

INFO 6205 Spring 2023 Project

Project-Travelling Salesman

Submitted by:

Name: Mulpuri Uday Kiran Reddy **NUID:**002781063
Nandre Anurag **NUID:**002785735

Aim:

The aim of the project is to solve the Travelling Salesman Problem, which is an NP-hard problem, using the Christofides algorithm and optimizing further with tactical and strategic optimizations

Approach:

- The first step of the project involved processing the given dataset (Data\tsppdatapoints.xlsx) into the necessary format (points.xlsx). Subsequently, we computed the distance between each vertex in terms of edges and recorded the results in a separate file (routes.xlsx).
- Next, we utilized Kruskal's Union Find Method to identify the minimum spanning tree (MST) of the graph. This was achieved by implementing the "minimumSpanningTree()" function within the MinWeightService Class.
- After obtaining the MST, we proceeded to extract the list of odd vertices from the MST Tour.
- The next step involved identifying the perfect matching for the list of odd. This was accomplished by utilizing the method "findMinimumWeightMatching()" present within the MinWeightServiceClass.
- Subsequently, we combined the Perfect Matching and MST to ensure that every vertex in the graph had an even number of edges. This allowed us to generate an Eulerian tour of the graph.
- Using the Eulerian tour obtained in the previous step, we aimed to create a TSP tour that visits each vertex in the graph and returns to the starting vertex. It was necessary to skip over any vertex that had already been encountered during the tour.
- In parallel with the previous step, we calculated the distance/weight of the TSP tour.
- Next, we performed two tactical optimizations, namely the 2-opt and 3-opt techniques, on the TSP tour obtained in the previous step.
- Furthermore, we applied two strategic optimizations, simulated annealing and ant colony optimization, to further enhance the TSP tour.

Algorithm:

Kruskal's Algorithm: The Minimum Spanning Tree (MST) algorithm is a commonly used approach to approximate a solution for the Traveling Salesman Problem (TSP). The MST algorithm begins by constructing a complete graph of the cities, and then computes the MST of the graph using Prim's or Kruskal's algorithm. The resulting MST is then traversed to obtain a sequence of cities, which is then converted into a Hamiltonian circuit by adding the starting city to the end. While the MST algorithm does not guarantee an optimal solution for the TSP, it is often used in practice because it can provide a good approximation. By using the MST to connect the cities with the minimum possible total weight, we can ensure that the resulting Hamiltonian circuit is relatively efficient and avoids unnecessary detours. Overall, the MST algorithm can be a useful tool for finding approximate solutions to the TSP in a variety of applications.

Tactical Optimizations:

2 opt:

The 2-opt algorithm is a heuristic optimization algorithm for the Traveling Salesman Problem (TSP) that improves an initial TSP tour generated from the Christofides algorithm. The algorithm starts by choosing two edges i and j in the tour such that the vertices do not share a common edge. It then generates a new tour by adding the elements from index 0 to i , followed by the elements of the original tour from j to i in reverse order, and finally the remaining elements from $j+1$. If the resulting tour is shorter than the original tour, it is accepted as the new tour. Otherwise, the algorithm repeats steps 2-4 for all possible pairs of edges until no improvement is possible. By iteratively applying this algorithm, the TSP tour can be gradually improved until no further improvements can be made. The 2-opt algorithm is a simple yet effective way to optimize TSP tours and is often used as a building block in more advanced optimization algorithms.

3 Opt:

To improve an initial TSP tour, the 3-opt algorithm selects three non-adjacent edges x , y , and z in the tour and generates a new tour by adding the elements from index 0 to x , followed by the elements of the original tour from y to x in reverse order, then the elements from z to y in reverse order, and finally the remaining elements from $z+1$. If the resulting tour is shorter than the original tour, it is accepted as the new tour. If not, the algorithm repeats steps 1 and 2 for all possible combinations of three non-adjacent edges until no improvement is possible. By iteratively applying this algorithm, the TSP tour can be gradually improved until no further improvements can be made. The 3-opt algorithm is a more advanced version of the 2-opt algorithm and is often used as a building block in more sophisticated optimization algorithms.

Strategic Optimizations:

Simulated Annealing:

Simulated annealing is a heuristic optimization algorithm for the Traveling Salesman Problem (TSP). It works by iteratively generating candidate solutions through 2-opt optimization, starting from an initial tour. If a new candidate solution is better than the current one, it is always accepted as the new solution. However, if the new solution is worse, it may still be accepted with a probability that decreases over time. This probability is determined by a temperature parameter, which controls the degree of randomness in the search process. As the algorithm progresses, the temperature parameter gradually decreases over iterations proportional to a cooling rate. This enables the algorithm to converge towards a better solution, while still allowing it to explore less promising solutions early on. By doing so, simulated annealing can discover TSP tours that are better than the tour generated by the Christofides algorithm, which is commonly used as a benchmark for TSP solvers.

Ant Colony Optimization

Ant colony optimization is a metaheuristic algorithm that uses a population of artificial ants to search for good solutions to a problem. For the Traveling Salesman Problem (TSP), ants construct solutions by iteratively selecting cities based on a pheromone trail that reflects the quality of previously constructed solutions. The pheromone trail is updated based on the quality of the constructed solutions. Ants are more likely to visit cities with a higher pheromone trail, as this indicates a better quality solution. However, to avoid getting trapped in local optima, ants also explore new paths randomly. Over time, the trail is reinforced on the best paths and evaporates on the less attractive paths. This helps the algorithm converge towards a good quality solution by focusing on the most promising paths while avoiding less optimal paths. By doing so, ant colony optimization can discover TSP tours that are better than the tour generated by other metaheuristic algorithms.

Data Structures:

1. Graph

- Adjacency Matrix
- Adjacency List

Used In: MinWeightService, ThreeOptService, TwoOptService, SimulatedAnnealingService, LinKernighan20pt.

