Nearest Neighbor Algorithm

1. Start at certain vertex.
2. Choose nearest unvisited vertex.
3. Go to next nearest unvisited vertex. (Repeat.)
4. Return to start when all vertices are exhausted.

Idea: Create a Hamiltonian Circuit, and so this algorithm should end with wiggly blue edges in a circuit, visiting each vertex only once.

Sorted Edges Algorithm

1. Arrange the edges of a complete graph in order of increasing cost/length.
2. Select the shortest edge and draw a wiggly blue line over that edge.
3. Repeat this process, UNLESS:
   (a) Three (3) used edges meet at a vertex. (Remember, HC uses ONLY 2 edges at each vertex.) or
   (b) You close up a circuit that doesn’t include all vertices.
4. Stop when HC is found.

Idea: Create a Hamiltonian Circuit, and so this algorithm should end with wiggly blue edges in a circuit, visiting each vertex only once.

Kruskal’s Algorithm

1. Arrange the edges of a complete graph in order of increasing cost/length.
2. Add (wiggly) edges to the graph in the order of cheapest cost, unless a circuit is formed.
3. Repeat step 2 until all vertices are included.

Idea: Do NOT create a Hamiltonian Circuit. Here, you are creating a minimum-cost spanning tree which connects all vertices, not forming a circuit. This algorithm should end with wiggly blue edges in a spanning, visiting each vertex only once.

Note: (The total number of wiggly edges)=(total number of vertices in graph)-1.