# Minimum Spanning Trees

A ***spanning tree*** in an undirected graph is a set of edges with no cycles that connects all nodes.

## ***Kruskal’s Algorithm:***

Remove all edges from the graph.

Repeatedly find the cheapest edge that doesn’t create a cycle and add it back.

The result is an MST of the overall graph.

# Maintaining Connectivity

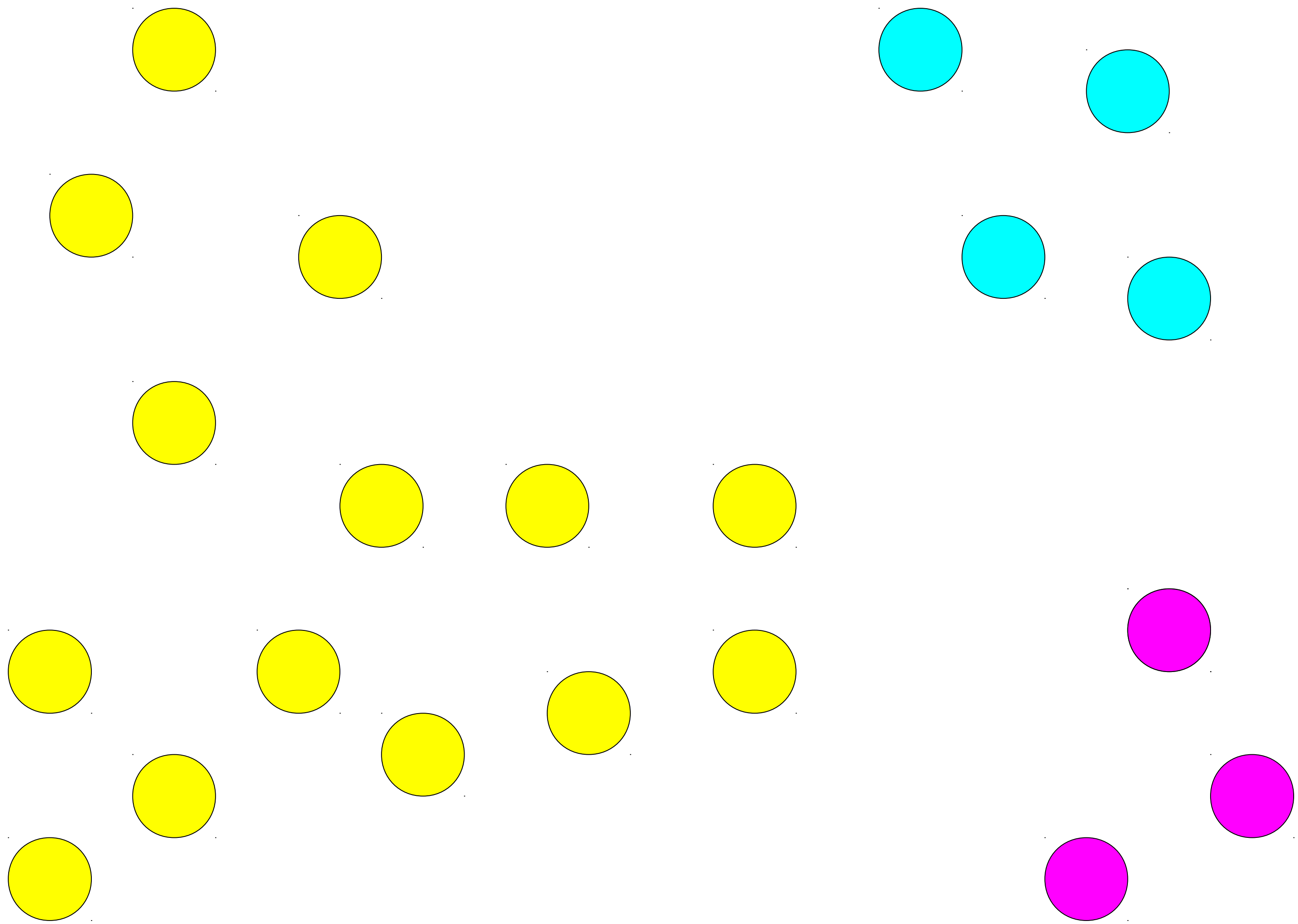
- The key step in Kruskal's algorithm is determining whether the two endpoints of an edge are already connected to one another.
- Typical approach: break the nodes apart into ***clusters***.
  - Initially, each node is in its own cluster.
  - Whenever an edge is added, the clusters for the endpoints are merged together into a new cluster.

# Implementing Kruskal's Algorithm

- Place every node into its own cluster.
- Place all edges into a priority queue.
- While there are two or more clusters remaining:
  - Dequeue an edge from the priority queue.
  - If its endpoints are not in the same cluster:
    - Merge the clusters containing the endpoints.
    - Add the edge to the resulting spanning tree.
- Return the resulting spanning tree.

# Applications of Kruskal's Algorithm

# Data Clustering



# Maximum-Separation Clustering

- A ***maximum-separation clustering*** is one where the distance between the resulting clusters is as large as possible.
- Specifically, it maximizes the minimum distance between any two points of different clusters.
- Very good on many data sets, though not always ideal.

# Maximum-Separation Clustering

- It is extremely easy to adopt Kruskal's algorithm to produce a maximum-separation set of clusters.
  - Suppose you want $k$ clusters.
  - Given the data set, add an edge from each node to each other node whose length depends on their similarity.
  - Run Kruskal's algorithm until only $k$ clusters remain.
  - The pieces of the graph that have been linked together are $k$ maximally-separated clusters.

# Maximum-Separation Clustering