Modelling and Optimization

INF170

#13:Heurisitcs for TSP

AHMAD HEMMATI

Optimization Group
Dept. of Informatics
University of Bergen

Fall Semester
2018

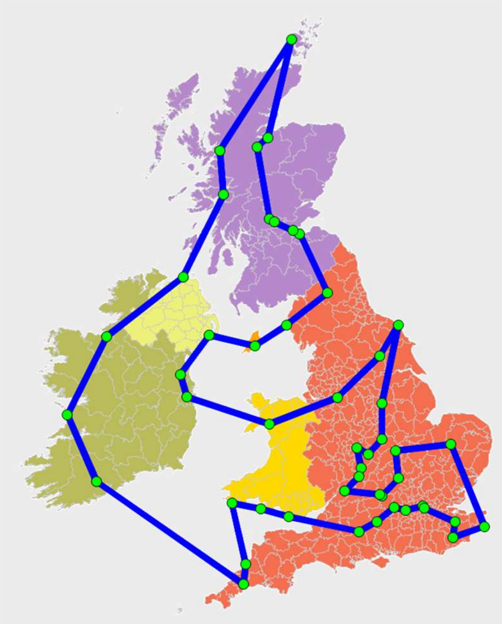
Agenda

- Construction heuristics

- Improvement heuristics (neighborhood search)

Travelling Salesman Problem

- Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

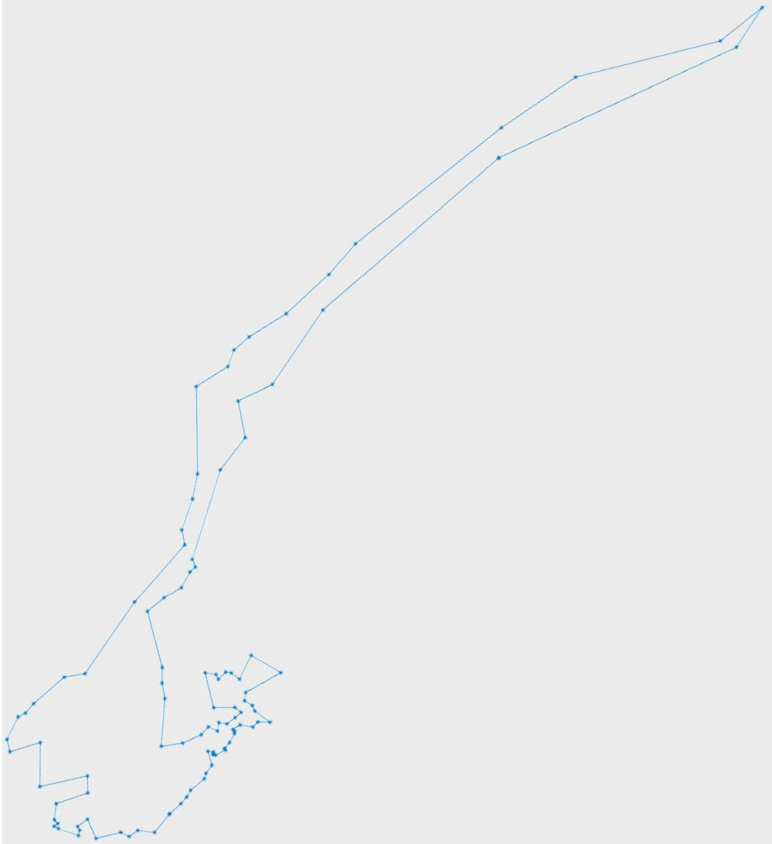


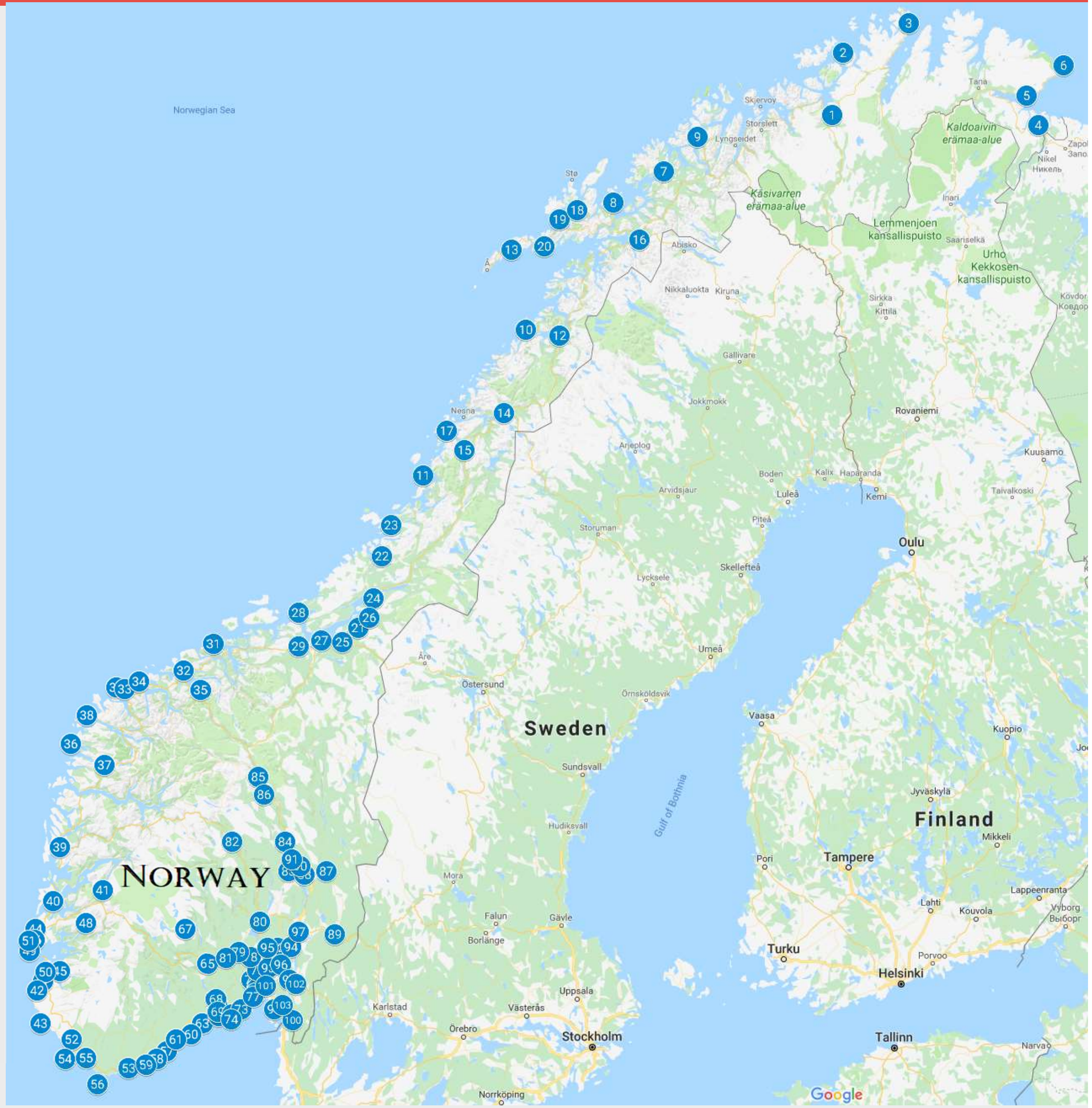
TSP – Implementation

20 nodes:

	MTZ	gain(Svestka)	Steps(Dantzig)
_nvars	400	760	7600
_ncons	420	459	420
Time	0.14	0.23	7.4

103 cities in Norway





TSP – Implementation

103 cities in Norway:

	MTZ	gain(Svestka)	Steps(Dantzig)
_nvars	10609	21012	1082120
_ncons	10712	10917	10712
Time	?	?	?

