

# The Traveling Salesman Problem

## Glossary of Terms

**Complete Graph:** A graph in which *every* pair of vertices is joined by exactly *one* edge.

**Hamilton Circuit:** A circuit that visits every *vertex* of a graph exactly once.

**Hamilton Path:** A path that visits every *vertex* of a graph exactly once.

**$K_N$ :** The complete graph on N vertices. Note: there is only one for each  $N=1, 2, 3, \dots$

**Traveling Salesman Problem (TSP):** Given a complete weighted graph, this problem tells you to find an optimal Hamilton circuit. That is, a Hamilton circuit with the least total weight.

**Weighted Graph:** A graph where each *edge* is given a number known as a *weight*.

## Algorithms

### Algorithm 1: The Brute Force Algorithm.

- Make a list of all possible Hamilton circuits for the graph.
- For each Hamilton circuit calculate its *total weight*.
- Find the circuits (there is always more than one) with the least total weight. Any one of these can be chosen as an optimal Hamilton circuit for the graph.

#### Notes:

It is important to remember that  $K_N$  has a total of  $(N-1)!$  Hamilton circuits. That can be a lot!! (Remember you calculate  $(n)!$  as  $(n)*(n-1)*(n-2)* \dots *(3)*(2)*(1)$ . Example:  $5!=5*4*3*2*1=120$ .)

### Algorithm 2: The Nearest Neighbor Algorithm.

- Pick a vertex as the starting point.
- From the starting vertex go to its *nearest neighbor* -the vertex for which the corresponding edge has the smallest weight.
- Continue this process remembering not to revisit any vertices until there is no choice but to go back to your starting vertex to complete the circuit.

#### Notes:

### **Algorithm 3: The Repetitive Nearest Neighbor Algorithm.**

- Apply the Nearest Neighbor Algorithm at each vertex and calculate the total cost for each Hamilton circuit obtained.
- Choose the Hamilton circuit with the lowest total cost.
- If a certain vertex was specified as the starting vertex, rewrite your the circuit obtained in the previous step beginning at the specified vertex.

**Notes:**

### **Algorithm 4: The Cheapest Link Algorithm.**

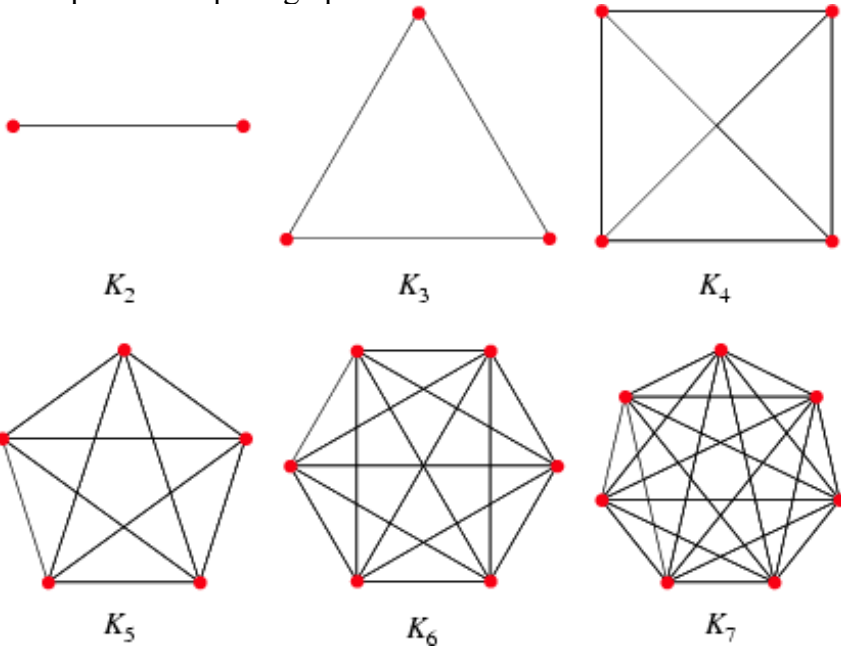
- Pick the edge with the smallest weight first. Mark the edge in red.
- Pick the next "cheapest" edge and mark it also in red.
- Continue picking the "cheapest" edge available and mark it in red except when
  - it closes a circuit.
  - it results in three edges coming out of a single vertex.
- When there are no more vertices to join, close the red circuit and you're done!

**Notes:**

# Examples

---

Example 1: Complete graphs.



Example 2:

---

---

Example 3:

---

Example 4:

---