TSP

DA PROJECT 2

G07_9: Bruno Huang 202207517
Ricardo Yang 202208465

List of Implemented Functions

- `void backtrackingAlgorithm()`

  Time complexity: O(V!)

  Space complexity: O(V)

- `void heldKarp(const int& origin)`

  Time complexity: O(V^2 * 2^V)

  Space complexity: O(V * 2^V)

List of Implemented Functions

- `void triangularApproximationMSTAlgorithm(int origin, bool fullyConnected)`

  Time complexity: O((V+E) * logV)

  Space complexity: O(V)

- `void nearestNeighborAlgorithm(const int& origin)`

  Time complexity: O(V^2)

  Space complexity: O(V)

List of Implemented Functions

- `void kNearestNeighborAlgorithm(const int& origin, int k)`

  Time complexity: O(V^2 * logV + V * k)

  Space complexity: O(V)

- `void twoOptAlgorithm(const int& origin)`

  Time complexity: O(k * V^2)

  Space complexity: O(V)

List of Implemented Functions

- `void threeOptAlgorithm(const int& origin)`

  Time complexity: O(k * V^3)

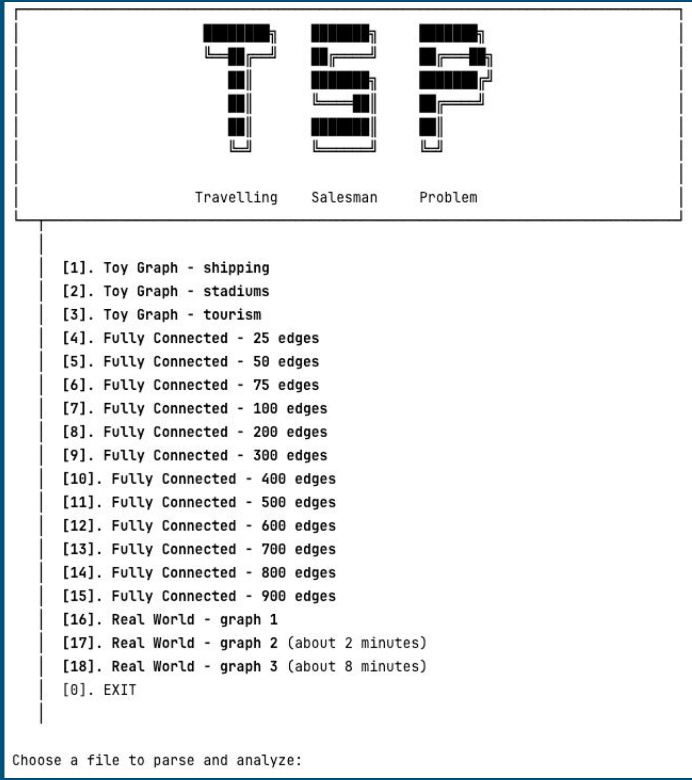
  Space complexity: O(V)

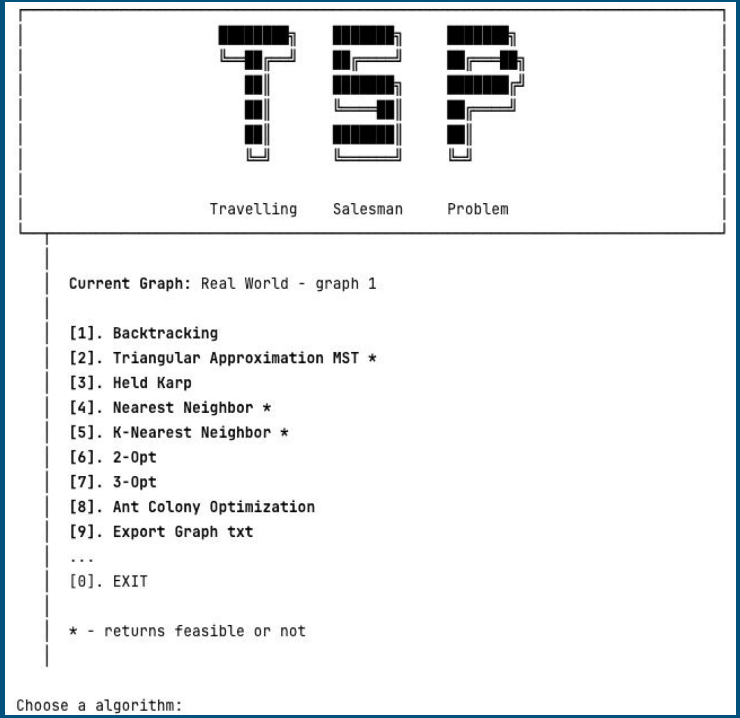
- `void antColonyOptimization(const int &origin, int numAnts, int numIterations, bool fullyConnected)`

  Time complexity: O(V^2 * numAnts * numIterations)

  Space complexity: O(V^2)