Heuristics

- Why do we use a heuristic method to solve a TSP?
  - ➢ The problem is difficult (known to be NP-Hard)
  - ➢ No polynomial time algorithm for solving it to optimality
  - ➢ Exponential in the number of cities
  - ➢ We must solve relatively "large" instances of the problem
- Heuristics aims to efficiently generate very good solutions. They do not find the optimal solution, or at least do not guarantee the optimality of the found solutions.

Heuristics

Type of heuristics for TSP

- Construction heuristics:
  - builds a solution from scratch (starting with nothing).
- Improvement heuristics (neighborhood search):
  - starts with a solution, and then tries to improve the solution, usually by making small changes in the current solution.
- Metaheuristics
  - a high-level problem-independent algorithmic framework that combines operators/heuristics intelligently and provide a sufficiently good solution!

Heuristics

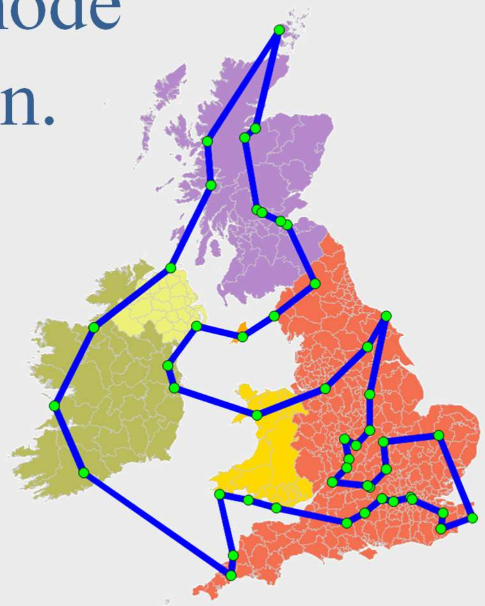
- Construction heuristics:
  - Nearest Neighbor Heuristic
  - Greedy Heuristic
  - Insertion Heuristics
    - Nearest insertion of arbitrary city
    - Nearest insertion
    - Farthest Insertion
    - Cheapest insertion
  - Christofides algorithm

Heuristics

- Improvement heuristics:
  - 2opt
  - 3opt
  - k-opt

TSP – Nearest Neighbor Heuristic

- Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

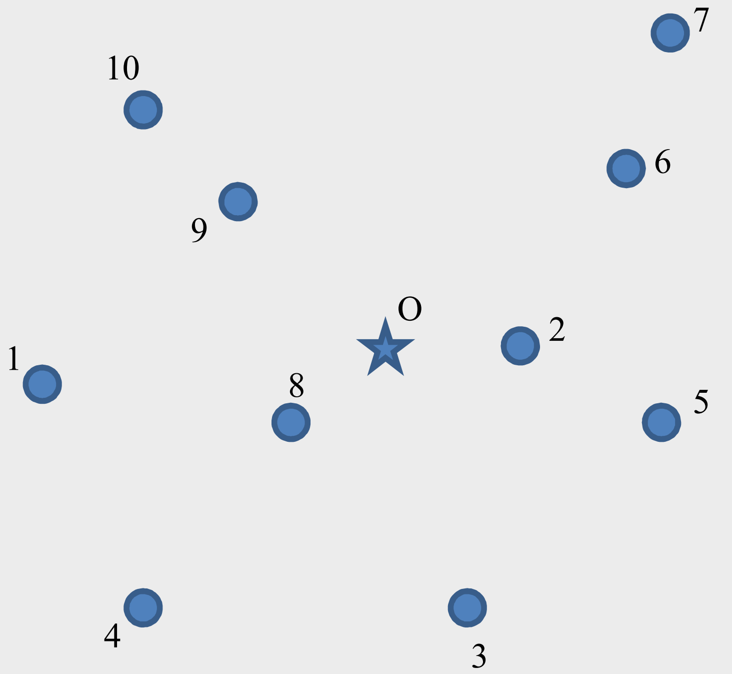


Nearest Neighbor Heuristic:

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

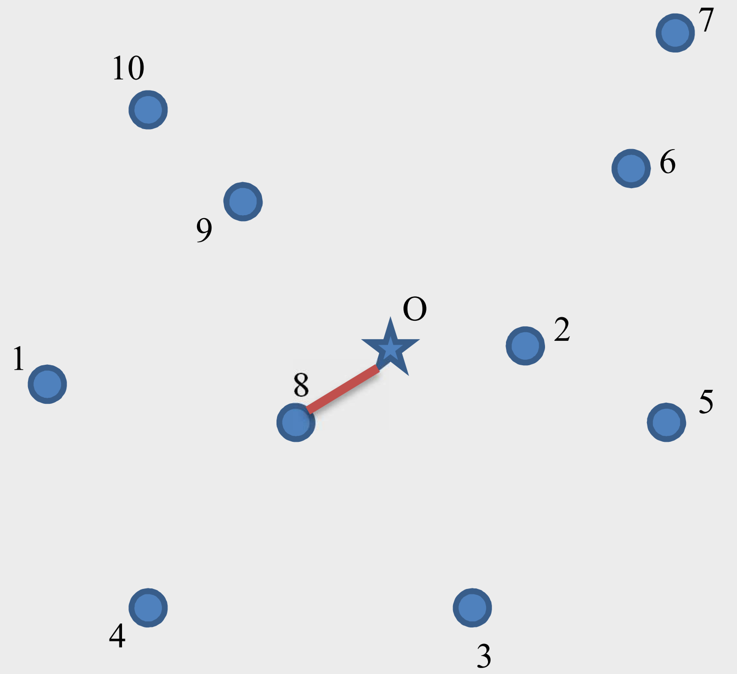
TSP – Nearest Neighbor Heuristic

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



TSP – Nearest Neighbor Heuristic

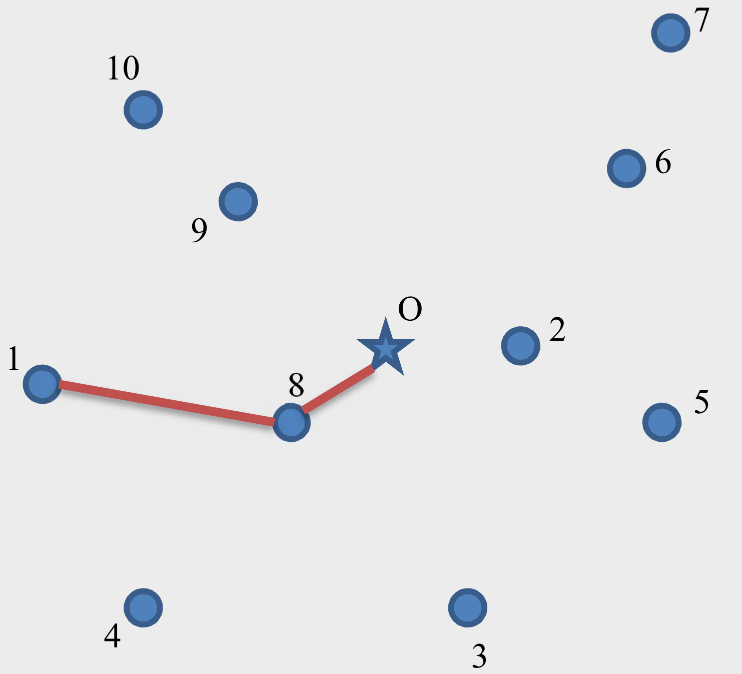
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	**7.1**	**3**	**7.3**	**8.6**	**6.3**	**7.1**	**10.8**	**2.8**	**5**	**8.6**	**0**



