Description:

The algorithm for solving TSP first constructs the MST by using Prim's algorithm. After constructing the MST, the algorithm does DFS traversal of MST. In the process of DFS traversal, the visited vertices are pushed to the list. After DFS traversal, the total cost is calculated from the list including visited vertices of MST, and then the total cost and visited vertices of MST are appended to the output file.

Pseudo Code:

(Assumption: the number of vertices of the graph is n.)

```
I.    For i: 1~n
      A.  Start[i] ← Input File
      B.  End[i] ← Input File
II.   Matrix[i][j] ← CalCost(start[i], start[j], end[i], end[j])
III.  For each vertex, visited[v] ← 0
IV.   Prim's_MST(matrix, n, MST) ; Construct MST by using Prim's Algorithm
V.    Adjacency Matrix ← MST
VI.   DFS
      A.  Index.pushback(start)
      B.  Visited[start] ← 1
      C.  For i: 2 ~ n
          i.  If AdjacencyMatrix[start][i] == 1 && visited[i] == 0
              1.  DFS(AdjacencyMatrix, i, start, visited, index)
VII.  Index.push_back(index[0]) (Making cycle)
VIII. For i: 1 ~ n-1
      A.  Cost += matrix[index[i-1]][index[i]]
IX.   Append Total Cost and Index array to the ".tour" file.
```

Theoretical Running Time:

(Assumption: the number of vertices of the graph is n.)

The program first calculates the distance between each vertex because the graph is complete. The calculated distances go to the distance matrix. The distance matrix needs to be filled the cost between each vertex. Thus, the theoretical running time for filling the distance matrix is $O(n^2)$.

Constructing MST with Prim's algorithm uses distance matrix so that the theoretical running time for constructing MST is $O(n^2)$.

Then the program fills adjacency matrix from the MST. Thus, the theoretical running time for filling adjacency matrix is $O(n^2)$.

Then the program does DFS traversal. The program recursively does DFS traversal for every vertex of the tree. Thus, the theoretical running time of DFS traversal is $O(nlgn)$.

Appending the total cost and index array to the ".tour" file takes $O(n)$ theoretical running time.

Therefore, the total theoretical running time of the problem is $O(n^2)$.
($\because O(n^2 + n^2 + n^2 + nlgn + n) = O(n^2)$)

Approximation Bound:

<Proof>

Let OPT denote the cost of the optimal TSP tour, let MST denote the total length of the minimum spanning tree, and let A be the length of the tour computed by our approximation algorithm. Consider the 'tour' obtained by walking through the minimum spanning tree in depth-first order. Since this tour traverses every edge in the tree exactly twice, its length is 2 · MST. The final tour can be obtained from this one by removing duplicate vertices, moving directly from each node to the next unvisited node.; the triangle inequality implies that taking these shortcuts cannot make the tour longer. Thus, A ≤ 2 · MST. On the other hand, if we remove any edge from the optimal tour, we obtain a spanning tree (in fact a spanning path) of the graph; thus, MST ≥ OPT. We conclude that A ≤ 2 · OPT; our algorithm computes a 2-approximation of the optimal tour.

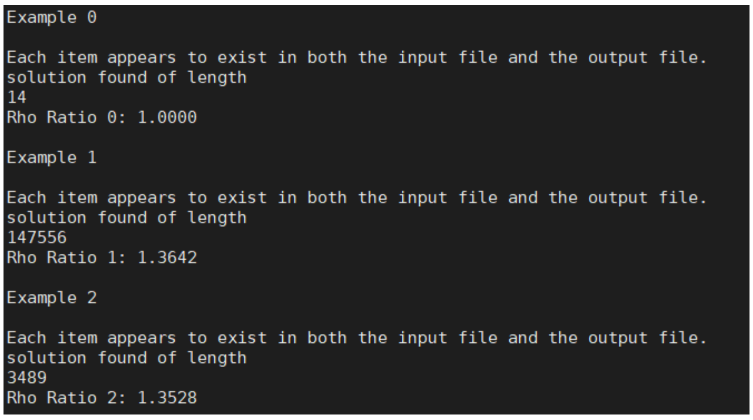
(Proof from http://jeffe.cs.illinois.edu/teaching/algorithms/notes/J-approx.pdf)

Thus, the bound of this approximation algorithm is less than or equal to twice of the optimal solution.

