**Graph properties, spanning trees and basic algorithms**

**Graph properties**

**Undirected**: order does not matter, directions go both ways {a, b} and {b, a} is the same

**Directed:** Order matters. An edge e, needs an ordered pair of vertices, and it can only traverse one way. Example edge x has the vertice set of (a, b). (b, a) will not wor

**Incident**: an edge and what vertecies it connects to

**Adjacent**: what other verticies a vertex is connected to by edges

**Isolated**: a single vertecie with no edges

**Open walk:** From x vertex to y vertex

**Closed walk:** From x vertex to same x vertex

**Trails:** Walk without any repeated edges

**Closed trails:** circuit, a trail with same rules as a closed walk

**Path:** a walk with no repeated vertices

**Closed path:** Cycle. End at same vertex as you start on

**Connected:** You can get to all vertices from any given vertex

**Spanning trees**

A spanning tree is a subset of a Graph , which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected. It does not care about the weight of an edge

**Minimum spanning tree**

In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight than all other spanning trees of the same graph. In real-world situations, this weight can be measured as distance, congestion, traffic load or any arbitrary value denoted to the edges.

**Basic algorithms**

**Prims**

to find the minimum cost spanning tree using a greedy approach. We build our minimum spanning tree from a random node. We then always choose the edge with the least weight, and build or tree like this. While we select the lowest weighted edge, we must make sure that our tree never becomes a cycle.

**Kruscals**

to find the minimum cost spanning tree using a greedy approach. First we remove all loops and parallel edges. Next we create a set of edges and their weight (could be distance) and arrange them in ascending order of the weight. Now we start building the minimum spanning tree of our original graph. We build it from the lower weight of edges first:

 