Tour: O-8

TSP – Nearest Neighbor Heuristic

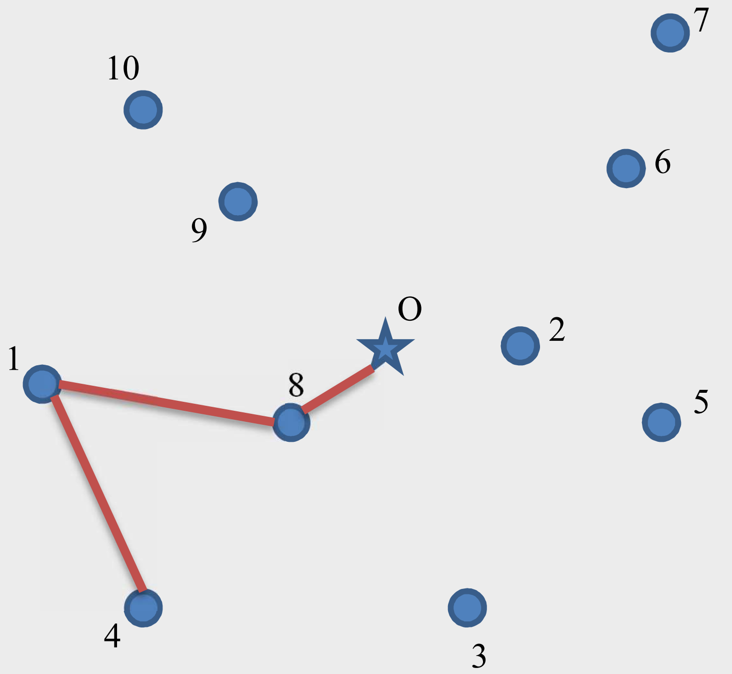
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1

TSP – Nearest Neighbor Heuristic

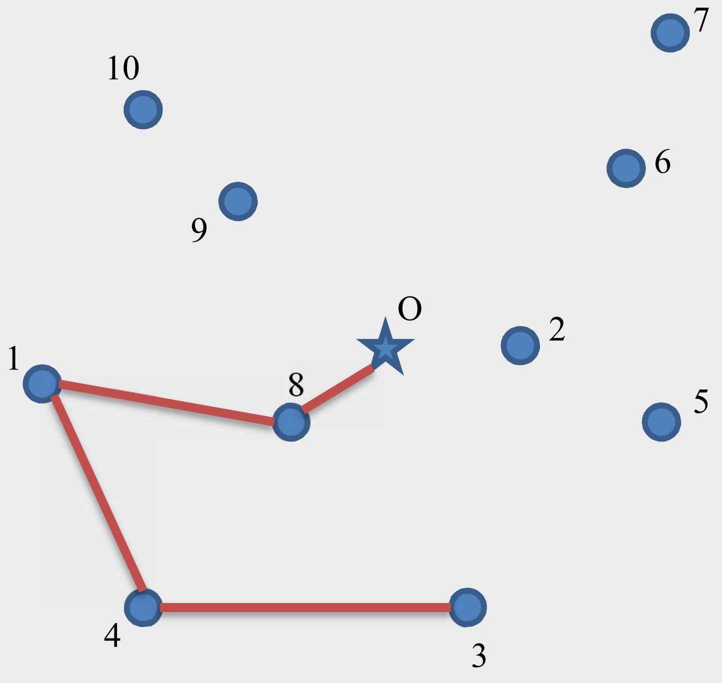
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4

TSP – Nearest Neighbor Heuristic

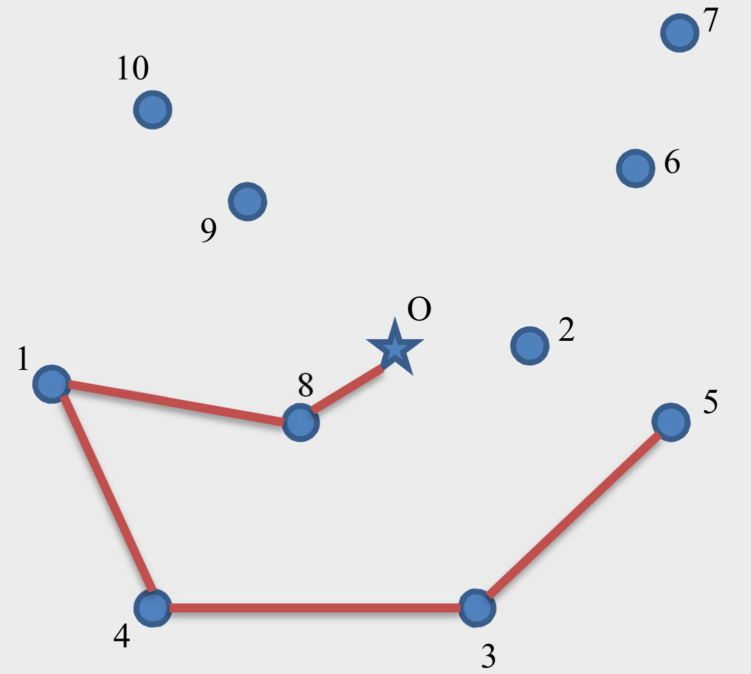
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3

TSP – Nearest Neighbor Heuristic

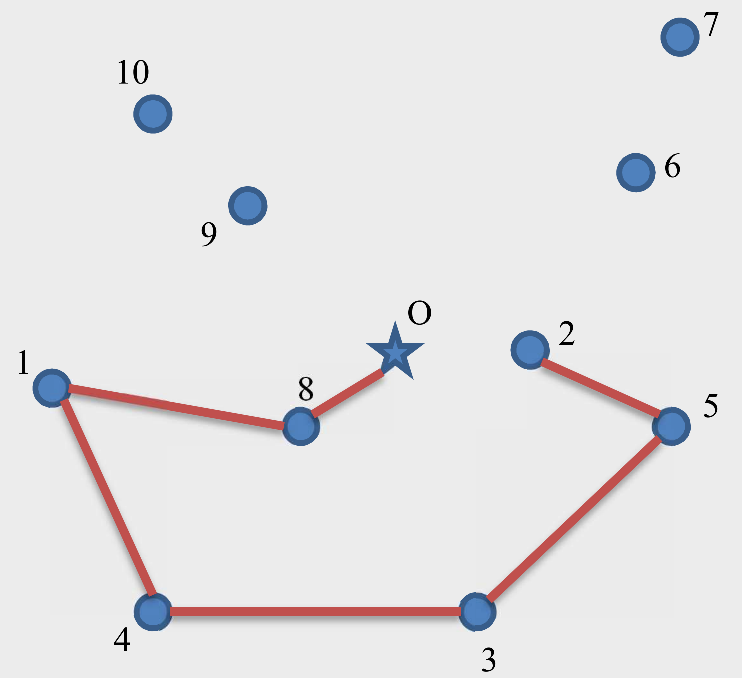
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3-5

