

Travelling Salesman Project

Submitted By:

Mayur Kenkre : 002771727

Naman Diwan : 002724115

INTRODUCTION:

The Traveling Salesman Problem (TSP) is a classic problem in computer science and optimization theory that involves finding the shortest possible route for a salesman to visit a set of cities and return to the starting city. It is an NP-hard problem, which makes finding the optimal solution exponentially difficult as the number of cities increases. The TSP has important applications in logistics, transportation, and supply chain management, with algorithms like brute-force methods, dynamic programming, nearest neighbor algorithm, and genetic algorithm being developed to solve it and provide approximate solutions for large-scale instances.

Aim:

The project aims to develop an efficient algorithm to solve the TSP, utilizing the Christofides algorithm as a basis for the solution. This heuristic guarantees a solution no worse than $3/2$ times the optimal solution. In addition, various tactical and strategic methods such as random swapping, 2-opt and 3-opt improvement, simulated annealing, ant colony optimization, genetic algorithms, etc., will be employed to further optimize the solution. The goal is to create an algorithm that can find high-quality solutions to the TSP in a reasonable amount of time for real-world applications.

Approach:

- The implementation of the Christofides algorithm for the TSP involved the following steps:
- Creating a minimum spanning tree (MST) using Prim's algorithm on the given set of cities.
- Identifying all vertices with odd degree in the MST and forming a minimum-weight perfect matching between them using the Hungarian algorithm.
- Combining the MST and the perfect matching to form a graph with all vertices having even degree.
- Finding an Eulerian circuit in the new graph.
- Converting the Eulerian circuit into a Hamiltonian circuit by skipping previously visited vertices.

Step 1 of the Christofides algorithm creates a minimum spanning tree using Prim's algorithm, which guarantees that the weight of the spanning tree is equal to or less than the optimal solution weight. Step 2 ensures that all vertices in the minimum spanning tree have an even degree, which is essential for a Eulerian circuit. In Step 3, the MST and the perfect matching are combined to form a new graph in which all vertices have even degree, enabling the discovery of a Eulerian circuit in Step 4. The Eulerian circuit visits every edge once, and in Step

5, it is converted to a Hamiltonian circuit by skipping previously visited vertices. As a result, the Christofides algorithm provides an effective solution to the TSP that balances efficiency and solution quality, making it a widely used approach for practical problem-solving.

For our project, we compared our Travelling salesman problem approach using 2 tactical and 2 strategic algorithms:

Tactical:

1. **2-opt:** Tactical 2-opt optimization is a heuristic algorithm that is commonly used to solve optimization problems, especially those related to the traveling salesman problem (TSP). The TSP involves finding the shortest route that visits a set of cities and returns to the starting city.

The algorithm operates by removing two edges from the current solution and reconnecting them in a different way. It examines all possible pairs of edges that can be removed and chooses the pair that results in the greatest improvement in the solution's length. The term "tactical" is used because the algorithm can be customized to consider additional information about the problem, such as the location and order of the cities, to guide its search and enhance its efficiency.

Tactical 2-opt optimization is a powerful approach for solving optimization problems, particularly those associated with the TSP. By iteratively improving the solution through local search, the algorithm can rapidly converge on an optimal or near-optimal solution.

2. **Random Swap:** Random swap is a simple heuristic algorithm used for optimization problems. It works by randomly swapping two elements in a given solution, and then evaluating whether the new solution is better than the original one.

The algorithm is commonly used in optimization problems where the goal is to find the optimal permutation of a set of elements. For example, in the context of the traveling salesman problem, the algorithm can be used to randomly swap two cities in a given solution and then evaluate whether the new solution has a shorter route.

The algorithm is typically repeated many times, with each iteration producing a new solution by randomly swapping two elements. If the new solution is better than the previous one, it is accepted and becomes the current solution. If it is worse, it is rejected and the current solution remains unchanged.

Random swap is a simple and fast algorithm that can be effective in certain types of optimization problems, particularly those involving permutations. However, it is not guaranteed to find the optimal solution and may get stuck in local optima. Therefore, it is often used as part of a larger optimization strategy, such as a genetic algorithm or simulated annealing, to explore the search space more thoroughly.

Strategic:

1. Simulated Annealing:

Simulated annealing is a metaheuristic algorithm used for optimization problems. It is particularly useful for solving problems where the search space is large and the optimal solution is difficult to find using traditional search techniques.