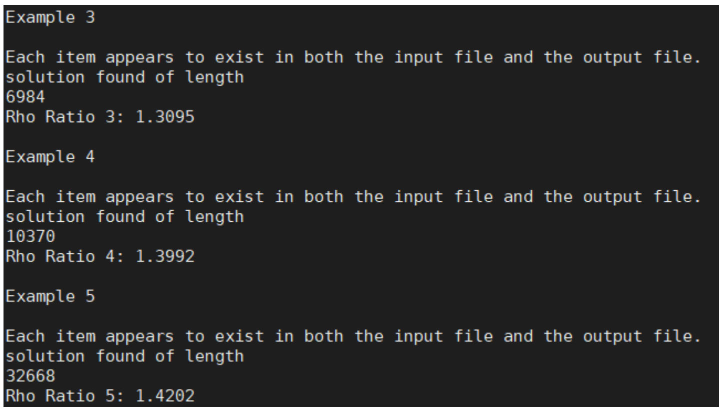
($Approximation\ Algorithm\ \leq 2 * OPT$).

Summary:

TSP Example	Rho
tsp _ example_0	1.0000
tsp _ example_1	1.3642
tsp _ example_2	1.3528
tsp _ example_3	1.3095
tsp _ example_4	1.3992
tsp _ example_5	1.4202



```
Example 0

Each item appears to exist in both the input file and the output file.
solution found of length
14
Rho Ratio 0: 1.0000

Example 1

Each item appears to exist in both the input file and the output file.
solution found of length
147556
Rho Ratio 1: 1.3642

Example 2

Each item appears to exist in both the input file and the output file.
solution found of length
3489
Rho Ratio 2: 1.3528
```



```
Example 3

Each item appears to exist in both the input file and the output file.
solution found of length
6984
Rho Ratio 3: 1.3095

Example 4

Each item appears to exist in both the input file and the output file.
solution found of length
10370
Rho Ratio 4: 1.3992

Example 5

Each item appears to exist in both the input file and the output file.
solution found of length
32668
Rho Ratio 5: 1.4202
```

Research Summary:

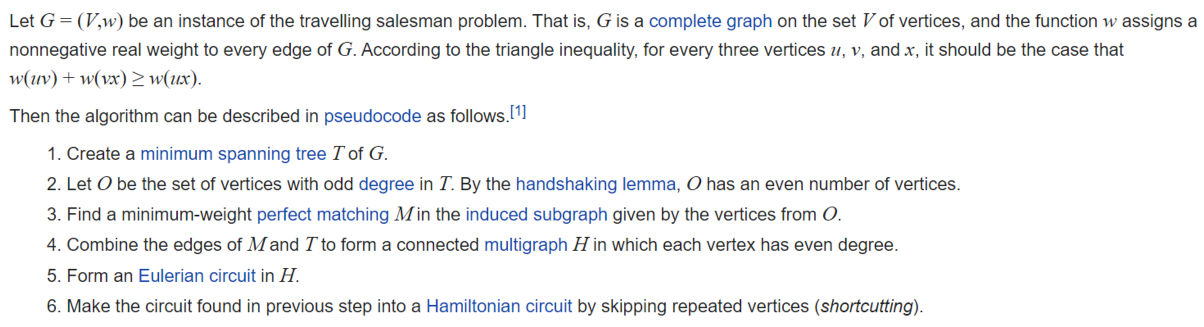
Heuristic

- Nearest Neighbor algorithm:
    - Chooses nearest unvisited cities until all of the cities are visited.
    - However, there is no bound for this algorithm
    - Simple and Fast, but there exist some cities give worst case
- Match Twice and Switch algorithm
    - Performs two sequential matchings where the second matching is executed after deleting all the edges of the first matching, to yield a set of cycles. The cycles are then stitched to produce the final tour

(From: https://en.wikipedia.org/wiki/Travelling_salesman_problem#Heuristic_and_approximation_algorithms)

Approximation Algorithm

- Christofide's Algorithm: 1.5-OPT algorithm



Let $G = (V,w)$ be an instance of the travelling salesman problem. That is, G is a complete graph on the set V of vertices, and the function w assigns a nonnegative real weight to every edge of G. According to the triangle inequality, for every three vertices u, v, and x, it should be the case that $w(uv) + w(vx) \geq w(ux)$.

Then the algorithm can be described in pseudocode as follows.[1]

1. Create a minimum spanning tree T of G.
2. Let O be the set of vertices with odd degree in T. By the handshaking lemma, O has an even number of vertices.
3. Find a minimum-weight perfect matching M in the induced subgraph given by the vertices from O.
4. Combine the edges of M and T to form a connected multigraph H in which each vertex has even degree.
5. Form an Eulerian circuit in H.
6. Make the circuit found in previous step into a Hamiltonian circuit by skipping repeated vertices (*shortcutting*).

(From: https://en.wikipedia.org/wiki/Christofides_algorithm)

- Double Tree algorithm: 2-OPT algorithm
    - Find a minimum spanning tree T.
    - Duplicate the edges of T. Find a Eulerian tour.
    - Shortcut the Eulerian tour.

(From: https://bochang.me/blog/posts/tsp/)