TSP – Nearest Neighbor Heuristic

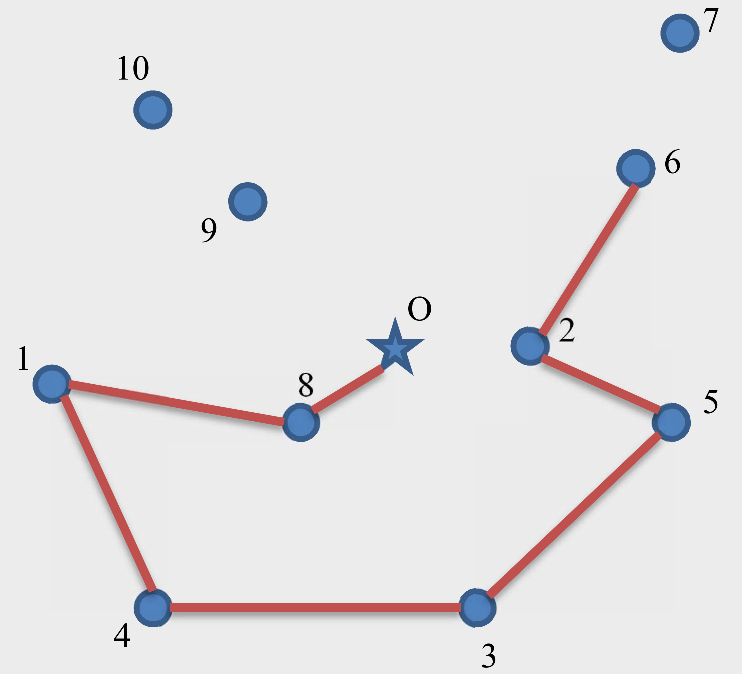
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3-5-2

TSP – Nearest Neighbor Heuristic

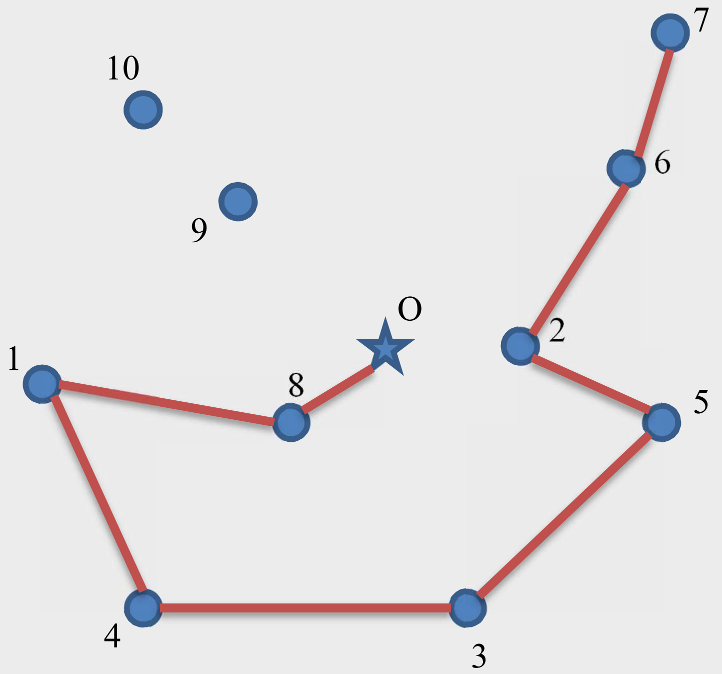
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3-5-2-6

TSP – Nearest Neighbor Heuristic

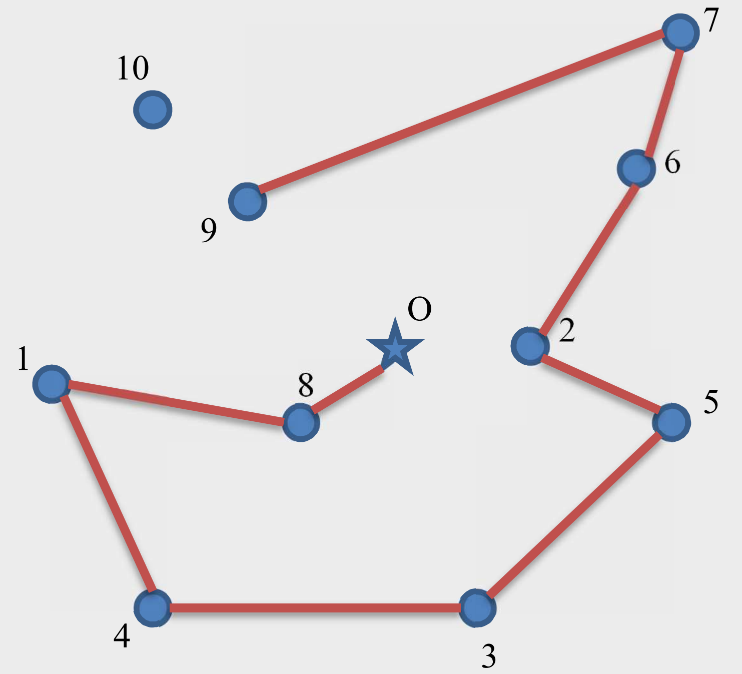
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3-5-2-6-7

TSP – Nearest Neighbor Heuristic

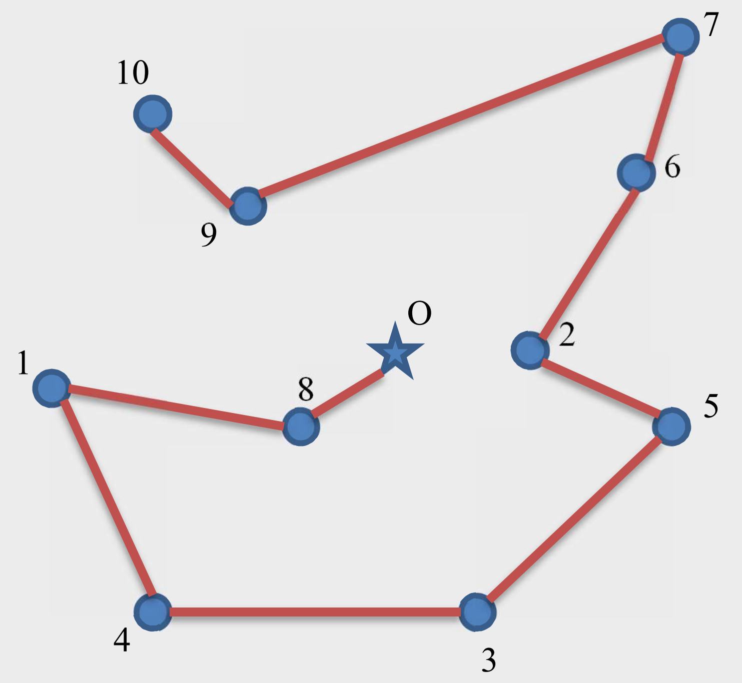
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3-5-2-6-7-9