2. Disjoint Set

- Union Find
- Path Compression

Used In: MinWeightService, UnionFind

3. List

Used In: MinWeightService, ThreeOptService, TwoOptService, SimulatedAnnealingService, LinKernighan20pt, ExcelFetchPointService, ExcelFetchRouteService, CsvUtilService, RouteWeightService

4. Set

Used In: MinWeightService, CsvUtilService

5. Map

Used In: MinWeightService, CsvUtilService

6. Priority Queue

Used In: MinWeightService

7. Arrays

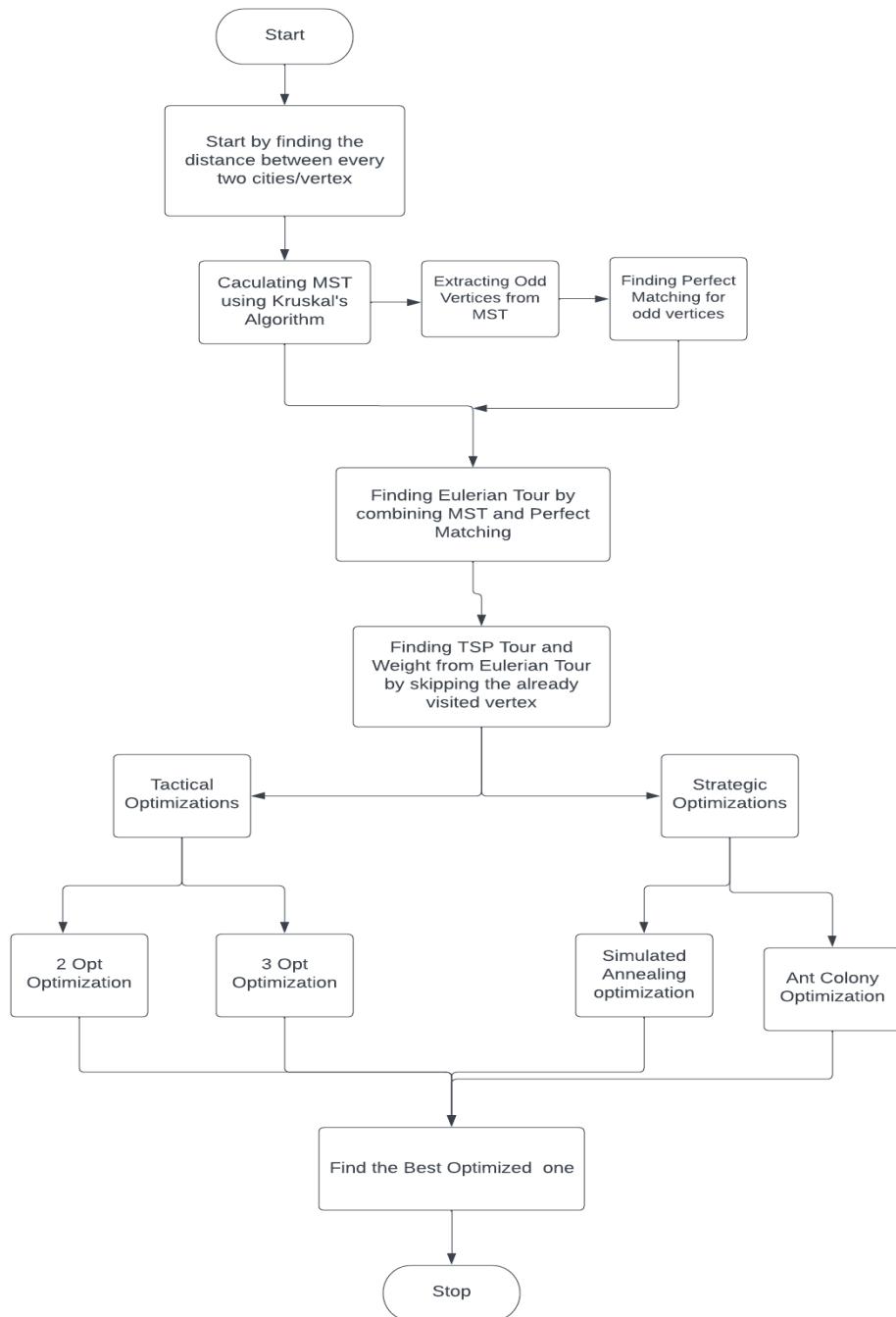
8. Stack

Used In: MinWeightService

Invariants:

- Temperature
- Rate of Cooling
- Iterations

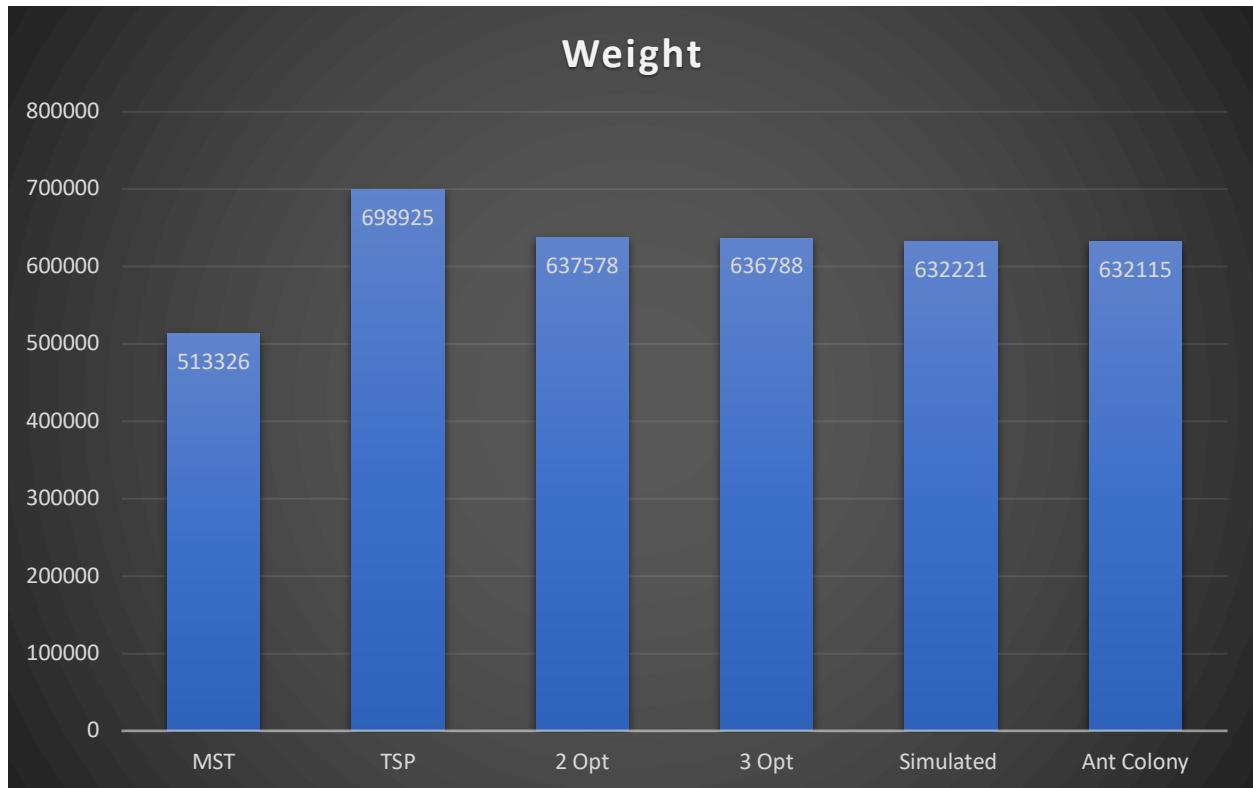
Flowchart:



Observations and Graphical Analysis:

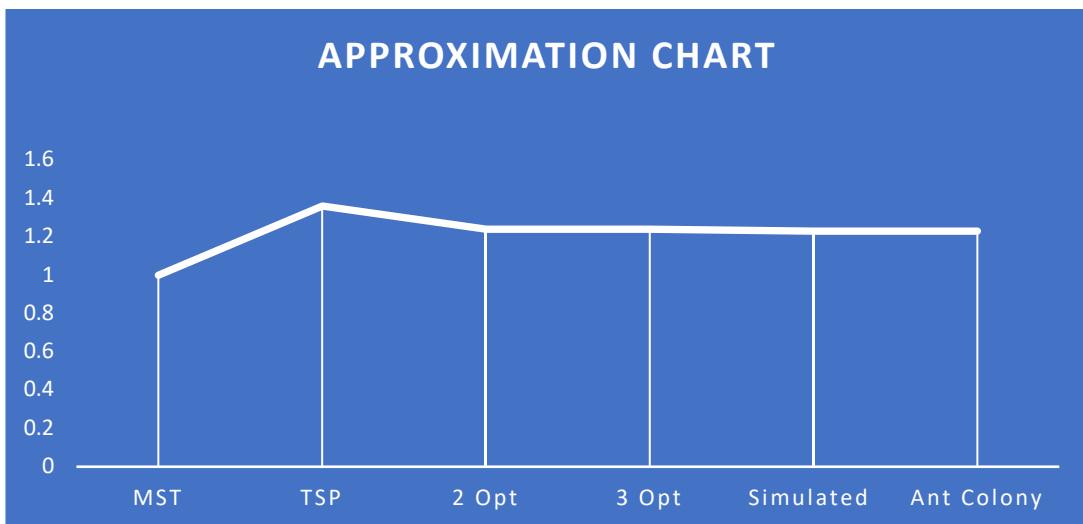
Graph for Distances of MST,TSP and Optimizations:

MST	TSP	2 Opt	3 Opt	Simulated	Ant Colony
513326	698925	637578	636788	632221	632115



Below are the values and Graph for Approximation calculation between MST and optimizations:

MST	TSP	2 Opt	3 Opt	Simulated	Ant Colony
1	1.36156166	1.24205281	1.24051383	1.23161695	1.23141045



Results and Mathematical Analysis:

MST Tour Weight(meters) = 513326.09539907286

TSP Tour Weight(Meters) = 698925.3607768689

Approx Ratio= TSP/MST=698925.3607768689/513326.09539907286

Approximation Ratio=1.36

MST Tour Weight(meters) = 513326.09539907286

After 2 Opt Weight = 637578.4578100945

Approx Ratio= 2 opt/MST=637578.4578100945/513326.09539907286

Approximation Ratio=1.24

The screenshot shows the Eclipse IDE interface with the following details:

- Package Explorer:** Shows the project structure for "INFO6205 [INFO6205 Spring2023]" and "PSA_Project [PSA_Project main]".
- TspApplication.java:** The active file in the editor.
- Console Output:** Displays the execution results of the application.

