**Shortest Paths in a nutshell**

Take a graph with weighted edges. These weights represent some cost (time, money, fuel, etc.). Find path from vertex x to vertex y with the lowest combined weight.

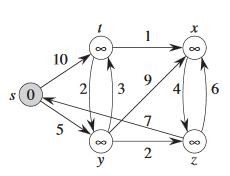
One simple but inefficient way of doing this is by taking all possible paths from x to y and taking the path(s) with lowest combined weight. But again, this is horrendously inefficient.

**Notes on that**

A shortest path will only include edges belonging to a minimum spanning tree of that graph.

**Djikstras algo**

Basically, take a point, and check the weight to adjacent vertices. Pick the shortest one. Check if any other discovered points point to that point. If none, repeat process at that point. If some, check if they have a shorter path to that point than the path you are already taken. If any are shorter, save that and go to that point and repeat. If not, do the same as if there were no discovered points connected to it.

More detailed:

Description of the graph: The marked vertex (the one with 0) is the start vertex (aka u). Other vertices are marked with infinity. Their marks change when reached, into the total cost used to reach it. This can change multiple times if it is reachable by multiple vertices. The numbers on each edge (the arrows) represent the cost for one vertex to reach another.

**Topologically sorted**

Unlike Djikstra, NO FUCKING CYCLES! In fact, you aren’t even allowed to be able to point to vertex that has already pointed at another vertex.