TSP – Nearest Neighbor Heuristic

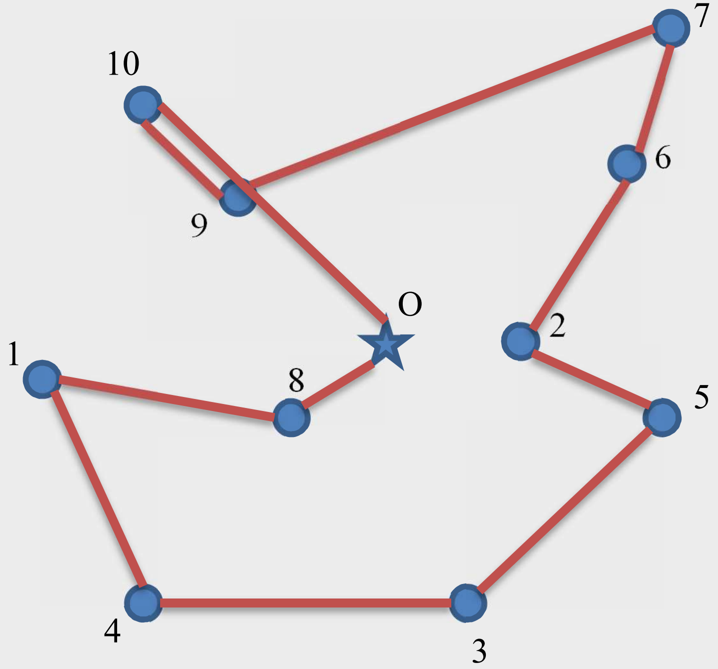
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3-5-2-6-7-9-10-O

TSP – Nearest Neighbor Heuristic

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3-5-2-6-7-9-10-O

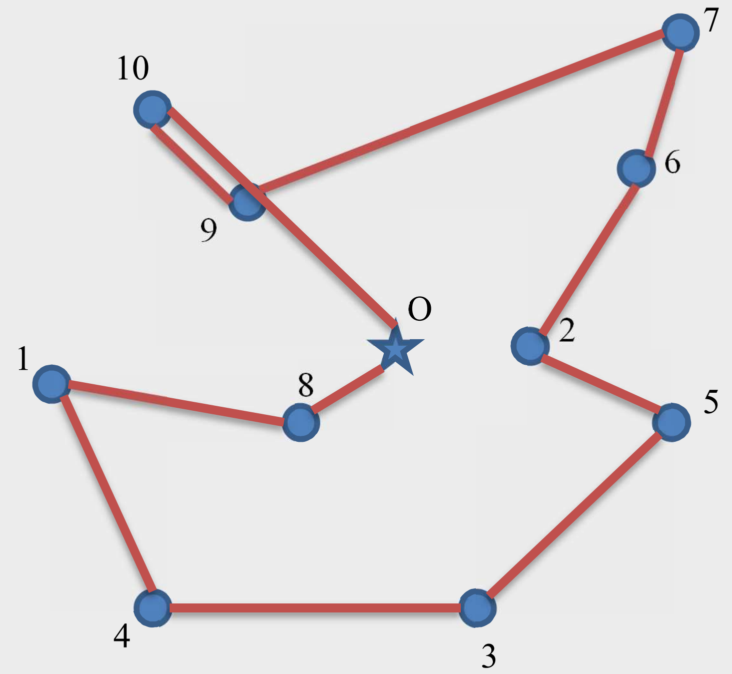
Length: 63.2

Improvement Heuristic: 2-Opt

1. Identify pairs of arcs (*i*-*j* and *k*-*l*), where
   $d(ij) + d(kl) > d(ik) + d(jl)$ (usually where they cross)
2. Select the pair with the largest difference, and re-connect the arcs (*i*-*k* and *j*-*l*)
3. Continue until there are no more crossed arcs.

TSP – 2-Opt

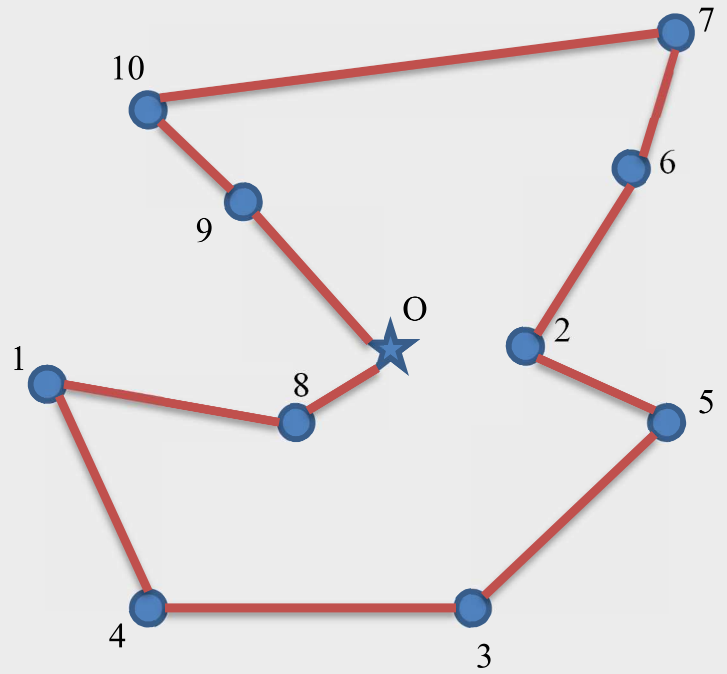
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



- Arcs 7-9 and 10-O cross
- d(79) + d(10-O) = 18.9 > d(7-10) + d(9-O) = 16.2
- Re-connect arcs 7-10 and 9-O

TSP – 2-Opt

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3-5-2-6-7-10-9-O

Length: 60.5

Tour length reduces from 63.2 to 60.5

TSP – Nearest Neighbor Heuristic

1. *Initialization* – Start with a partial tour with just one city i, randomly chosen;

2. *Selection* – Let $(1, \ldots, k)$ be the current partial tour ($k < n$). Find city $k + 1$ that is not yet in the tour and that is closer to k.

3. *Insertion* – Insert $k + 1$ at the end of the partial tour.

4. If all cities are inserted then STOP, else go back to 2.

TSP – Greedy Heuristic

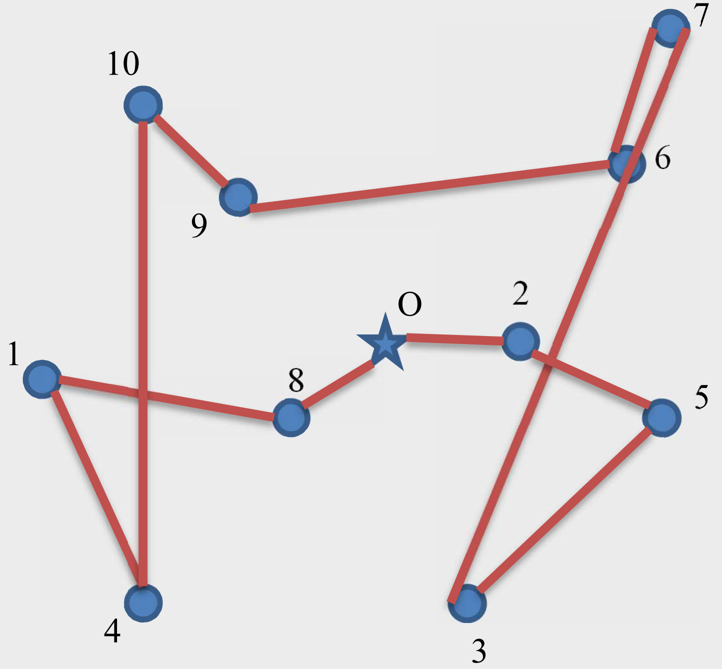
1. Sort all edges.