The algorithm is inspired by the process of annealing in metallurgy, where a metal is heated and then slowly cooled to improve its properties. In simulated annealing, the search process begins by randomly generating an initial solution. The algorithm then evaluates the solution and uses a "temperature" parameter to control the probability of accepting a worse solution. At high temperatures, the algorithm is more likely to accept a worse solution, whereas at low temperatures, it becomes increasingly difficult to accept worse solutions.

During the search process, the temperature is gradually reduced over time, allowing the algorithm to escape local optima and converge on a better solution. The cooling schedule, or rate at which the temperature decreases, is an important parameter that can affect the quality of the solution obtained.

Simulated annealing has been applied to a wide range of optimization problems, including the traveling salesman problem, graph coloring, and protein folding. It is a powerful technique that can find near-optimal solutions in a reasonable amount of time, even for very large search spaces. However, it does not guarantee finding the global optimum, and the quality of the solution obtained can be sensitive to the choice of parameters and cooling schedule.

2. Ant Colony Algorithm:

Ant Colony Optimization (ACO) is a metaheuristic algorithm that mimics the foraging behavior of ants to solve optimization problems related to routing or scheduling. ACO works by simulating the movement of ants through a search space. The ants randomly generate an initial set of solutions and move from one solution to another based on the strength of the pheromone trails. The stronger the trail, the more likely it is for an ant to choose that solution. However, the ants also introduce randomness to encourage exploration. As the ants move through the search space, they deposit pheromone trails that represent the quality of the solution they find. The pheromone trails are updated based on the quality of the solution, so that better solutions have stronger pheromone trails. The algorithm continues until the pheromone trails converge on a good solution.

ACO has been used to solve various optimization problems, such as the traveling salesman problem, vehicle routing problem, and scheduling problems. It is a powerful algorithm that can find near-optimal solutions in a reasonable amount of time, particularly for problems with large search spaces or complex constraints. However, the quality of the solution obtained may be sensitive to the selection of parameters and the size of the ant colony.

We will evaluate the performance of each algorithm by comparing its solution to known optimal solutions or by analyzing its time and space complexity.

Program: 2 Opt:

The program is a Java implementation of the 2-opt heuristic algorithm that takes a Hamiltonian circuit and a distance matrix as input, and outputs an optimized Hamiltonian circuit and the total distance covered.

Data Structures & classes:

- List<Integer>: Used to store the Hamiltonian circuit as a list of node indices.
- double[][]: Used to store the distance matrix, where distanceMatrix[i][j] represents the distance between node i and node j.
- ParseCSV: A utility class that is used to parse the input data.

Algorithm: The program implements the 2-opt heuristic algorithm to improve the given Hamiltonian circuit. The algorithm works as follows:

- Make a copy of the input circuit to a new list.
- Enter a loop that continues until no further improvement can be made.
- Within each loop iteration, iterate over each pair of edges in the circuit and check if swapping them would result in a shorter total distance.
- If a shorter distance is found, use the reverseSublist method to reverse the order of nodes between the two edges in the circuit.
- Output the optimized circuit and the total distance covered.

Invariants:

- The input Hamiltonian circuit and the output optimized circuit have the same number of nodes.
- The input distance matrix is a square matrix with non-negative values along the diagonal and symmetric entries.
- The distance between a node and itself is 0, i.e., distanceMatrix[i][i] = 0 for all i.

Program: Tactical Random Swap:

This program implements the Tactical Random Swap optimization algorithm for finding the optimal path in a traveling salesman problem. The algorithm takes a Hamiltonian path as input, along with a distance matrix and the number of iterations to perform, and returns an optimized path using random swaps of two nodes.

Data Structures & classes: The program uses the following data structures and classes:

- List<Integer>: A list of integers representing a Hamiltonian path.
- double[][]: A two-dimensional array representing the distance matrix of the graph.

Algorithm: The algorithm works as follows:

- Create a copy of the Hamiltonian path to be optimized.
- Initialize the best distance to the total distance of the Hamiltonian path.
- For the specified number of iterations, do the following:
 - Generate two random indices within the path.
 - Swap the elements at these indices.
 - Calculate the new distance of the path.

- If the new distance is better than the best distance, update the best distance and the optimized path.
- Return the optimized path.

Invariants:

- The input Hamiltonian path is not modified.
- The output path is a valid Hamiltonian path.
- The distance between any two nodes in the optimized path is less than or equal to the distance between the same two nodes in the input path.
- The time complexity of the algorithm is $O(\text{iterations} * n)$, where n is the number of nodes in the path, since it performs a constant number of operations per iteration.

