Kruskal's Algorithm and Clustering
(following Kleinberg and Tardos, *Algorithm design*, pp 158–161)

Recall that Kruskal's algorithm for a graph with weighted links gives a minimal spanning tree, i.e., with minimum total weight. This solves, for example, the problem of constructing the lowest cost network connecting a set of sites, where the weight on the link represents the cost.

The minimal spanning tree is also relevant to the clustering problem. Given some notion of similarity between objects, it is frequently useful to group objects into clusters, where clusters contain objects that are in some sense most similar. The objects could be photographs, documents, micro-organisms, ... Given a set of objects $p_1, \ldots, p_n$, a distance function $d(p_i, p_j)$ specifies their similarity (or lack thereof). This function is symmetric: $d(p_i, p_j) = d(p_j, p_i)$, and satisfies $d(p_i, p_j) \geq 0$ (=0 iff $i = j$).

Suppose the n objects are to be separated into k clusters $C_1, \ldots C_k$. The "spacing" of any particular clustering is defined as the minimum distance between objects in any pair of different clusters. One reasonable criterion for a "good" clustering is to find k clusters with maximum spacing. Since the number of possible clusterings grows exponentially with the number of objects, we need an efficient algorithm to find the one with maximum spacing.

Consider growing a graph on the objects p_i considered as vertices. Start by drawing an edge between the closest pair of points, then next closest, etc., and at any given configuration the sets of connected vertices represent the clusters. (This procedure is known as single-link agglomerative clustering.) Note that since only the sets of clusters are of interest, it is not necessary to add any edges that connect vertices already in the same connected component. Hence the graph has no cycles — it is a union of trees.

This graph-growing procedure, though motivated by the idea of merging clusters, is identically Kruskal's algorithm. To produce a k-clustering of the objects, Kruskal's algorithm is simply halted when there are k connected components, and the last $k - 1$ edges are not added. This iterative merging procedure is also equivalent to computing the full minimal spanning tree, then deleting the $k - 1$ most expensive edges and taking the resulting k connected components to define a clustering $\mathcal{C} = \{C_1, \ldots C_k\}$.

To see that this indeed produces k clusters with maximal spacing, note that the spacing of $\mathcal{C} = \{C_1, \ldots C_k\}$ is the weight d^* of the $(k-1)^{\text{st}}$ most expensive edge (i.e., the next edge that would have been added) in the mimimal spanning tree. If $\mathcal{C}' = \{C'_1, \ldots C'_k\}$ is some other clustering, then there is some cluster $C_r \not\subseteq C'_s$ such that there exist $p_i, p_j \in C_r$ with $p_i \in C'_s$ and $p_j \in C'_t \neq C'_s$. Note that each edge on the path from p_i to p_j within C_r has weight $\leq d^*$. Let p' be the first vertex along this path no longer in C'_s and let p be the one just before p' (i.e., still in C'_s). p and p' are in different clusters of $\mathcal{C}'$ but $d(p, p') \leq d^*$, so the spacing of $\mathcal{C}'$ is no greater than that of $\mathcal{C}$. The clustering $\mathcal{C}$ defined by the above procedure thus identifies a k-clustering with maximum spacing.

INFO 295, 5 Oct 06