**Articulation Point** a vertex in an undirected graph iff removal disconnects the graph

**Diameter** the largest shortest path between the nodes of a graph

**Directed acyclic graph** (**DAG**) finite directed graph with no directed cycles.

**Simple** a graph without multiple edges or self loops

**Path** sequence of alternating vetches and edges.  
**Cycle** path that starts and end at the same vertex.

**Completed Graph** undirected graph, every pair of vertices is connected by an edge

**Clique** a complete subgraph

**Bipartiteness** vertices can be divided into two disjoint and independent sets

**Connected graph** has all pairs of vertices connected by at least one path (undirected). contains a directed path from *u* to *v* or a directed path from *v* to *u* for every pair of vertices *u, v* (directed)

**Connected component** is a connected subgraph of a unconnected graph.

**Strong component** is maximal strongly connected subgraphs

**Strongly connected (Disconnected, simply strong)**: a directed graph where e(u,v) and e(u,v) exist for all vertices pairs

**Weakly connected**: a directed graph that would be connected if turned into an undirected graph.

**Minimum Spanning Tree** subset of edges of a weighted directed graph that connects all vertices, without any cycles and with the minimum total weight

**Tree** undirected graph in which any two vertices are connected by exactly one path

|  |  |  |
| --- | --- | --- |
|  |  | Is directed graph is strongly connected  Finding connected/strongly connected components  Test bipartiteness  Depth first search (DFS)  Breadth first search (BFS)  Topological ordering in a directed graph  Is an undirected graph is connected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Running time | Algorithm | Running time | prove that (n − 1)! is the worst-case number of topological orderings of a DAG with n nodes.  Induction Step: for n-1, is O((n-2)!)  Then: prove with *Tree* (remove one, the rest is in (n-2)!, add back anywhere and it’s (n-1)\*(n-2)!) |
| Gale Shapley |  | MST Prim | \*\*\* |
| DFS, BFS | \*\*\* | MST | *or* |
|  |  |  |  |

\*adjacency list | \*\*adjacency matrix

|  |  |
| --- | --- |
| Gale Shapley | Intervals |
| function stableMatching {  All m ∈ M and w ∈ W to free  while ∃ free m, m has a w to propose to {  w = first woman on m’s list he has not yet proposed  if w is free  (m, w) become engaged  else some pair (m', w) already exists  if w prefers m to m'  m' becomes free  (m, w) become engaged  else  (m', w) remain engaged }} | R set of intervals in S not considered yet  A set of selected intervals  set R = S and A={}  while R 6={} {  choose an interval i in R where f(i) is smallest  add interval i to A  delete intervals from R incompatible with i (\*)} |