1. Topological sorting
2. Minimal spanning Trees
3. Shortest path algorithms

**Topological sorting:**

* For directed acyclic graph, not possible otherwise.
* Linear ordering of vertices such that every directed edge uv, vertex u comes before v in the ordering.
* There can be more than one topological sorting for a graph
* Order vertices so that edges point from lower order to higher order

Applications:

* Build systems
* Advanced-packaging tool
* Task scheduling
* Pre-req problems

References:

* <https://www.geeksforgeeks.org/topological-sorting/>
* <https://www.youtube.com/watch?v=AfSk24UTFS8> from 42:00
* <https://www.youtube.com/watch?v=HyVI8-nHgEg>
* <https://www.youtube.com/watch?v=ddTC4Zovtbc>
* <https://m.blog.naver.com/PostView.nhn?blogId=ndb796&logNo=221236874984&proxyReferer=https%3A%2F%2Fwww.google.com%2F>
* <https://m.blog.naver.com/occidere/220921661731>

**Shortest Path Algorithm:**

* Family of algorithms designed for solving shortest path problems.
* Shortest path problem => given 2 points A and B find shortest path between them.
* Two main types of shortest path algorithm: 1. single-source 2. All-pairs.
* Main types of algorithms: Bellman-Ford, Dijkstra, Topological sort, Floyd-Warshall, Johnson

Single-Source shortest path algorithm:

* Def’n: Given a graph G, with vertices V, edges E with weight function w(u,v) = Wu,v and a single source vertex s, return shortest paths from s to all other vertices in V.
* Bellman-Ford:
  + Solve single-source problem in general case, where edges can have negative weights and graph is directed. If graph is un-directed must make it directed by including two edges in each direction.
  + Has property that can detect negative weight cycles reachable from the source => no shortest path exist.
  + If no negative weight cycle, then Bellman-Ford returns weight of shortest path along with path itself.
* Dijkstra:
  + Uses Breadth First Search(Not a single source SPA) to solve single source problem.
  + Graph cannot have negative weight edges because of this Dijkstra improves runtime of Bellman-Ford.

All-pairs:

* Def’n: Given a graph G, with vertices V, edges E with weight function w(u,v) = Wu,v. Return shortest path from u to v for all (u,v) in V.
* Floyd-Warshall:
  + Use dynamic programming approach.
  + Can have negative weight edges.
  + Works well with dense graphs
* Johnson’s:
  + Works best for sparse graphs
  + Take advantage of the concept of reweighting and use Dijkstra’s algorithm on many vertices to find shortest path once it has finished reweighting the edges.

Applications:

* Maps ex: google map, kakao map, etc…
* Networks, operations, and logistic research

References:

* <https://brilliant.org/wiki/shortest-path-algorithms/>
* <https://medium.com/basecs/finding-the-shortest-path-with-a-little-help-from-dijkstra-613149fbdc8e>