2. Select the shortest edge and add it to our tour if it doesn't violate any of the above constraints.

3. Do we have N edges in our tour? If no, repeat step 2.

TSP – Greedy Heuristic

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-10-9-6-7-3-5-2-O

Length: 73.5

TSP – Nearest Insertion of Arbitrary City

1. *Initialization* – Start with a partial tour with just one city i, randomly chosen; find the city j for which c_{ij} (distance or cost from i to j) is minimum and build the partial tour (i, j).

2. *Selection* – Given a partial tour, arbitrary select a city k that is not yet in the partial tour.

3. *Insertion* – Find the edge $\{i, j\}$, belonging to the partial tour, that minimizes $c_{ik} + c_{kj} - c_{ij}$. Insert k between i and j.

4. If all cities are inserted then STOP, else go back to 2.

TSP – Nearest Insertion

1. *Initialization* – Start with a partial tour with just one city i, randomly chosen; find the city j for which c_{ij} (distance or cost from i to j) is minimum and build the partial tour (i, j).

2. *Selection* – Find cities k and j (j belonging to the partial tour and k not belonging) for which c_{kj} is minimized.

3. *Insertion* – Find the edge $\{i, j\}$, belonging to the partial tour, that minimizes $c_{ik} + c_{kj} - c_{ij}$. Insert k between i and j.

4. If all cities are inserted then STOP, else go back to 2.

TSP – Farthest Insertion

1. *Initialization* – Start with a partial tour with just one city i, randomly chosen; find the city j for which c_{ij} (distance or cost from i to j) is minimum and build the partial tour (i, j).
2. *Selection* – Find cities k not belonging to the partial tour that is farthest from any of the cities belonging to the partial tour.
3. *Insertion* – Find the edge $\{i, j\}$, belonging to the partial tour, that minimizes $c_{ik} + c_{kj} - c_{ij}$. Insert k between i and j.
4. If all cities are inserted then STOP, else go back to 2.

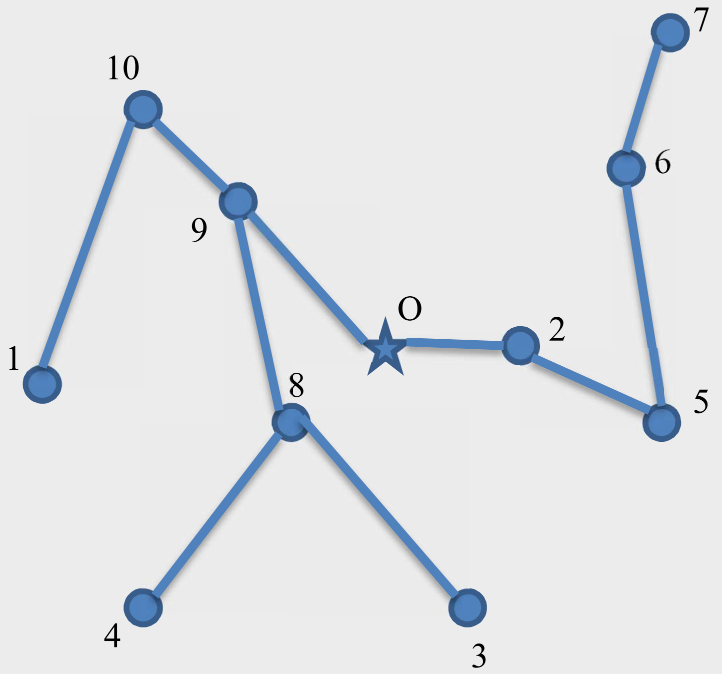
TSP – Cheapest Insertion

1. *Initialization* – Start with a partial tour with just one city i, randomly chosen; find the city j for which c_{ij} (distance or cost from i to j) is minimum and build the partial tour (i, j).
2. *Selection* – Find cities k, i and j (i and j being the extremes of an edge belonging to the partial tour and k not belonging to that tour) for which $c_{ik} + c_{kj} - c_{ij}$ is minimized.
3. *Insertion* – Insert k between i and j.
4. If all cities are inserted then STOP, else go back to 2.

Spanning Tree

- A tree which includes all of the vertices of a graph

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Length: 52.9

Minimum Spanning Tree

Objective: Find the minimum distance such that all nodes are visited once (i.e. no cycles).

The following algorithms run in polynomial time

- *Kruskal*'s algorithm
- *Prim*'s algorithm

Minimum Spanning Tree – Algorithms

Kruskal's algorithm

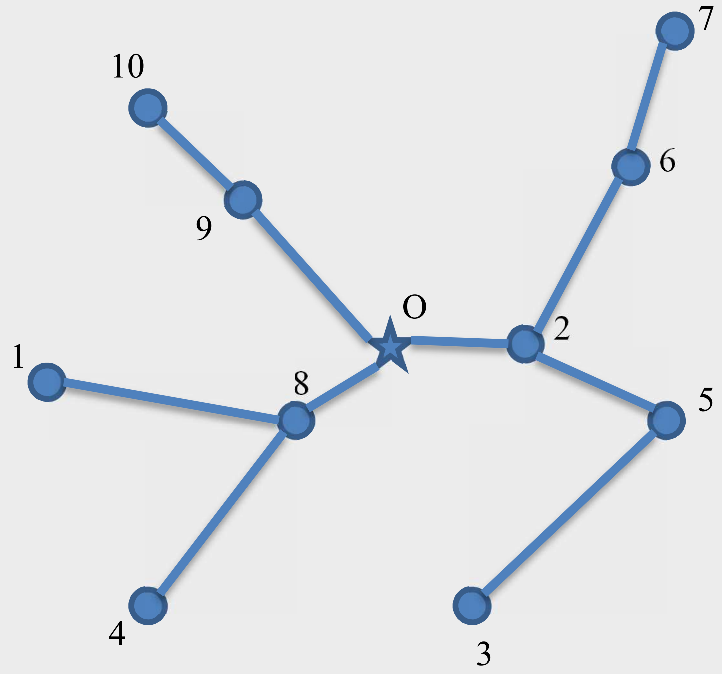
1. Select the shortest edge in a network
2. Select the next shortest edge which does not create a cycle
3. Repeat step 2 until all vertices have been connected

Prim's algorithm

1. Select any vertex
2. Select the shortest edge connected to that vertex
3. Select the shortest edge connected to any vertex already connected
4. Repeat step 3 until all vertices have been connected

