C, then you expect that A is better than C.

Teams don't all play each other – from data of games that have been played, determine if the "transitive law" is realistic, or misleading about at least one outcome.

What are the vertices?

What are the edges?

What are we looking for?

What do we run?