Remember: The Brute Force Algorithm finds the weight of every Hamilton Circuit and chooses the cheapest one.

Pro: Guaranteed to find the most efficient circuit

Con: Can be a lot of work to carry out the algorithm

(each increase in vertices increases the work by a factor equal to the # of vertices in the graph)

Inefficient algorithm = an algorithm for which the number of steps needed to carry it out grows disproportionally with the size of the problem

<table>
<thead>
<tr>
<th>vertices</th>
<th># of circuits</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>120</td>
</tr>
<tr>
<td>7</td>
<td>720</td>
</tr>
<tr>
<td>8</td>
<td>5040</td>
</tr>
<tr>
<td>9</td>
<td>362880</td>
</tr>
<tr>
<td>10</td>
<td>39916800</td>
</tr>
</tbody>
</table>

Brute Force Algorithm is inefficient
Nearest Neighbor Algorithm - start at home and follow the cheapest choices from each vertex

1. Pick a vertex as the starting point
2. From the starting vertex, go to the vertex for which the corresponding edge has the smallest weight
3. Continue building the circuit, one vertex at a time.
   - Do not go to a vertex that has already been visited
   - Do not go to a vertex if it creates a small circuit
4. From the last vertex, return to the start.

![Graph](image)

Start at vertex A. What is the smallest number from A?

Go to vertex B, again, what is the smallest number from B?

Go to vertex D, what is the smallest number from D? (We can’t go to A)

Only choice is C, and then back to A.

Circuit: ABDCA

Weight: 80
Is this the most efficient route?

Pro: Much less work!

Con: Doesn’t necessarily produce the optimal Hamilton circuit (compromise answer)

Efficient Algorithm = an algorithm in which the number of steps needed to carry it out grows in proportion to the size of the input to the problem

Nearest Neighbor is efficient

Repetitive Nearest Neighbor Algorithm- Builds on Nearest Neighbor, but does the problem for each vertex.

1. Let X be any vertex. Apply the nearest neighbor algorithm using X as the starting vertex and calculate the total cost of the circuit obtained.
2. Repeat the process using each of the other vertices of the graph as the starting vertex.
3. Of the Hamilton circuits obtained, keep the best one.

We’ve already done the Nearest Neighbor starting with vertex A: weight = 80.
Pick a different vertex, say B.

From B, the nearest neighbor is D, then A, and finally C.

Route: BDACB. Is this the same as ACBDA?

Weight: 74

Pick yet another vertex, C.

From C, nearest neighbor is B, then D and then A.

Route: CBDAC, same as ACBDA, weight 74.

Try starting at D:

From D, nearest neighbor is B, then C then A.

Route: DBCAD, which is the same as ADBCA, weight 74.

Best route found had a weight of 74. Is this the optimal route?
Cheapest Link Algorithm

1. Pick the link with the smallest weight first. Mark the corresponding edge.

2. Pick the next cheapest link and mark the corresponding edge (note- This edge does not have to touch the edge already marked.)

3. Continue picking the cheapest link available and marking the corresponding edge except when:
   (a) It closes a circuit
   (b) It results in three edges coming out of a single vertex

4. When there are no more vertices to link, close the marked circuit.

Mark 8 first, then 10.

The next cheapest is 12, but picking AB gives us three lines coming out of B.

The next is 18, and we have now linked all vertices. Close the circuit by connecting A and C.

Route: ADBCA or ACBDA, weight 74.

Is this the optimal route?
Approximate Algorithm- an algorithm that produces solutions that are most of the time reasonably close to the optimal solution.

Note: Last 3 methods are all Approximate Algorithms!

Try Nearest Neighbor, Repetitive Nearest Neighbor and Cheapest Link algorithms on the following examples: