Clustering

Suppose we have a collection of n points in space and we want to partition them into k clusters.

- One way to think about clusters is to choose k centers in such a way that every point is as close as possible to one of those centers.
- Another way is to choose clusters so that the distance between any two points in different clusters is as great as possible.

There are algorithms for each of these approaches, and for still other approaches to clustering, which is a very important practical problem for many applications. The next algorithm focuses on the second approach.

.

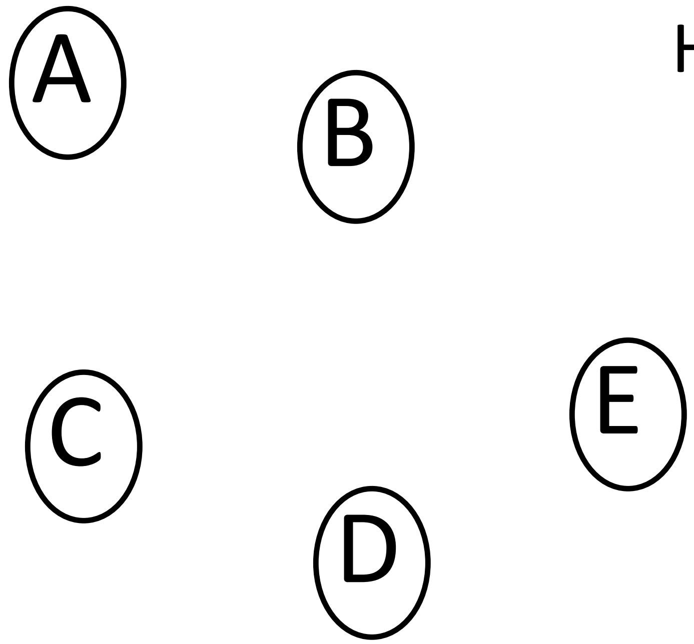
We will formalize this by defining the distance between two different clusters as the shortest distance between two points, one of which is in each cluster. We can define the *spacing* of a set of clusters to be the minimum distance between any two clusters. We want to find a partitioning of our points into k clusters whose spacing is as large as possible

Here is an intuitive algorithm. If points are close together we want them to be in the same cluster so consider pairs of points in order from closest to farthest apart. If the two close-together points are in the same cluster ignore them because they are already grouped together. If they are in different clusters merge the clusters. Continue this until there are k clusters left.

Note that this is exactly what Kruskal's Minimum Spanning Tree algorithm would do on the graph where every point is connected to every other point with an edge whose cost is the distance between the points. We stop the Kruskal algorithm when there are k disjoint sets remaining.

From our analysis of Kruskal's algorithm we know the running time is $O(|E| \log(|V|)) = O(n^2 \log(n))$. This sounds large, but remember that it takes time $O(n^2)$ to just find the distance between each pair of points.

For example, consider the following points:



Here are their distances apart:

B	2			
C	5	6		
D	9	8	3	
E	7	4	8	3
	A	B	C	D

We start with 5 clusters. Each time we choose an edge we reduce the number of clusters by 1. If we want 2 clusters we need to choose 3 edges: the three shortest are A-B, C-D. and D-E. The resulting clusters are {A,B}, and {C,D,E}.

How do we know the clusters this algorithm produces are maximally spaced?

Let d be the cost of the last edge added by Kruskal. The spacing of the Kruskal clusters can't be less than d, for if it was we would have chosen different sets of points to merge. Consider any other way to partition the points into k clusters. That clustering must have some pair of points (p1, p2) in different clusters and the Kruskal algorithm would put in the same cluster. This means that there is a path fromp1 to p2 where each edge costs no more than d. One of these edges must join points in two different clusters in the alternative grouping, so its spacing can't be greater than d. This means that the alternative clustering can't have larger spacing than the Kruskal clustering.