**Nearest Neighbor Algorithm**

<https://en.wikipedia.org/wiki/Nearest_neighbour_algorithm>

In this algorithm, the salesman starts at a random city and repeatedly visits the nearest city until all have been visited. It quickly yields a short tour, but usually not the optimal one.

This algorithm has a greedy nature.

These are the steps of the algorithm:

1. start on an arbitrary vertex as current vertex.
2. find out the shortest edge connecting current vertex and an unvisited vertex V.
3. set current vertex to V.
4. mark V as visited.
5. if all the vertices in domain are visited, then terminate.
6. Go to step 2.

Pseudocode:

V = {1 … n-1} //Vertices except for 0 – not yet visited

U = {0} //Vertex 0 – visited nodes

while V is not empty

x = most recently added vertex to U

Find vertex y in V closest to x

Add y to U and remove y from V

Output vertices in the order that they were added to U

**Repetitive Nearest-Neighbor Algorithm**

<https://www.math.ku.edu/~jmartin/courses/math105-F11/Lectures/chapter6-part4.pdf>

Execute the Nearest-Neighbor Algorithm once for every vertex as the beginning vertex. Then pick the Hamilton circuit with the lowest total weight of the N tours.

Efficient, but may not produce an optimal route.

**Greedy Algorithm**

<http://lcm.csa.iisc.ernet.in/dsa/node186.html>

Based on Kruskal’s algorithm, usually gives a suboptimal solution. Works for complete graphs, might not work for incomplete graphs.

These are the steps of the algorithm:

1. As in Kruskal’s algorithm, first sort the edges in the increasing order of weights

2. Starting with the least cost edge, look at the edges one by one and select an edge only if the edge, together with already selected edges:

a. does not cause a vertex to have degree three or more

b. does not form a cycle, unless the number of selected edges equals the number of vertices