Algorithmics	Student information	Date	Number of session
	UO: 271407	16/03/2020	4
	Surname: Llera		
	Name: Elías		

Activity 1. Salesman.

For greedy one, the algorithm needs a starting node. For the times taken in this submission, this source node will always be 0 (this won't affect times, since all values are assigned randomly). What this algorithm does is dividing the graph in two sets of nodes: visited and not visited. At the begging of the execution the source node is set as visited. After that, at the end of each iteration a not visited node will be set as visited. To do so, we have to pick the last visited node, and choose from its edges the cheapest one that leads to a not visited node. When the set of not visited nodes is empty, we have to pick the last visited node and add to the result the edge that connects this node to the source.

The complexity for this algorithm is $O(n^2)$, since it has two nested loops to iterate over the nodes and edges. We are able to have this complexity and not a cubic one by using a Boolean array size the number of nodes, that stores in each position true if the node has been visited and false if not. If we didn't use this, in the checking of the node been visited or not, we would have to use something similar to the contains method, which would increment the complexity.

Greedy two could not be implemented. The approach that it should have is picking all the edges in the graph and sorting them by weight in ascending order. Then we iterate over the edges, and if that edge doesn't create any cycle in the resulting graph, we add it to the result. At the end, we pick the last node and join it with the initial one (which was determined by the smallest edge in the graph).

Since this algorithm was not implemented, we cannot check its complexity. However, we can presume that it is going to be at least the same than the previous algorithm.

Algorithmics	Student information	Date	Number of session
	UO: 271407	16/03/2020	4
	Surname: Llera		
	Name: Elías		

Activity 2. Salesman times 1.

The first heuristic gives pretty acceptable results, being close to the optimal one. However, the fact that we have to join the last node with the first one no matter what gives a pretty big error margin, since we can have a scenario in which this last edge is much bigger than all the edges traversed all combined. This is not a common case though, and the average case should be not optimal but good.

The second heuristic, although not checked (since it is not implemented), should give better results, since we are always checking the smallest possible edge to take.

Algorithmics	Student information	Date	Number of session
	UO: 271407	16/03/2020	4
	Surname: Llera		
	Name: Elías		

Activity 3. Salesman times 2.

N	Time
10	0 ms
20	0 ms
40	0 ms
80	18 ms
160	0 ms
320	15 ms
640	69 ms
1280	285 ms
2560	1038 ms
5120	3931 ms
10240	15011 ms
20480	58777 ms

In the graphic we can see the times taken for greedy 1 with nTimes=250. We can see that it takes a while to reach valid times (approximately n = 640). Taking into account that the complexity for this algorithm is square, we have that $t_2=k*t_1$, with $k=n_2^2/ n_1^2$. Assuming, as an example, $n_1=1280$ (and then $t_1=285$ms) and $n_2=5120$, we should have that $t_2 =$ 4560ms. Having a look at the graph we see the real time obtained was 3931ms. Although we have a slight difference (of about 600ms), the times are pretty close so we can assume that the complexity is indeed $O(n^2)$.

CPU: Intel® Core™ i5-3470 CPU @ 3.20GHz

RAM: 8GB