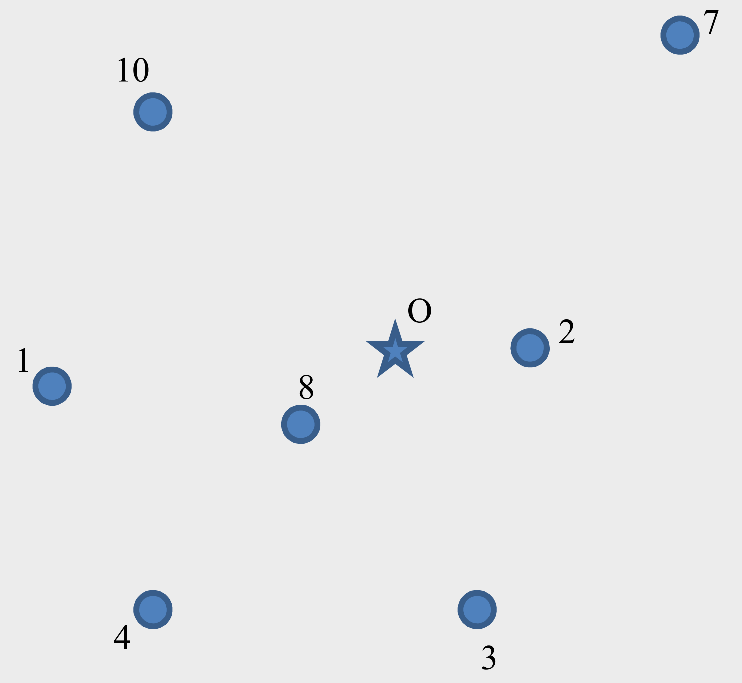
Minimum Spanning Tree

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



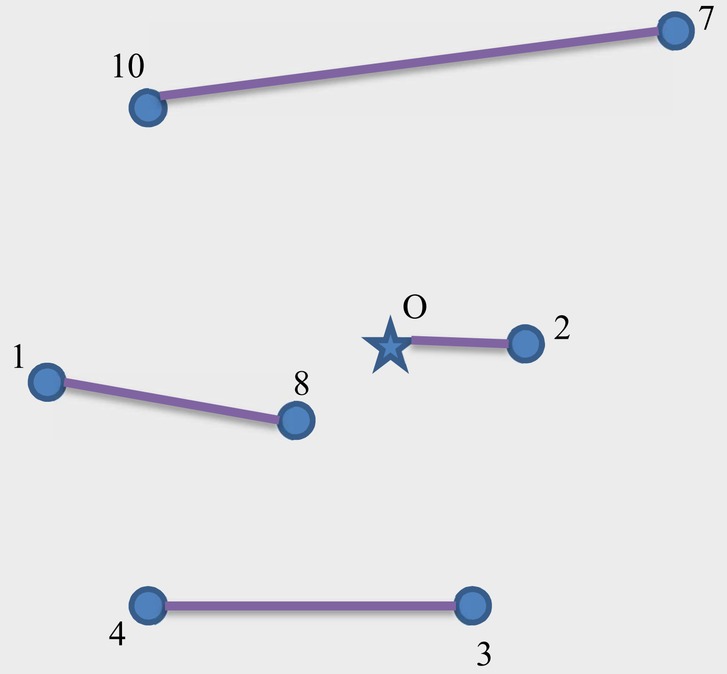
Minimum Spanning Tree

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



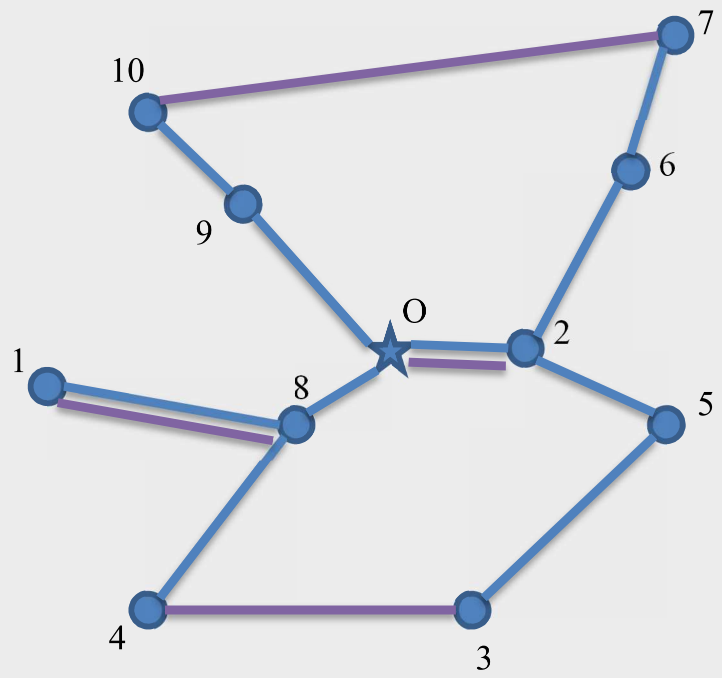
Minimum Spanning Tree

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Minimum Spanning Tree

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



TSP – Christofides Algorithm

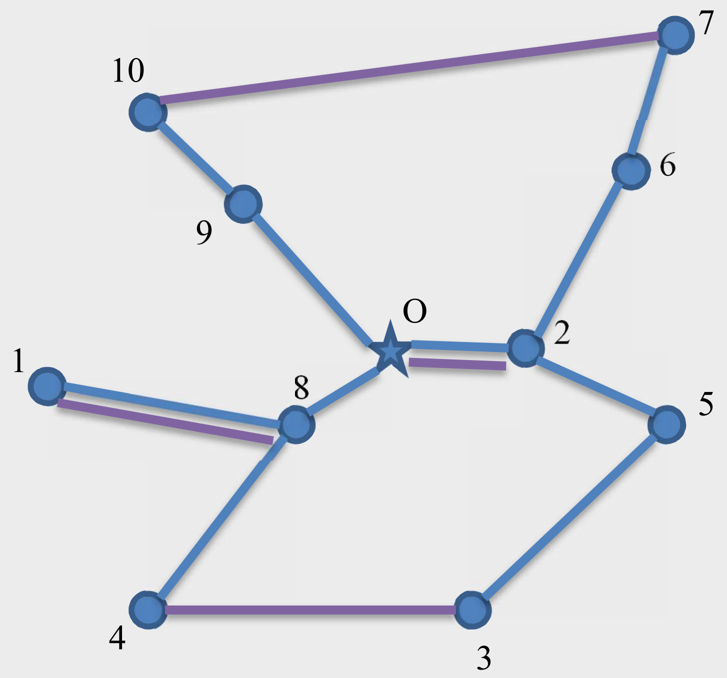
- It is an approximation algorithm that guarantees that its solutions will be within a factor of 3/2 of the optimal solution length
- As of 2017, this is the best approximation ratio that has been proven for the traveling salesman problem on general metric spaces, although better approximations are known for some special cases.

TSP – Christofides Algorithm

1. Create a minimum spanning tree T of G.
2. Let O be the set of vertices with odd degree in T.
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).

EULER CYCLE

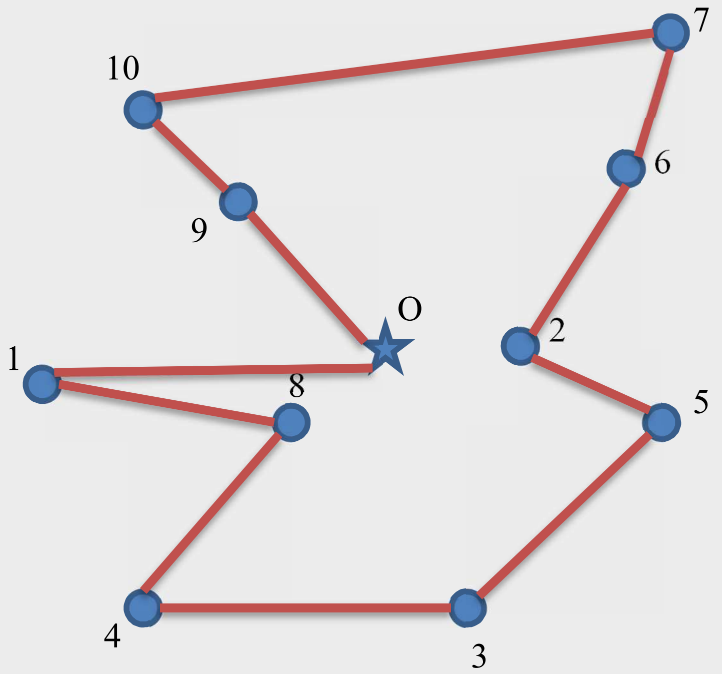
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