Output from TspApplication.java:

```
174     tspSimulated = getTour().stream().forEach(e -> {
175         Vertex v = new Vertex(e, vertexMap, getE());
176         simulatedAnnealingVertexTour.add(v);
177     });
178
179
180     vertexId = 141 CrimeId = ea746vertexId = 0 CrimeId = 47823
181
182 ****
183 **** 2 Opt Execution Time in Millisecs: 3999
184 **** 2 Opt Tsp Weight in meters 637578.4578100945
185 ****
186 vertexId = 0 CrimeId = 47823vertexId = 368 CrimeId = 64e7vertexId = 106 CrimeId = 74aa7vertexId = 305 CrimeId = b9086
187 vertexId = 7 CrimeId = cee3bvertexId = 418 CrimeId = d1eavertexId = 62 CrimeId = 987b5vertexId = 112 CrimeId = f37cc
188 vertexId = 140 CrimeId = c8283vertexId = 382 CrimeId = 7773fvertexId = 414 CrimeId = d4615vertexId = 427 CrimeId = aae64
189 vertexId = 53 CrimeId = 8933dvertexId = 528 CrimeId = e0345vertexId = 345 CrimeId = 98035vertexId = 515 CrimeId = 8e835
190 vertexId = 158 CrimeId = f993cvertexId = 425 CrimeId = 84266vertexId = 445 CrimeId = 9a639vertexId = 503 CrimeId = 56691
191 vertexId = 553 CrimeId = b2d5dvertexId = 324 CrimeId = 7040fvertexId = 186 CrimeId = af4cevertexId = 446 CrimeId = 53135
192 vertexId = 400 CrimeId = 6c267vertexId = 54 CrimeId = faeaavertexId = 217 CrimeId = b71c9vertexId = 577 CrimeId = 7283d
193 vertexId = 176 CrimeId = fdcf4vertexId = 142 CrimeId = f5b6fvertexId = 578 CrimeId = 83178vertexId = 406 CrimeId = bdaf6
194 vertexId = 566 CrimeId = 3ee1vertexId = 71 CrimeId = ba3aevertexId = 136 CrimeId = 53257vertexId = 83 CrimeId = 5da87
195 vertexId = 285 CrimeId = a833dvertexId = 199 CrimeId = 9e51cvertexId = 63 CrimeId = e85eevertexId = 244 CrimeId = 44fb2
196 vertexId = 53 CrimeId = 36caevertexId = 549 CrimeId = c0d5fvertexId = 497 CrimeId = fa4e7vertexId = 525 CrimeId = 6a7d5
197 vertexId = 236 CrimeId = 4dc5fvertexId = 351 CrimeId = ddbd1vertexId = 88 CrimeId = 2d0aevertexId = 385 CrimeId = ja25c
198 vertexId = 79 CrimeId = 34277vertexId = 474 CrimeId = e8c89vertexId = 541 CrimeId = c539avertexId = 68 CrimeId = 19cb3
199 vertexId = 238 CrimeId = b97dvertexId = 374 CrimeId = 2cf9bvertexId = 426 CrimeId = b06c7vertexId = 241 CrimeId = f688b
200 vertexId = 334 CrimeId = c5c16vertexId = 22 CrimeId = 400a2vertexId = 569 CrimeId = b46afvertexId = 70 CrimeId = 4dff
201 vertexId = 311 CrimeId = 1c7evertexId = 207 CrimeId = 85357vertexId = 483 CrimeId = f2783vertexId = 211 CrimeId = 594a7vertexId = 2 CrimeId = db60f
202 vertexId = 512 CrimeId = 2bd1dvertexId = 48 CrimeId = 88656vertexId = 535 CrimeId = 594a7vertexId = 32 CrimeId = 594a7
203 vertexId = 301 CrimeId = 703bvertexId = 39 CrimeId = e5239vertexId = 406 CrimeId = 98494vertexId = 464 CrimeId = 11727vertexId = 558 CrimeId = d3b73
204 vertexId = 310 CrimeId = ccf3fvertexId = 544 CrimeId = 4fc37vertexId = 77 CrimeId = 38c6evertexId = 377 CrimeId = e9ca5
205 vertexId = 346 CrimeId = 040d0vertexId = 452 CrimeId = 29645vertexId = 588 CrimeId = a30f9vertexId = 485 CrimeId = f9816
206 vertexId = 36 CrimeId = e3b1vertexId = 280 CrimeId = ec3b1vertexId = 179 CrimeId = 6af9dvertexId = 52 CrimeId = Saad7
207 vertexId = 522 CrimeId = b737evertexId = 223 CrimeId = d6ebfvertexId = 44 CrimeId = 31838vertexId = 478 CrimeId = 5bf63
208 vertexId = 429 CrimeId = 93956vertexId = 538 CrimeId = d816evertexId = 348 CrimeId = 937c0vertexId = 502 CrimeId = 110be
209 vertexId = 63 CrimeId = b893dvertexId = 549 CrimeId = 68669vertexId = 345 CrimeId = 937c0vertexId = 502 CrimeId = 110be
210 vertexId = 63 CrimeId = b893dvertexId = 475 CrimeId = 68669vertexId = 165 CrimeId = e5df9vertexId = 314 CrimeId = 4bd67
211 vertexId = 476 CrimeId = 0bc0dvertexId = 92 CrimeId = c1fb6vertexId = 396 CrimeId = e5239vertexId = 282 CrimeId = afe1b
212 vertexId = 493 CrimeId = d76b8vertexId = 325 CrimeId = 8064fvertexId = 548 CrimeId = c516evertexId = 5 CrimeId = e93d1
213 vertexId = 547 CrimeId = ce6evertexId = 86 CrimeId = d5aebvertexId = 366 CrimeId = f22f2vertexId = 13 CrimeId = 6c3a7
214 vertexId = 522 CrimeId = 6dd4bvertexId = 344 CrimeId = 35c5cvertexId = 137 CrimeId = 4972evertexId = 99 CrimeId = 829fb
215 vertexId = 500 CrimeId = fcbbvertexId = 125 CrimeId = 9866evertexId = 585 CrimeId = 1ea0vertexId = 108 CrimeId = 497b9
216 vertexId = 284 CrimeId = 39356vertexId = 388 CrimeId = d4fbvertexId = 365 CrimeId = 7989vertexId = 151 CrimeId = 70610
217 vertexId = 44234vertexId = 505 CrimeId = 68669vertexId = 305 CrimeId = 937c0vertexId = 502 CrimeId = 110be
218 vertexId = 363 CrimeId = 8d4c3vertexId = 221 CrimeId = 35c3dvertexId = 438 CrimeId = e5239vertexId = 282 CrimeId = 110be
219 vertexId = 372 CrimeId = ca212vertexId = 507 CrimeId = 266f4vertexId = 327 CrimeId = 9790vertexId = 518 CrimeId = a02b4
220 vertexId = 428 CrimeId = c543evertexId = 355 CrimeId = 8c1e3vertexId = 344 CrimeId = 79146vertexId = 509 CrimeId = 30d1a
221 vertexId = 107 CrimeId = 480evertexId = 397 CrimeId = b560cvertexId = 265 CrimeId = 70700vertexId = 78 CrimeId = a14f0
222 vertexId = 477 CrimeId = 51a1vertexId = 257 CrimeId = bc97fvertexId = 39 CrimeId = 957vertexId = 436 CrimeId = 70610
223 vertexId = 539 CrimeId = d594vertexId = 214 CrimeId = 68669vertexId = 388 CrimeId = 937c0vertexId = 502 CrimeId = 110be
224 vertexId = 359 CrimeId = c0d92vertexId = 213 CrimeId = fd168vertexId = 288 CrimeId = faeaevertexId = 499 CrimeId = b559f
225 vertexId = 218 CrimeId = ead04vertexId = 195 CrimeId = 5b318vertexId = 355 CrimeId = daabevertexId = 229 CrimeId = c7b1e
226 vertexId = 375 CrimeId = 8f6cvertexId = 575 CrimeId = 8d338vertexId = 529 CrimeId = 78b7vertexId = 557 CrimeId = 95d5b
227 vertexId = 302 CrimeId = hh552vertexId = 176 CrimeId = a2f4vertexId = 09 CrimeId = 34h46vertexId = 557 CrimeId = 95d5b
```

