1. ***Clustering.*** There are different ways to formalize the problem of *clustering*, where the goal is to divide up a collection of "objects" into groups that are similar to one another.

   A natural way to express the input to a clustering problem is via a set of objects $p_1, p_2, \ldots, p_n$ with a numerical distance $d(p_i, p_j)$ defined on each pair. We require only that $d(p_i, p_i) = 0$; that $d(p_i, p_j) > 0$ for distinct p_i and p_j; and that distances are symmetric: $d(p_i, p_j) = d(p_j, p_i)$.

   A reasonable formulation of the clustering problem can be defined as follows: Divide the objects in k sets so as to *maximize* the minimum distance between any pair of objects in distinct clusters. This turn out to be solvable by a nice application of the Minimum Spanning Tree Problem.

   A different but seemingly related way to formalize the clustering problem would be as follows: Divide the objects into k sets so as to *minimize* the maximum distance between any pair of objects in the same cluster. Notice that where the formulation in the previous paragraph sought clusters so that not two were "close" together, this formalization seeks clusters that none of them is too "wide" – that is, no cluster contains two points at a large distance from each other.

   Given the similarities, it's perhaps surprising that this new formulation is computationally hard to solve optimally. Let's write it first as a decision problem, the Clustering problem:

   > Given n objects $p_1, \ldots, p_n$ with distances $d(p_i, p_j)$ for each pair p_i, p_j as above, an integer $k \leq n$, and a bound B,
   >
   > decide whether the objects can be partitioned into k sets, so that no two points in the same set are at distance greater than B from each other?

   Prove that Clustering is NP-complete. Hence, if it would exist a polynomial time algorithm that computes a k-partition of the objects so as to minimize the maximum distance between any pair of objects in the same cluster, then P = NP.

El Clustering problem es NP ya que dado un verificador podemos comprobar si es la solución del problema en tiempo polinómico. El verificador que consiste en un vector con k posiciones diferentes y que cada posición i contiene todos los vértices que pertenecen al cluster i. La verificación consiste en ir de cluster en cluster mirando que la distancia entre cada par de vértices sea menor que B. Esto tiene coste $O(n^2)$.

Reducir de k-coloring a k-cluster (karp)
Dado un grafo G, procedemos a hacer lo siguiente.
Decidimos que B sea igual a |V|,
Para cada par de vértices u v, si (u,v) $\in$ E haremos que la distancia entre ellos en el k-cluster sea mayor a B. En el caso en que no haya arista entre ellos, deben tener distancia menor a B.

De este modo, los vértices adyacentes nunca pertenecerán al mismo cluster y los que no sean adyacentes pueden estar o no en el mismo cluster.

Si existe k-coloring $\Rightarrow$ existe k-cluster :

Si existe k-coloring $\Rightarrow$
$\forall$ u, v vértices del mismo color $\nexists$(u,v) arista que las une $\Rightarrow$
$\forall$ u, v vértices del mismo color, la arista en el grafo nuevo es menor que B y por lo tanto, pueden estar dentro del mismo cluster $\Rightarrow$
La coloración es una posible asignación de clusters válida $\Rightarrow$ existe k-cluster

Si no existe k-coloring $\Rightarrow$ no existe k-cluster
Contrarrecíproco: Si existe k-cluster $\Rightarrow$ existe k-coloring

Si existe un k-cluster $\Rightarrow$
Podemos crear k clusters donde la distancia entre cada par de vértices del mismo cluster es menor a B y la distancia entre vértices de otros clusters puede ser mayor o menor a B.
Si la distancia entre par de vértices es mayor a B, significa que estos vértices eran adyacentes en el k-coloring y por lo tanto, tendrán color diferente, tal como en el k-cluster están en clusters diferentes.
Si la distancia entre par de vértices es menor a B, significa que los vértices no tenían porque tener color diferente y por eso, pueden ser (o no) del mismo cluster.

Con esto hemos demostrado que k-Clustering es NP-Completo.