Part C: Improved_Greedy

My idea was inspired by the previous greedy algorithm. I noticed that the greedy algorithm only contained one permutation which always started from the same initial city. I wondered if making the initial city different obtains a better result.

I created a loop outside the original greedy algorithm. There is also a list called “all_travelled” which is a collection of permutations resulted from different initial cities.

After the loops, we calculate the tour value out of those n permutations. The results are kept in a list called “results”.

Then we find the best/smallest value from “results” and return the corresponding permutaiton.

It is clear to see that there are 3 nested for loops that all run for n times. Therefore time complexity of the improved greedy algorithm is O(n^3) for n equals to the total number of cities.

Part D: Experiments

I wrote functions that create both euclidean graphs and general graphs. Then I tested with random generated graphs and obtained the following outputs.

Note: All results in the table are the average of 25 outputs.

Euclidean	Original	Swap	2_Opt	Greedy	Improved Greedy
20	82.13	80.14	**73.43**	81.85	76.13
40	128.14	121.19	**107.85**	126.63	119.24
60	148.9	139.05	**118.86**	154.57	130.56
80	167.07	157.99	**131.74**	152.82	141.41

General	Original	Swap	2-Opt	Greedy	Improved Greedy
4	6.885	6.878	**6.828**	8.242	**6.828**
6	16.532	16.522	**16.243**	20.485	**16.243**
8	30.331	30.321	29.69	36.73	**29.657**
10	48.288	48.278	47.154	56.971	**47.071**

For Euclidean graphs 2-Opt is the best algorithm since for all number of cities they have the shortest route. Improved Greedy is not far behind 2-Opt and is better than all the other algorithm.

For General graphs Improved Greedy is better than or equal to 2-Opt. There are cases where they share the same shortest route value and as number of cities increases Improved Greedy surpasses 2-Opt.