User Interface





User Interface

Current Graph: Real World - graph 1

```
[2]. Triangular Approximation MST *
Choose a algorithm: 2

Choose origin (int): 0

Considering graph fully connected? [1-YES / 0-NO]: 1

----------------------------------------
Distance: 1121403
Number of Nodes: 1000
Path: 0 -> 496 -> 878 -> 221 -> 98 -> 134 -> 7 -> 152
----------------------------------------

Time taken by function: 0 s

Press ENTER to continue...
```

```
[4]. Nearest Neighbor *
Choose a algorithm: 4

Choose origin (int): 0

----------------------------------------
Distance: 1004706
Number of Nodes: 1000
Path: 0 -> 496 -> 878 -> 221 -> 98 -> 134 -> 7 -> 152
----------------------------------------

Time taken by function: 0 s

Press ENTER to continue...
```

```
[5]. K-Nearest Neighbor *
Choose a algorithm: 5

Choose origin (int): 0

Choose K (int > 0): 5

----------------------------------------
Distance: 1420416
Number of Nodes: 1000
Path: 0 -> 496 -> 878 -> 221 -> 632 -> 662 -> 362 -> 161
----------------------------------------

Time taken by function: 0 s

Press ENTER to continue...
```

User Interface

```
Current Graph: Fully Connected - 100 edges
```

```
[6]. 2-Opt
Choose a algorithm: 6

Choose origin (int): 0

-----------------------------------------
Distance: 698082
Number of Nodes: 100
Path: 0 -> 55 -> 72 -> 42 -> 14 -> 58 -> 5 -> 12 -> 78
-----------------------------------------

Time taken by function: 0 s

Press ENTER to continue...
```

```
[7]. 3-Opt
Choose a algorithm: 7

Choose origin (int): 0

-----------------------------------------
Distance: 669506
Number of Nodes: 100
Path: 0 -> 55 -> 72 -> 42 -> 56 -> 8 -> 26 -> 21 -> 16 -> 4
-----------------------------------------

Time taken by function: 10 s

Press ENTER to continue...
```

User Interface

```
Current Graph: Fully Connected - 100 edges
```

```
[8]. Ant Colony Optimization
Choose a algorithm: 8

Choose origin (int): 0

Choose number of ants (int > 0): 10

Choose number of iterations (int > 0): 10

Considering graph fully connected? [1-YES / 0-NO]: 1

----------------------------------------
Distance: 632855
Number of Nodes: 100
Path: 0 -> 55 -> 72 -> 42 -> 7 -> 60 -> 84 -> 91 -> 64 -> 47 -> 11 -> 51 -> 39 -> 36 -> 98 -> 68
----------------------------------------

Time taken by function: 0 s

Press ENTER to continue...
```

User Interface

```
Current Graph: Fully Connected - 100 edges
```

```
                                      [9]. Export Graph txt
Choose a algorithm: 9
Exporting graph text to "../output/Fully Connected - 100 edges.txt"
( It might take some time )

Time taken by function: 0 s

Press ENTER to continue...
```

```
Vertices: 100
(target vertex, edge weight)

Vertex [0] :
    Adjacent edges: (1,40920.9) (2,31761.5) (3,37649.3) (4,38500.3) (5,19127.2) (6,33731.7) (7,15474.5) (8,3279
Vertex [1] :
    Adjacent edges: (0,40920.9) (2,38026.7) (3,27787.1) (4,60886.3) (5,33079) (6,16107.1) (7,30300.1) (8,55183.
Vertex [2] :
    Adjacent edges: (0,31761.5) (1,38026.7) (3,39858.5) (4,56830) (5,31792) (6,37193.7) (7,22720.4) (8,51127.3)
Vertex [3] :
    Adjacent edges: (0,37649.3) (1,27787.1) (2,39858.5) (4,46400.4) (5,18215.6) (6,11683) (7,31522.3) (8,40697.
Vertex [4] :
    Adjacent edges: (0,38500.3) (1,60886.3) (2,56830) (3,46400.4) (5,32081.5) (6,51495.1) (7,38045.6) (8,13013.
Vertex [5] :
    Adjacent edges: (0,19127.2) (1,33079) (2,31792) (3,18215.6) (4,32081.5) (6,22971.4) (7,16923.3) (8,28176.9)
Vertex [6] :
    Adjacent edges: (0,33731.7) (1,16107.1) (2,37193.7) (3,11683) (4,51495.1) (5,22971.4) (7,28513.7) (8,46509.
Vertex [7] :
    Adjacent edges: (0,15474.5) (1,30300.1) (2,22720.4) (3,31522.3) (4,38045.6) (5,16923.3) (6,28513.7) (8,3075
Vertex [8] :
    Adjacent edges: (0,32797.7) (1,55183.7) (2,51127.3) (3,40697.7) (4,13013.1) (5,28176.9) (6,46509.2) (7,3075
Vertex [9] :
    Adjacent edges: (0,36635) (1,28985.6) (2,26609.6) (3,33646.2) (4,59206.1) (5,28540.1) (6,30812.6) (7,26270)
```

k-nearest neighbor is by default K = 1
*ACO considering 10 ants, 10 iterations, alfa = 1.0, beta = 5.0, evaporationRate = 0.5