MST Tour Weight(meters) = 513326.09539907286

After 3 Opt Weight = 636788.2026585272

Approx Ratio= 3 opt/MST= 636788.2026585272/513326.09539907286

Approximation Ratio=1.24

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "PSA_Project".
- TspApplication.java:** The active file contains Java code for a simulated annealing algorithm to find a TSP tour. It includes imports for various classes like MinWeightService, SimulatedAnnealing, TspApplication, etc.
- Console Output:** Displays the execution results:
 - Timestamp: 2022-09-18 17:04:40 PM (pid: 90376)
 - 3 Opt Tsp Weight in meters: 636788.2026585272
 - *****3 Opt Tsp Tour*****
 - vertexId = 0 CrimeId = 47823vertexId = 368 CrimeId = 64e2vertexId = 106 CrimeId = 74aa7vertexId = 305 CrimeId = b9086
 - vertexId = 7 CrimeId = cee3bvertexId = 540 CrimeId = c51e6vertexId = 5 CrimeId = e93d1vertexId = 418 CrimeId = d1ea8
 - vertexId = 62 CrimeId = 987b5vertexId = 112 CrimeId = f37ccvertexId = 140 CrimeId = c828avertexId = 382 CrimeId = 7773f
 - vertexId = 414 CrimeId = d4615vertexId = 22 CrimeId = 400a2vertexId = 569 CrimeId = b464fvertexId = 287 CrimeId = 85357
 - vertexId = 311 CrimeId = 1c7e4vertexId = 70 CrimeId = 4dfffevertexId = 541 CrimeId = 5439wvertexId = 68 CrimeId = 19cb3
 - vertexId = 200 CrimeId = 23017vertexId = 106 CrimeId = 2c016vertexId = 493 CrimeId = 54015vertexId = 241 CrimeId = f6683
 - vertexId = 334 CrimeId = c5192vertexId = 61 CrimeId = 3b3a6vertexId = 157 CrimeId = 7ae64vertexId = 54 CrimeId = 84cfc
 - vertexId = 324 CrimeId = 0a79bvertexId = 515 CrimeId = 8e189vertexId = 158 CrimeId = f98c7vertexId = 187 CrimeId = 8428d
 - vertexId = 445 CrimeId = ba063vertexId = 455 CrimeId = b7681vertexId = 593 CrimeId = b85d9vertexId = 446 CrimeId = 53135
 - vertexId = 400 CrimeId = 6c267vertexId = 577 CrimeId = 7283dvertexId = 504 CrimeId = faeaawvertexId = 217 CrimeId = b71c9
 - vertexId = 324 CrimeId = 70407vertexId = 188 CrimeId = a74cevertexId = 302 CrimeId = f1507vertexId = 9 CrimeId = ed4c1
 - vertexId = 336 CrimeId = 3a074vertexId = 513 CrimeId = 42ba6vertexId = 321 CrimeId = d4299vertexId = 379 CrimeId = dbd67
 - vertexId = 122 CrimeId = 1fc21vertexId = 499 CrimeId = 9e912vertexId = 387 CrimeId = a661bvertexId = 189 CrimeId = f1a4e
 - vertexId = 541 CrimeId = 54179vertexId = 514 CrimeId = 54179vertexId = 541 CrimeId = 54179vertexId = 541 CrimeId = 16
 - vertexId = 288 CrimeId = 84393vertexId = 124 CrimeId = 647a5vertexId = 237 CrimeId = 5fc4vertexId = 511 CrimeId = e8c5b
 - vertexId = 458 CrimeId = 0d210vertexId = 238 CrimeId = 46262vertexId = 190 CrimeId = 36acavertexId = 364 CrimeId = 29847
 - vertexId = 185 CrimeId = 33610vertexId = 181 CrimeId = 96d44vertexId = 114 CrimeId = bd42evertexId = 19 CrimeId = 88b5c
 - vertexId = 8 CrimeId = f4751vertexId = 511 CrimeId = 78999vertexId = 49 CrimeId = 4ddd2vertexId = 45 CrimeId = 6cb16
 - vertexId = 282 CrimeId = c80c3vertexId = 272 CrimeId = 1e779vertexId = 153 CrimeId = c726bvertexId = 167 CrimeId = 97aa5
 - vertexId = 449 CrimeId = b1b9fvertexId = 115 CrimeId = 58222vertexId = 453 CrimeId = 62a2bvertexId = 326 CrimeId = 66ceb
 - vertexId = 117 CrimeId = 117fb2vertexId = 434 CrimeId = d9336vertexId = 147 CrimeId = 771b5vertexId = 118 CrimeId = 36c1d
 - vertexId = 417 CrimeId = c0592vertexId = 434 CrimeId = d9336vertexId = 147 CrimeId = 771b5vertexId = 118 CrimeId = 36c1d
 - vertexId = 448 CrimeId = c904dvertexId = 58 CrimeId = c3fa5vertexId = 249 CrimeId = cce32vertexId = 194 CrimeId = cfec3
 - vertexId = 29 CrimeId = ba74evertexId = 347 CrimeId = ba542vertexId = 395 CrimeId = 38704vertexId = 146 CrimeId = 2b693
 - vertexId = 102 CrimeId = af637vertexId = 388 CrimeId = 61e92vertexId = 68 CrimeId = 50ed1vertexId = 267 CrimeId = 662e9
 - vertexId = 362 CrimeId = fe7aevertexId = 454 CrimeId = 8d2d7vertexId = 458 CrimeId = 801c9vertexId = 295 CrimeId = d1ba6
 - vertexId = 200 CrimeId = 140b0vertexId = 75 CrimeId = d2c1dvertexId = 437 CrimeId = 9bd4vertexId = 263 CrimeId = d9298
 - vertexId = 215 CrimeId = 215fb2vertexId = 257 CrimeId = 33686vertexId = 526 CrimeId = 33686vertexId = 526 CrimeId = d9297
 - vertexId = 3 CrimeId = add5fvertexId = a099 0 vertexId = 33686vertexId = 526 CrimeId = 33686vertexId = 526 CrimeId = 4d2c0
 - vertexId = 287 CrimeId = dded9vertexId = 499 CrimeId = efb0b1vertexId = 392 CrimeId = e1b9bvertexId = 89 CrimeId = d9c20
 - vertexId = 361 CrimeId = 6b397vertexId = 510 CrimeId = b82a2vertexId = 473 CrimeId = 09e90vertexId = 149 CrimeId = 18474
 - vertexId = 571 CrimeId = ffe9evertexId = 488 CrimeId = 2d349vertexId = 546 CrimeId = 8bfdbvertexId = 329 CrimeId = c5f90