Program: Simulated Annealing

This program implements the Simulated Annealing algorithm to optimize a given Hamiltonian circuit in a graph. The Hamiltonian circuit is represented as a list of vertices. The program uses a distance matrix to calculate the distance between any two vertices.

Data Structures & classes: The program uses the following data structures and classes:

- `List<Integer>`: represents the Hamiltonian circuit
- `double[][]`: represents the distance matrix

Algorithm: The algorithm used in the program is the Simulated Annealing algorithm. It starts with an initial Hamiltonian circuit and a high temperature. It then randomly swaps two vertices in the circuit and calculates the new distance. If the new distance is better, the swap is accepted. If the new distance is worse, the swap is accepted with a certain probability. The probability of accepting a worse swap decreases as the temperature decreases. The temperature is cooled down in each iteration until it reaches a low value. The final Hamiltonian circuit is returned.

Invariants:

- The Hamiltonian circuit should not be null.
- The distance matrix should not be null and should have the same number of rows and columns.
- The temperature should be greater than 0.
- The cooling rate should be between 0 and 1.
- The number of swaps per temperature should be greater than 0.

Program: Ant Colony Optimization

The code is an implementation of Ant Colony Optimization algorithm to solve the Traveling Salesman Problem (TSP).

Data Structures & classes:

- `StrategicAntColonyOpt` - main class that contains the implementation of the algorithm.
- `ParseCSV` - utility class to parse CSV files.
- `List<Integer>` - list to hold the hamiltonian path and best solution.

- double[][] - matrix to hold the distances and pheromone values.
- ArrayList - used to hold the ant paths and ant path costs.

Algorithm:

- Initialize the pheromone matrix with a constant value.
- Set the best solution cost to be maximum.
- Repeat for a maximum number of iterations:
 - a. Construct ant paths:
 - i. For each ant, start from the first node in the hamiltonian path.
 - ii. For each subsequent node, choose the next node based on the pheromone and heuristic values.
 - iii. Store the path and its cost.
 - b. Evaporate the pheromone on each edge in the pheromone matrix.
 - c. Update the pheromone on each edge in the pheromone matrix based on the ant paths.
 - d. If an ant path has a lower cost than the best solution cost, update the best solution.
- Output the best solution.

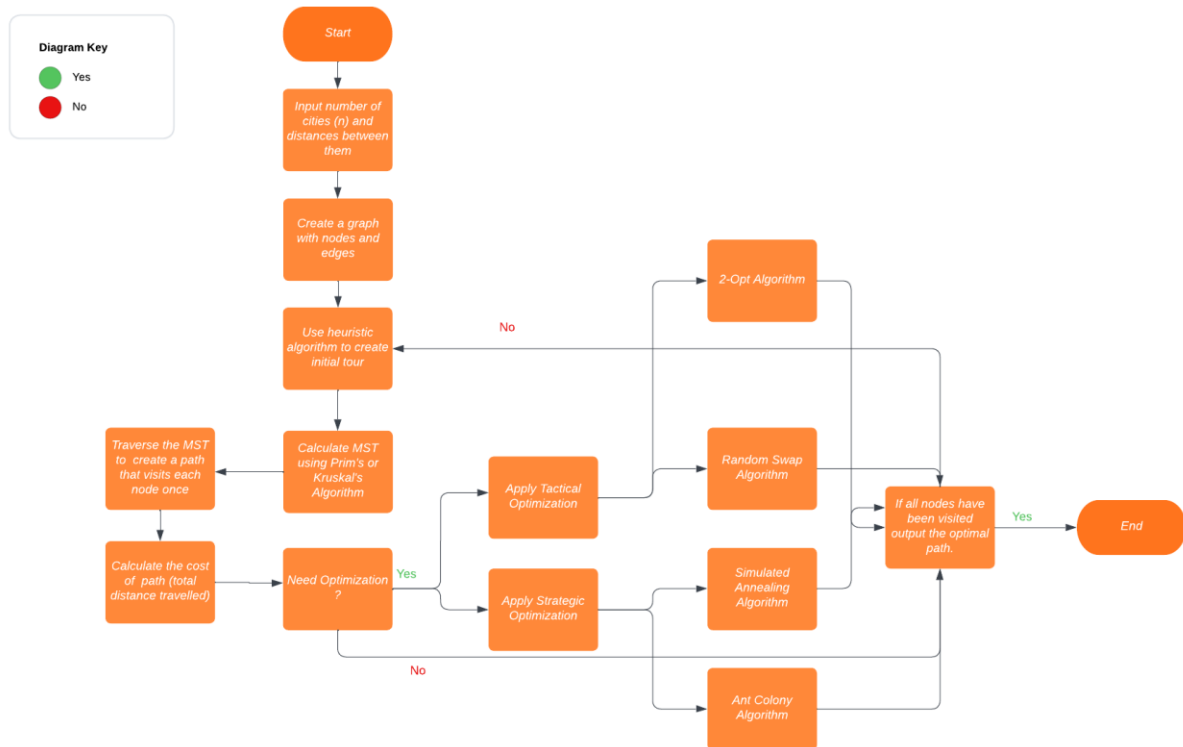
Invariants:

- The hamiltonian path and distance matrix must be provided as inputs.
- The pheromone importance factor (ALPHA), heuristic information importance factor (BETA), pheromone evaporation rate (RHO), pheromone deposit amount (Q), and number of ants (NUM_ANTS) are constant values.
- The pheromone matrix is initialized with a constant value of 1.0.
- The best solution cost is initially set to the maximum possible value.
- The ant paths are constructed by starting at the first node in the hamiltonian path and choosing the next node based on the pheromone and heuristic values.
- The pheromone on each edge in the pheromone matrix is evaporated by a constant rate (RHO) and updated based on the ant paths.
- The best solution is updated if an ant path has a lower cost than the current best solution cost.

FLOW CHARTS

Travelling salesman problem flow chart:

Travelling Salesman Problem

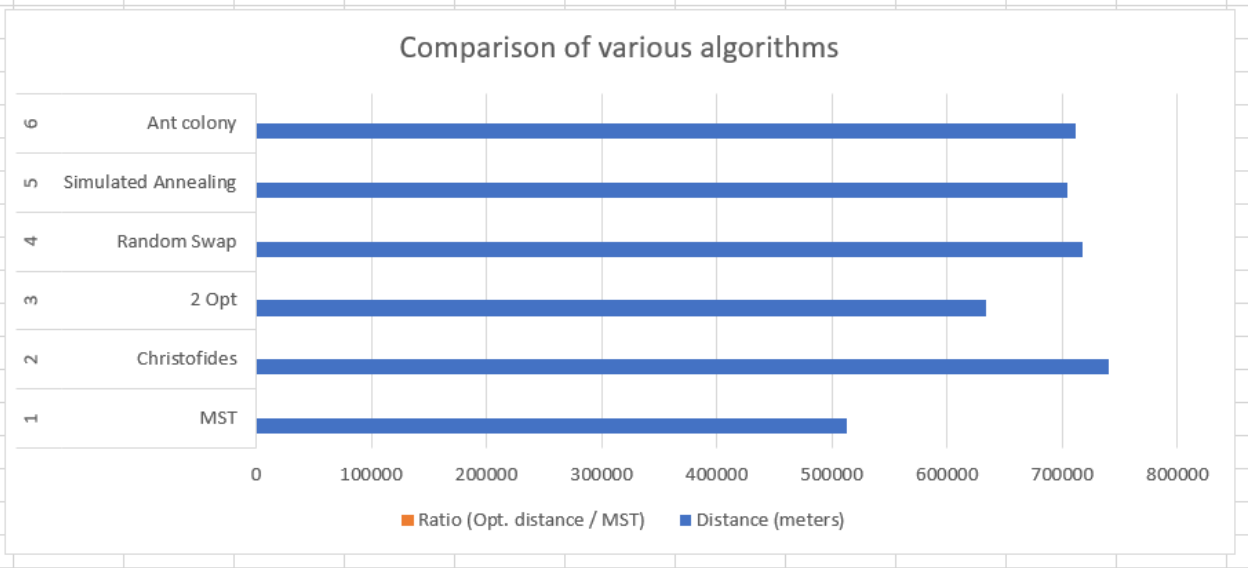


OBSERVATIONS AND GRAPHICAL ANALYSIS

Serial No.	Algorithm	Distance (meters)	Ratio (Opt. distance / MST)
1	MST	513326.09	1
2	Christofides	740861.13	1.443256332
3	2 Opt	633321.84	1.233761253
4	Random Swap	717689.96	1.398117053
5	Simulated Annealing	704369.23	1.372167212
6	Ant colony	711864.83	1.386769237

Here, we can see that 2 Opt algorithm is the best optimization algorithm which generates the shortest path.

