Nº pareja: 05
Nombres: Pablo Almarza y Carlos García

I-C Questions about Disjoint sets and Connected Components

1. We didn't realize it, but in our algorithm for CCs we are running through a list with repeated edges and with edges in which the vertices are repeated in different order. Will this affect the result of the algorithm? Why?

   No, it will not affect the result of the algorithm. The reason why is that, since we are building a Disjoint Set, the nodes will have a representative; therefore, if two nodes have the same representative, the algorithm will know that they are already connected, so it will not perform any operation.

2. Argue the correctness of the CCs algorithm, that is, argue that at the end of the algorithm the sets that we obtain do indeed represent the connected components of the graph.

   It will represent the components of the graph, because we will iterate all over the list of tuples that represent the edges and we will check if the two nodes of the tuple have the same representative, if so, that means they are already connected and the algorithm will continue to the next element; if not, we perform a union so they will belong to the same set.

3. The size of a graph is determined by the number n of nodes and the length l of the list of edge. Estimate, as a function of both parameters, the cost of the connected components algorithm implemented using disjoint sets. Justify your answer.
   Cost = O(l * log(n)). We perform a loop through all the elements in the list and then we do the union of the elements (if the elements do not have the same representative) so we will have the cost of the union (that is O(log(n)) performed l times.

II-B Questions about the Traveling Salesman Problem

1. Estimate the cost of the greedy algorithm as a function of the number of cities n; justifying your question and show your reasoning. What would be the cost of applying exhaustive_greedy_tsp? How about that of repeated_greedy_tsp?

   For the greedy_tsp, the cost would be $O(n^2)$ because for every node, it checks all the nodes looking for the one with the least cost to travel to.

   In case of the repeaded_greedy_tsp, it would jump up to $O(n^3)$ since we compute greedy_tsp swapping the starter node between all the nodes looking for the least cost circuit.

   The exhaustive_tsp will have n! circuits to check the length of, so we could say the cost is O(n!).

2. Using the code developed in this exercise, find a graph for which the greedy solution is not optimal/

   One solution to this question could be the following:

   0 1 2 9
   1 0 1 2
   2 1 0 1
   9 2 1 0

   In this case, the greedy algorithm takes each step with the least cost accordingly, but when reaching the last step, it's stuck with a high cost step which adds a lot to the total cost.