MST Tour Weight(meters) = 513326.09539907286

After Simulated Annealing Weight = 632221.2026585272

Approx Ratio= simulated/MST= 632221.2026585272
/513326.09539907286

Approximation Ratio=1.23

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "PSA_Project [PSA_Project_main]".
- TspApplication.java:** The active file in the editor, containing code related to simulated annealing and MST tour execution times.
- Console:** Displays the output of the application, which includes:
 - INFO6205 [INFO6205 Spring2023]
 - Simulated Annealing Opt Execution Time in Millisecs: 655578
 - Simulated Annealing Opt Tour: [Output of vertex IDs]
- Code Snippet from TspApplication.java:**

```
189 // System.out.println("Ant Colony Opt Execution Time in Millisecs: " + (new Timestamp(System.currentTimeMillis())-startTime.getTime()));  
190 // List<Vertex> AntColonyTour = new ArrayList<Vertex>();  
191 // antTour.getTour().stream().forEach(e -> {  
192 //
```
- Output in Console:**

```
*****+ Simulated Annealing Tsp Tour *****+  
vertexId = 0 CrimeId = 47823 vertexId = 368 CrimeId = 64e7a vertexId = 106 CrimeId = 74aa7 vertexId = 305 CrimeId = b9086, vertexId = 7 CrimeId = cee38 vertexId = 540 CrimeId = c51e vertexId = 5 CrimeId = e93d1 vertexId = 418 CrimeId = d1eaa  
vertexId = 62 CrimeId = 987b5 vertexId = 112 CrimeId = f37c vertexId = 140 CrimeId = c828a vertexId = 382 CrimeId = 7773f  
vertexId = 414 CrimeId = d4615 vertexId = 22 CrimeId = 409a2 vertexId = 569 CrimeId = b46a7 vertexId = 287 CrimeId = 85327  
vertexId = 151 CrimeId = 3d612 vertexId = 141 CrimeId = 462a1 vertexId = 544 CrimeId = 5a40d vertexId = 193 CrimeId = 18033  
vertexId = 238 CrimeId = b97cd vertexId = 376 CrimeId = 2cf0h vertexId = 426 CrimeId = 5a6c7 vertexId = 241 CrimeId = 160b0  
vertexId = 334 CrimeId = c5c16 vertexId = 61 CrimeId = hb3a vertexId = 427 CrimeId = aae64 vertexId = 528 CrimeId = 01c6c  
vertexId = 343 CrimeId = 0a7a9 vertexId = 515 CrimeId = 8e10s vertexId = 158 CrimeId = f98e7 vertexId = 187 CrimeId = 0428d  
vertexId = 445 CrimeId = ba063 vertexId = 455 CrimeId = b7681 vertexId = 551 CrimeId = b85d9 vertexId = 441 CrimeId = 53135  
vertexId = 408 CrimeId = 6c267 vertexId = 579 CrimeId = 7283d vertexId = 54 CrimeId = faea9 vertexId = 217 CrimeId = b71c9  
vertexId = 324 CrimeId = 7040f vertexId = 186 CrimeId = af4ce vertexId = 301 CrimeId = f1507 vertexId = 9 CrimeId = ed4c1  
vertexId = 336 CrimeId = 3a074 vertexId = 513 CrimeId = 42ba0 vertexId = 321 CrimeId = d4299 vertexId = 379 CrimeId = dbd67  
vertexId = 472 CrimeId = 5a111 vertexId = 115 CrimeId = 471b1 vertexId = 115 CrimeId = 5a111 vertexId = 266 CrimeId = 144de  
vertexId = 72 CrimeId = ffb25 vertexId = 537 CrimeId = 198d0 vertexId = 451 CrimeId = 57aa7 vertexId = 21 CrimeId = bff1e  
vertexId = 288 CrimeId = 84393 vertexId = 124 CrimeId = 647a7 vertexId = 237 CrimeId = afcc4 vertexId = 511 CrimeId = e8c5b  
vertexId = 458 CrimeId = 0d210 vertexId = 230 CrimeId = 46e26 vertexId = 191 CrimeId = 36acavertexId = 364 CrimeId = 29847  
vertexId = 188 CrimeId = 3361b vertexId = 181 CrimeId = 96dd4 vertexId = 114 CrimeId = bd42e vertexId = 19 CrimeId = 88bdc  
vertexId = 84 CrimeId = f4751 vertexId = 514 CrimeId = 789d9 vertexId = 49 CrimeId = 4ddd4 vertexId = 45 CrimeId = 6cb16  
vertexId = 282 CrimeId = cbc32 vertexId = 272 CrimeId = 1e795 vertexId = 193 CrimeId = c7268 vertexId = 167 CrimeId = 97a55  
vertexId = 151 CrimeId = 3d1b1 vertexId = 215 CrimeId = 459d0 vertexId = 459 CrimeId = 5a956 vertexId = 256 CrimeId = 16609  
vertexId = 288 CrimeId = f2b87 vertexId = 81 CrimeId = 7c08b vertexId = 591 CrimeId = 7aa15 vertexId = 264 CrimeId = 1512a  
vertexId = 417 CrimeId = f2eb2 vertexId = 434 CrimeId = d9536 vertexId = 147 CrimeId = e71b5 vertexId = 118 CrimeId = 36c1d  
vertexId = 498 CrimeId = b9b4d vertexId = 58 CrimeId = c3fa7 vertexId = 249 CrimeId = cce32 vertexId = 194 CrimeId = cfec3  
vertexId = 29 CrimeId = ba74d vertexId = 347 CrimeId = ba542 vertexId = 399 CrimeId = 30704 vertexId = 146 CrimeId = 2b603  
vertexId = 182 CrimeId = 6fe37 vertexId = 386 CrimeId = 61e92 vertexId = 68 CrimeId = 50ed1 vertexId = 267 CrimeId = 662e9  
vertexId = 362 CrimeId = f7a2e vertexId = 454 CrimeId = 8d2d7 vertexId = 509 CrimeId = 801c9 vertexId = 295 CrimeId = d1ba6  
vertexId = 422 CrimeId = 3d156 vertexId = 415 CrimeId = 424c4 vertexId = 431 CrimeId = 5a111 vertexId = 266 CrimeId = 180cc  
vertexId = 271 CrimeId = 3d168 vertexId = 318 CrimeId = cfc4e vertexId = 176 CrimeId = fdcf4 vertexId = 142 CrimeId = 15b0f  
vertexId = 578 CrimeId = 83178 vertexId = 566 CrimeId = 3ee1c vertexId = 406 CrimeId = bdafb vertexId = 71 CrimeId = ba3ae  
vertexId = 136 CrimeId = 53257 vertexId = 83 CrimeId = 5da87 vertexId = 285 CrimeId = a833d vertexId = 88 CrimeId = 2d0ae  
vertexId = 79 CrimeId = 34272 vertexId = 470 CrimeId = e8c89 vertexId = 385 CrimeId = 2a25c vertexId = 351 CrimeId = d8b01  
vertexId = 199 CrimeId = 9e51c vertexId = 236 CrimeId = 4dcfa vertexId = 487 CrimeId = a7bb0 vertexId = 65 CrimeId = e85ee  
vertexId = 244 CrimeId = 44f12 vertexId = 53 CrimeId = 36ca vertexId = 541 CrimeId = cd05f vertexId = 497 CrimeId = f4a47  
vertexId = 43 CrimeId = edf2b vertexId = 8 CrimeId = 428d1 vertexId = 525 CrimeId = 601d0 vertexId = 601d0  
vertexId = 56 CrimeId = 3228b vertexId = 288 CrimeId = ec3b1 vertexId = 409 CrimeId = 5a816 vertexId = 295 CrimeId = faaf9  
vertexId = 52 CrimeId = 5aa37 vertexId = 580 CrimeId = a39f9 vertexId = 452 CrimeId = 29645 vertexId = 377 CrimeId = 0e0a5  
vertexId = 544 CrimeId = 4fc27 vertexId = 77 CrimeId = 38c6e vertexId = 346 CrimeId = 049d1 vertexId = 310 CrimeId = cc9f3  
vertexId = 558 CrimeId = 63b73 vertexId = 502 CrimeId = 11b0e vertexId = 348 CrimeId = 937c0 vertexId = 429 CrimeId = b37e5  
vertexId = 223 CrimeId = d6ebf vertexId = 478 CrimeId = 5bf3e vertexId = 44 CrimeId = 31838 vertexId = 459 CrimeId = 93956
```