O-9-10-7-6-2-O-2-5-3-4-8-1-8-O

TSP – Christofides Algorithm

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0

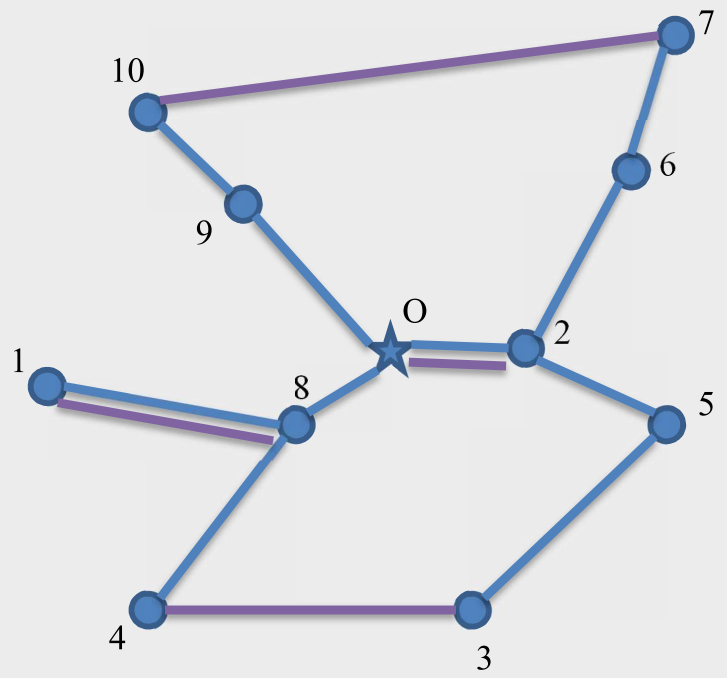


Tour: O-9-10-7-6-2-5-3-4-8-1-O

Length: 64.3

Euler Cycle

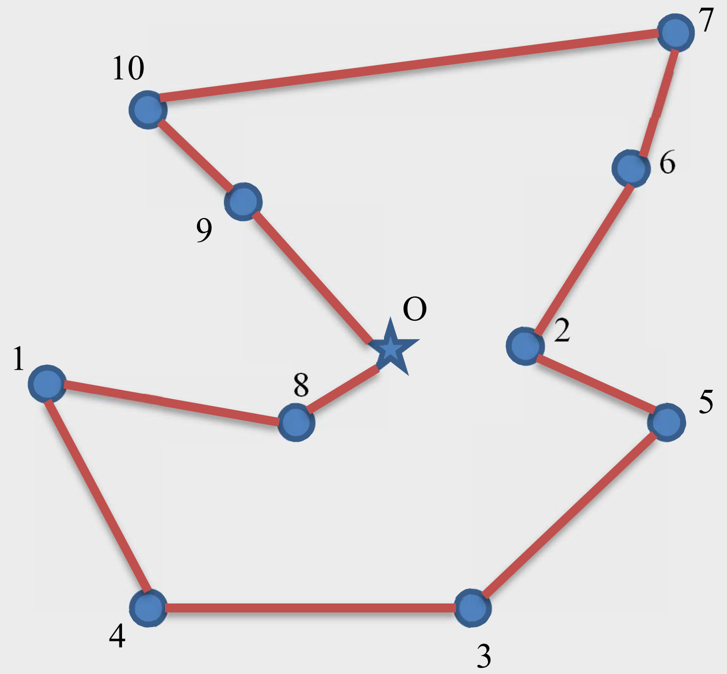
Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



O-8-1-8-4-3-5-2-O-2-6-7-10-9-O

TSP – Christofides Algorithm

Dis.	1	2	3	4	5	6	7	8	9	10	O
1	0	10	10.8	6.3	13	13.4	16.4	5.1	6.4	8.2	7.1
2	10	0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3
3	10.8	7.1	0	7	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7	0	12.1	15.6	19.4	5.8	11.2	14	8.6
5	13	3.6	6.4	12.1	0	7.1	11	8	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11	4.1	0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8	9.9	13.6	0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0	3.6	5
10	8.2	10.6	15.7	14	14.2	10.2	11.2	9.5	3.6	0	8.6
O	7.1	3	7.3	8.6	6.3	7.1	10.8	2.8	5	8.6	0



Tour: O-8-1-4-3-5-2-6-7-10-9-O

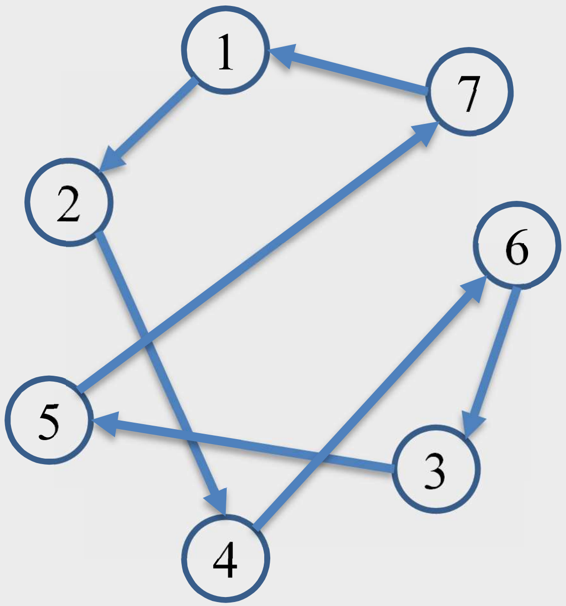
Length: 60.5

TSP – 2-Opt

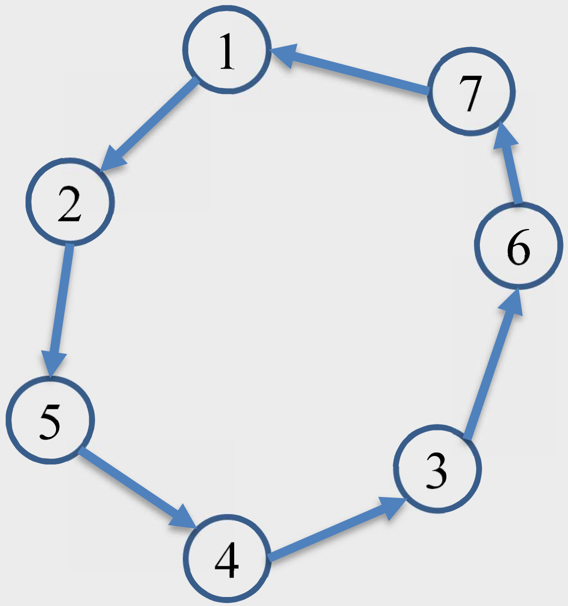
- *2-opt*
  - Note: a finite sequence of "swap (2-exchange)" can generate any tour (for TSP), including the optimum tour
  - Strategy:
    - Select the best swap among $N * (N - 1) / 2$ possible swap
    - Repeat this process until no improvement can be made)

TSP – 3-Opt

- *3-exchange → 3-opt*



1	2	4	6	3	5	7



1	2	5	4	3	6	7

Next Lecture

Lecture #14:

Vehicle Routing Problem