MAKECLUSTER (Clusters[], PointP1, PointP2)

Input: A list of current Clusters, points p1 and p2 which are used to create or expand on a cluster.

Output: The list of Clusters

```
in1 = searchCluster(clusters, p1)
in2 = searchCluster(clusters, p2)
if in1 == in2:
    pass
elif len(clusters[in1])>1 and len(clusters[in2])>1:
    joinClusters(clusters, in1, in2)
elif len(clusters[in1])==1 and len(clusters[in2])>1:
    joinClusters(clusters, in1, in2)
elif len(clusters[in1])>1 and len(clusters[in2])==1:
    joinClusters(clusters, in2, in2)
elif len(clusters[in1])==1 and len(clusters[in2])==1:
    makeCluster(clusters, in1, in2)
return Clusters
```

Complexity $O(n^2 \log n)$ (for the implemented pseudocode)

Lemma: Suppose your algorithm returns a K-clustering X and there exists an optimal K-clustering OPT. If X differs from OPT by i points, then OPT can be transformed into X without decreasing its spacing.

Proof

Basis

Let $i=0$. Then $X = OPT$ and certainly OPT can be transformed into X.

Hypothesis

Suppose, for all $0 \le i \le K$, OPT can be transformed into X.

Inductive Step

- Let $i = K+1$. Since $K \ge 0$, $i \ge 1$. Then there exists a point p in the cluster C_{opt} under OPT but in cluster C_X under X. Let u be the closest point to p in C_X and v be the closest point to p in C_{opt}.
- Since our algorithm greedily choses to maximize spacing by clustering in increasing order of distance, it must be that the distance between u and p is less than that of v and p.

If p were to be clustered with u in C_{opt}, the distance between C_{opt} and C_X would increase. Thus OPT would not cluster greedily by distance as X had. Thus OPT can be transformed into X without decreasing its spacing.