MST Tour Weight(meters) = 513326.09539907286

After Ant Colony opt Weight = 632221.2026585272

Approx Ratio= Ant/MST= 632115.7917132177/513326.09539907286

Approximation Ratio=1.232

INFO6205 [INFO6205 Spring2023]

psa

PSA_Project [PSA_Project main]

src/main/java

edu.neu.coe.csye6205.tsp

TspApplication.java

edu.neu.coe.csye6205.tsp.controller

edu.neu.coe.csye6205.tsp.model

Ant.java

Graph.java

GraphDto.java

MstTour.java

Point.java

Route.java

TspTour.java

Vertex.java

edu.neu.coe.csye6205.tsp.repository

edu.neu.coe.csye6205.tsp.service

AntColonyOptimizationService.java

CsvUtilService.java

ExcelFetchPointService.java

ExcelTspRouteService.java

LinkMichigan20pt.java

MinWeightService.java

RouteWeightService.java

SimulatedAnnealingService.java

ThreeOptService.java

TwoOptService.java

edu.neu.coe.csye6205.tsp.util

src/main/resources

src/test/java

JRE System Library [JavaSE-17]

Maven Dependencies

bin

Data

src

target

.spoints.xlsx

.points.xlsx

pom.xml

README.md

routes.xlsx

MinWeightService.java [SimulatedAnneal, TspApplication, TspApplicationT, UnionFind.java, AntColonyOptimi..., Ant.java, README.md] *15

148 tspOptTour.getTour().stream().forEach(e ->

149 Vertex v = new Vertex(e, vertexMap.get(e));

150 opt2Vertextour.add(v);

151 });

152 System.out.println(" 2 Opt Tsp Weight in meters " + tspOptTour.getLength() * 1000);

153 System.out.println("*****2 Opt Tsp Tour *****");

154 System.out.println(printUtil(opt2Vertextour));

155 System.out.println("*****2 Opt Tsp Tour *****");

***** Ant Colony Opt Execution Time in MilliSecs: 189356

Ant Colony opt Tsp Tour vertexId = 0 CrimeId = 6232 vertexId = 5232 vertexId = 5232

Ant Colony opt Tsp Tour [vertexId = 0 CrimeId = 47823, vertexId = 141 CrimeId = ea746, vertexId = 313 CrimeId = 916a4, vertexId = 554 CrimeId = f384c, vertexId = 175 CrimeId = 6b010vertexId = 334 CrimeId = 7d25bvertexId = 527 CrimeId = 9182vertexId = 73 CrimeId = 6774a

vertexId = 539 CrimeId = bf56dvertexId = 328 CrimeId = 59advertexId = 117 CrimeId = fe0d5vertexId = 309 CrimeId = 22d14

vertexId = 269 CrimeId = 5b793vertexId = 31 CrimeId = 9db58vertexId = 55 CrimeId = 02857vertexId = 227 CrimeId = 3ce3f

vertexId = 465 CrimeId = 39747vertexId = 301 CrimeId = 13e20vertexId = 214 CrimeId = 71eabvertexId = 192 CrimeId = 8951a

vertexId = 345 CrimeId = 29949vertexId = 176 CrimeId = 13e20vertexId = 214 CrimeId = 71eabvertexId = 192 CrimeId = 8951a

vertexId = 480 CrimeId = 84390vertexId = 244 CrimeId = 64746vertexId = 237 CrimeId = 8ad98vertexId = 158 CrimeId = b594

vertexId = 511 CrimeId = e8c52vertexId = 230 CrimeId = 46e26vertexId = 198 CrimeId = 36cavertexId = 558 CrimeId = eas9b

vertexId = 47 CrimeId = ad1c1vertexId = 168 CrimeId = da01lvertexId = 368 CrimeId = 8f8b5vertexId = 139 CrimeId = 7804a

vertexId = 232 CrimeId = 6ed74vertexId = 466 CrimeId = a7556vertexId = 182 CrimeId = c3355vertexId = 183 CrimeId = e9a50

vertexId = 584 CrimeId = 98ae6vertexId = 299 CrimeId = 8c8levertexId = 174 CrimeId = 1eb20vertexId = 161 CrimeId = 3f416

vertexId = 250 CrimeId = cbe84vertexId = 191 CrimeId = 10777vertexId = 543 CrimeId = 92526vertexId = 399 CrimeId = 681a5

vertexId = 447 CrimeId = 874e3vertexId = 261 CrimeId = e9727vertexId = 143 CrimeId = c8783vertexId = 342 CrimeId = e8d70

vertexId = 428 CrimeId = 10242vertexId = 203 CrimeId = 10242vertexId = 143 CrimeId = c8783vertexId = 342 CrimeId = e8d70

vertexId = 93 CrimeId = 93619vertexId = 131 CrimeId = f118bvertexId = 99 CrimeId = a8047vertexId = 408 CrimeId = 71a9

vertexId = 389 CrimeId = 8e3a9vertexId = 183 CrimeId = 49fc4vertexId = 545 CrimeId = 7fc2dvertexId = 159 CrimeId = 5db1a

vertexId = 97 CrimeId = 7fe42vertexId = 519 CrimeId = 10c55vertexId = 34 CrimeId = 573b3vertexId = 373 CrimeId = 12dd6

vertexId = 532 CrimeId = 738bavertexId = 533 CrimeId = 2cc42vertexId = 487 CrimeId = 29946vertexId = 418 CrimeId = 810b6

vertexId = 104 CrimeId = 50735vertexId = 247 CrimeId = fd53cvertexId = 98 CrimeId = 7df2evertexId = 466 CrimeId = 0c987

vertexId = 168 CrimeId = 2b0f0vertexId = 82 CrimeId = 61a40vertexId = 559 CrimeId = 90125vertexId = 234 CrimeId = 486aa

vertexId = 41 CrimeId = 69178vertexId = 474 CrimeId = 394d7vertexId = 296 CrimeId = 99429vertexId = 441 CrimeId = 7d2d7

vertexId = 53 CrimeId = 10242vertexId = 203 CrimeId = 10242vertexId = 143 CrimeId = c8783vertexId = 342 CrimeId = 672a2

vertexId = 33 CrimeId = 31888vertexId = 93 CrimeId = 5f5c5vertexId = 543 CrimeId = 959abvertexId = 216 CrimeId = 656ff

vertexId = 388 CrimeId = c2694vertexId = 37 CrimeId = d102dvertexId = 378 CrimeId = c286evertexId = 469 CrimeId = 4fa9b

vertexId = 284 CrimeId = 83996vertexId = 69 CrimeId = 09868vertexId = 471 CrimeId = a471avertexId = 312 CrimeId = c9b73

vertexId = 50 CrimeId = 436eevertexId = 253 CrimeId = dg972vertexId = 564 CrimeId = 0637hvertexId = 276 CrimeId = 17724

vertexId = 319 CrimeId = 33902vertexId = 415 CrimeId = dd03bvertexId = 166 CrimeId = 5ff6bvertexId = 457 CrimeId = e1cd5