Below graph represents the comparison of various algorithms:



RESULTS AND MECHANICAL ANALYSIS

Outputs:

Minimum Spanning Tree (MST):

- 51,51 - 177,51 - 430,430 - 524,524 - 371,371 - 164,371 - 339,339 - 496,496 - 335,335 - 572,4 - 215,215 - 178,178 - 133,133 - 322,164 - 484,484 - 222,322 - 560,560 - 273,468 - 93,93 - 433,327 - 507,507 - 372,186 - 382,382 - 9,9 - 336,336 - 513,513 - 321,321 - 379,379 - 122,122 - 498,382 - 318,318 - 171,171 - 293,293 - 403,403 - 15,15 - 423,423 - 299,299 - 134,134 - 584,584 - 329,584 - 488,488 - 571,571 - 149,149 - 473,571 - 89,423 - 356,89 - 392,329 - 546,473 - 510,510 - 361,392 - 287,287 - 561,561 - 409,287 - 490,409 - 11,561 - 24,11 - 424,424 - 556,424 - 437,437 - 75,75 - 200,154 - 20,427 - 414,414 - 382,382 - 22,22 - 569,22 - 376,376 - 426,426 - 241,241 - 334,376 - 238,238 - 60,60 - 541,541 - 470,470 - 79,79 - 385,385 - 88,88 - 285,285 - 83,83 - 71,71 - 136,71 - 406,406 - 566,406 - 578,578 - 142,142 - 176,154 - 390,390 - 216,216 - 388,216 - 378,378 - 469,378 - 37,37 - 316,316 - 333,333 - 412,412 - 277,60 - 70,70 - 311,378 - 284,284 - 69,277 - 16,397 - 265,265 - 78,78 - 477,477 - 252,69 - 471,471 - 312,312 - 253,253 - 564,253 - 60,564 - 276,276 - 319,319 - 562,562 - 96,96 - 266,319 - 144,96 - 233,233 - 275,275 - 48,48 - 549,549 - 271,271 - 6,6 - 534,534 - 482,482 - 74,74 - 260,260 - 565,271 - 574,574 - 408,408 - 118,118 - 147,147 - 434,434 - 264,264 - 417,574 - 58,484 - 248,222 - 530,530 - 220,220 - 130,130 - 105,105 - 463,463 - 228,228 - 380,380 - 383,383 - 330,330 - 535,535 - 552,552 - 80,80 - 126,126 - 393,393 - 557,557 - 575,575 - 375,375 - 229,557 - 529,220 - 212,212 - 18,18 - 201,18 - 26,26 - 411,411 - 189,189 - 219,201 - 354,411 - 87,569 - 207,207 - 2,2 - 425,425 - 341,425 - 32,341 - 506,2 - 211,211 - 40,40 - 512,512 - 42,42 - 483,32 - 501,501 - 95,95 - 494,494 - 81,81 - 567,567 - 91,200 - 295,295 - 450,450 - 454,454 - 362,362 - 267,267 - 102,102 - 386,564 - 415,415 - 166,166 - 457,7 - 140,140 - 112,112 - 62,62 - 418,418 - 5,5 - 325,325 - 547,325 - 86,86 - 366,366 - 13,13 - 522,522 - 137,137 - 99,522 - 349,88 - 351,351 - 199,199 - 236,236 - 487,229 - 355,355 - 195,225 - 369,369 - 10,10 - 66,66 - 110,264 - 59,5 - 540,487 - 43,43 - 8,8 - 155,86 - 493,493 - 63,63 - 475,475 - 165,165 - 314,63 - 28,277 - 259,259 - 320,259 - 14,320 - 370,370 - 328,188 - 438,556 - 257,494 - 502,502 - 558,502 - 348,348 - 536,536 - 459,348 - 429,429 - 223,252 - 391,391 - 436,266 - 249,249 - 194,314 - 476,219 - 526,526 - 254,254 - 491,491 - 129,99 - 500,500 - 125,125 - 583,583 - 100,100 - 204,204 - 306,306 - 152,152 - 151,151 - 365,188 - 221,221 - 363,363 - 121,121 - 401,401 - 542,542 - 173,173 - 308,308 - 35,35 - 196,476 - 396,396 - 202,436 - 440,440 - 374,362 - 68,199 - 65,65 - 244,244 - 53,53 - 548,548 - 497,558 - 310,310 - 377,377 - 77,77 - 544,77 - 452,377 - 346,452 - 580,546 - 405,405 - 179,405 - 36,36 - 280,194 - 29,355 - 258,258 - 213,213 - 359,328 - 489,489 - 332,489 - 367,367 - 242,367 - 480,480 - 304,480 - 163,304 - 197,480 - 127,127 - 3,3 - 279,3 - 505,197 - 350,550 - 64,64 - 184,184 - 145,145 - 256,64 - 132,132 - 231,231 - 224,224 - 17,17 - 12,12 - 203,203 - 274,274 - 251,251 - 389,184 - 416,12 - 268,268 - 245,245 - 570,570 - 398,570 - 404,404 - 94,404 - 340,268 - 208,94 - 193,193 - 270,270 - 23,257 - 315,315 - 263,580 - 52,155 - 525,242 - 384,384 - 209,16 - 198,223 - 44,44 - 478,585 - 581,476 - 92,332 - 443,443 - 123,123 - 495,495 - 243,243 - 576,243 - 520,520 - 444,444 - 461,461 - 462,462 - 283,462 - 538,461 - 128,128 - 551,551 - 381,381 - 57,57 - 479,479 - 448,448 - 394,394 - 439,448 - 323,355 - 499,499 - 218,581 - 303,303 - 101,101 - 331,374 - 486,102 - 146,146 - 395,395 - 347,486 - 294, Total distance covered: 513326.0953998729