Average Execution Time (s)

	Graphs	Backtracking	Triangular Approximation	Held-Karp	Nearest Neighbor	K-Nearest Neighbor	2-Opt	3-Opt	Ant Colony Optimization*
TOYS	shipping	< 1,0	-	< 1,0	-	< 1,0	-	-	-
	stadiums	1	< 1,0	< 1,0	< 1,0	< 1,0	< 1,0	< 1,0	< 1,0
	tourism	< 1,0	< 1,0	< 1,0	< 1,0	< 1,0	< 1,0	< 1,0	< 1,0
EXTRA FULLY CONNECTED	edges_25	-	< 1,0	37 min	< 1,0	< 1,0	< 1,0	< 1,0	< 1,0
	edges_50	-	< 1,0	-	< 1,0	< 1,0	< 1,0	< 1,0	< 1,0
	edges_75	-	< 1,0	-	< 1,0	< 1,0	< 1,0	4	< 1,0
	edges_100	-	< 1,0	-	< 1,0	< 1,0	< 1,0	10	< 1,0
	edges_200	-	< 1,0	-	< 1,0	< 1,0	5	6 min 20 s	1
	edges_300	-	< 1,0	-	< 1,0	< 1,0	49	19	5
	edges_400	-	< 1,0	-	< 1,0	< 1,0	1	7 min 15 s	12
	edges_500	-	< 1,0	-	< 1,0	< 1,0	2 min 40 s	-	24
	edges_600	-	< 1,0	-	< 1,0	< 1,0	3 min 19 s	17 min 51 s	42
	edges_700	-	< 1,0	-	< 1,0	< 1,0	4 min 9 s	-	1 min 05 s
	edges_800	-	< 1,0	-	< 1,0	< 1,0	12 min 8 s	-	1 min 41 s
	edges_900	-	< 1,0	-	< 1,0	< 1,0	12 min 21 s	-	2 min 13 s
REAL	real world 1	-	< 1,0	-	< 1,0	< 1,0	-	-	3 min 07s
	real world 2	-	< 1,0	-	-	-	-	-	-
	real world 3	-	3	-	-	-	-	-	-

k-nearest neighbor is by default K = 1
*ACO considering 10 ants, 10 iterations, alfa = 1.0, beta = 5.0, evaporationRate = 0.5

Distance - Consider graphs as how they are

	Graphs	Backtracking	Triangular Approximation	Held-Karp	Nearest Neighbor	K-Nearest Neighbor	2-Opt	3-Opt	Ant Colony Optimization*
TOYS	shipping	86,7	-	86,7	-	96 (k=2)	-	-	-
	stadiums	341	393	341	403	403	358	390	390
	tourism	2600	2600	2600	2600	2600	2600	2600	2600
EXTRA FULLY CONNECTED	edges_25	-	364925	280592	300938	300938	294806	293456	294467
	edges_50	-	542163	-	534124	534124	528723	527650	505878
	edges_75	-	626244	-	613453	613453	608498	600360	609842
	edges_100	-	671365	-	705224	705224	698082	669506	657769
	edges_200	-	891268	-	848805	848805	846191	841349	885140
	edges_300	-	1134182	-	1099096	1099096	1095670	1098243	1209880
	edges_400	-	1330621	-	1407873	1407873	1403651	1402851	1424019
	edges_500	-	1422107	-	1366828	1366828	1364619	-	1511514
	edges_600	-	1579860	-	1604239	1604239	1602233	1600495	1744216
	edges_700	-	1741831	-	1741831	1741831	1713539	-	1858184
	edges_800	-	1838552	-	1836108	1836108	1834990	-	1968617
	edges_900	-	1990961	-	1888266	1888266	1887096	-	2181466
REAL	real world 1	-	1121403	-	1004706	1004706	-	-	1108087
	real world 2	-	-	-	-	-	-	-	-
	real world 3	-	-	-	-	-	-	-	-

BEST	8 points
GOOD	5 points
OK	3 points
BAD	1 point
N/A or TLE	0 points

	Backtracking	Triangular Approximation	Held-Karp	Nearest Neighbor	K-Nearest Neighbor	2-Opt	3-Opt	Ant Colony Optimization*
Score	78	200 + 1	57	189	200	161	104	147

TRIANGULAR APPROXIMATION*

K-NEAREST NEIGHBOR

NEAREST NEIGHBOR

2-OPT (WITH NEAREST NEIGHBOR)

* our triangular approximation is adapted to handle with fully/not fully connected graphs, for this we gave an extra point

Highlight Functionalities

We provide users with the flexibility to select and analyze the graph of their choice. Additionally, users can choose an origin node different from the default, enabling a more comprehensive analysis of how various algorithms perform on diverse graphs.

Main Difficulties and Participation of Each Member

The main challenge was likely implementing the algorithms and adapting them to graphs that could be either fully connected or not. Additionally, parsing real-world graphs and running the algorithms on them took considerable time and effort.

Effort:

- Bruno Huang - 50%
- Ricardo Yang - 50%