vertexId = 289 CrimeId = 9fe9avertexId = 388 CrimeId = af279vertexId = 242 CrimeId = 4ee9avertexId = 367 CrimeId = cd848

vertexId = 488 CrimeId = 5570fvertexId = 163 CrimeId = 000deverteId = 304 CrimeId = b1f03vertexId = 197 CrimeId = bad43

vertexId = 160 CrimeId = 10242vertexId = 203 CrimeId = 10242vertexId = 143 CrimeId = c8783vertexId = 342 CrimeId = 4789f

vertexId = 322 CrimeId = 45495vertexId = 54 CrimeId = 54de4vertexId = 3 CrimeId = 40999vertexId = 370 CrimeId = 622

vertexId = 320 CrimeId = 4fb49vertexId = 259 CrimeId = f0199vertexId = 14 CrimeId = 77115vertexId = 277 CrimeId = 7b357

vertexId = 316 CrimeId = 4cfc86vertexId = 333 CrimeId = bc3d3vertexId = 412 CrimeId = f5894vertexId = 16 CrimeId = 26d4a

vertexId = 198 CrimeId = 69847vertexId = 28 CrimeId = 31a00vertexId = 154 CrimeId = 515b5vertexId = 27 CrimeId = 7ac7b

vertexId = 442 CrimeId = 3f894vertexId = 148 CrimeId = e924dvertexId = 337 CrimeId = aa7bavertexId = 357 CrimeId = 49896

vertexId = 297 CrimeId = a38d8vertexId = 420 CrimeId = ccae0vertexId = 563 CrimeId = 15c85vertexId = 222 CrimeId = d4d65

vertexId = 66 CrimeId = ceae0vertexId = 110 CrimeId = 7813lvertexId = 369 CrimeId = 1a8d0vertexId = 10 CrimeId = f5d8a

vertexId = 555 CrimeId = 395c9vertexId = 180 CrimeId = 22e22vertexId = 170 CrimeId = 22e48vertexId = 76 CrimeId = c232d

vertexId = 113 CrimeId = 6c07dvertexId = 464 CrimeId = 94e98vertexId = 67 CrimeId = 1c000vertexId = 567 CrimeId = 567a

vertexId = 161 CrimeId = 16153vertexId = 239 CrimeId = 98awvertexId = 177 CrimeId = 31fdvertexId = 438 CrimeId = 969b

vertexId = 524 CrimeId = 8c7b9vertexId = 339 CrimeId = cad0cvertexId = 496 CrimeId = 9be80vertexId = 572 CrimeId = 9793

vertexId = 335 CrimeId = 6e85avertexId = 484 CrimeId = 144e9vertexId = 222 CrimeId = 9e57lvertexId = 530 CrimeId = 0b647

Unit Tests:

Below are the Unit Tests results along with Code ScreenShots :

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Shows the Eclipse logo, menu items like File, Edit, View, Search, etc., and a search bar.
- Left Sidebar:** "Package Explorer" view showing a tree structure of Java packages and classes. Under the package `TspApplicationTests`, there are several test methods listed with their execution times: `Opt2Test()` (0.000 s), `Opt3Test()` (0.000 s), `findMinimumWeightMatchingTest()` (0.217 s), `minimumSpanningTreeTest()` (0.008 s), `getTSPPathAndWeightTest()` (0.008 s), `findEulerianTourTest()` (0.003 s), and `OddDegreeVertexTest()` (0.002 s).
- Middle Area:** The main workspace displays the source code for the `TspApplicationTests` class. The code contains various JUnit test cases for graph-related operations like minimum spanning trees, odd-degree vertex detection, and Eulerian tours.
- Bottom Bar:** Shows the "Console" tab open, displaying log output from the tests. The log includes entries for DEBUG and INFO levels from the Spring framework and the application's service layer.

The screenshot shows the Eclipse IDE interface with several open windows:

- Package Explorer**: Shows the project structure with **TspApplicationTests** as the active runner.
- JUnit X**: Displays the test results: Finished after 2.302 seconds, 7/7 runs, 0 errors, 0 failures.
- MinWeightService**, **CsvUtilService**, **TspApplicationT**, **LinKernighan2Op**, **ExcelFetchPoint**, **RouteWeightServ**, **UnionFind.java**: These are other Java files in the project.
- TspApplicationTests [Runner: JUnit 5] (0.259 s)**: The current test runner window.
- Opt2Test() (0.000 s)**, **Opt3Test() (0.000 s)**, **findMinimumWeightMatchingTest() (0.217 s)**, **minimumSpanningTreeTest() (0.008 s)**, **getTspPathAndWeightTest() (0.008 s)**, **findEulerianTourTest() (0.003 s)**, **OddDegreeVertexTest() (0.002 s)**: Test cases listed under the runner.
- Failure Trace**: A window showing the failure trace for the tests.
- Console**: Shows standard output and error messages from the tests.

The code editor displays the **TspApplicationTests** file with various test methods and their implementations. The code uses Graph DTOs and various service interfaces like **MinWeightService** and **TspOpt2Service**.

Conclusion:

In conclusion, our project utilized various algorithms and techniques to solve the Traveling Salesman Problem (TSP) for a given dataset of vertices. We first processed the dataset and computed the distances between each vertex. Next, we implemented Kruskal's Union Find Method to identify the minimum spanning tree (MST) of the graph. We then extracted the list of odd vertices from the MST Tour and identified the perfect matching for them using the minimum weight matching method. We combined the Perfect Matching and MST to generate an Eulerian tour of the graph, which we used to create a TSP tour that visited each vertex and returned to the starting vertex. The resulting TSP tour was optimized using the 2-opt and 3-opt techniques, as well as strategic optimizations like simulated annealing and ant colony optimization.

Our project achieved a minimum spanning tree value of 513326 and a TSP tour distance of 698925. After applying the 2-opt and 3-opt techniques, the TSP tour distance was further reduced to 637578 and 636788, respectively. The strategic optimization techniques of simulated annealing and ant colony optimization also contributed to reducing the TSP tour distance, resulting in values of 632221 and 632115, respectively.

Overall, our project demonstrated the effectiveness of various algorithms and techniques in solving the Traveling Salesman Problem, and highlighted the importance of optimization in achieving a more efficient and accurate solution.

References:

- <https://www.youtube.com/watch?v=GiDsjlBOVoA&t=396s>
- https://en.wikipedia.org/wiki/Christofides_algorithm
- https://en.wikipedia.org/wiki/Travelling_salesman_problem
- https://en.wikipedia.org/wiki/Simulated_annealing
- <https://slowandsteadybrain.medium.com/traveling-salesman-problem-ce78187cf1f3>