Christofides:

7283d, b71c9, 6c267, 53135, b85d9, 7040f, af4ce, f1507, c15c0, 421bf, ffe9e, d8c20, e1b9b, dded9, ef8b1, dded9, 4d2c0, ca598, 4d2c0, f3ee1, a0def, 76697, d9298, 43aa8, 0c956, 3a496, 76697, 9bdee, d2c1d, 14b0b, d1ba6, 801c9, 8d2d7, fe7ae, 50ed1, fe7ae, 662e9, af637, 61e92, af637, 2b603, 30704, ba542, 6b397, b82a2, 09e90, 18474, ffe9e, 2d349, e701b, 8bd9b, c5f98, e701b, cd9cf, 2ccf2, c15c0, cc04c, 595f8, 803cc, 3d168, cfcbe, f1507, ed4c1, 3a074, 42ba6, d4299, dbd67, 1fc21, 9e912, 7cd8b, f2fb7, 66ceb, 62a5f, 58d22, c726b, a661b, f1a4e, ffb25, 198db, bff1e, 57aa7, 1b369, 0ad98, b5bdf, b58c4, 0ad98, 1b369, ae1e1, 29943, 98ae6, e9a50, c3355, a75ae, c3355, cbe84, 3f416, 1eb20, 8e81e, 50735, fd53c, 2bf09, 90125, 2bf09, 7df02, 0c987, 7d2d7, d24e7, 3f4f6, 116db, 67e12, ca409, fd49e, 0ba06, 9302e, 1f4d0, a4cdd, 99d29, ca409, 67e12, 3f884, 5f5cc, 67275, ba8bd, c3faf, ba8bd, c9b4d, 36c1d, e71b5, d9536, 1512a, 74a15, 1512a, ceab2, ba74e, cfec3, cce32, 861e4, 79845, a47d7, 33902, 0637b, 0db30, 5ffeb, e1cd5, 4fa0b, 2cc86, d182d, 4cfc86, bc3a7, f5094, 7b357, 09a87, 26d4a, 7b357, 0f199, 97f15, 0f199, 4af3b, 8b962, d0993, 54e6b, ab447, 15f40, 1c795, 5099b, 1d259, fdde0, fe864, fdde0, 470f9, 55f0d, f08de, cd848, 55f0d, bf103, bad43, 15e84, d0c60, 292c6, 36116, 0ee10, 27e0c, 292c6, d0c60, 12325, e53f8, bbb28, 86e93, b4b46, 96939, bebb5, 9b30d, 5e408, 9b30d, 61f08, 3ffb2, 56bcd, 90290, 87975, 8e6f9, 61f08, e280f, a8c25, 8ad97, 23239, b4b46, 96939, a7ebe, 9fe9a, af279, 4ee9a, cd848, 54e6b, f4456, 99bdc, ec311, 4521a, c59bd, 5da55, c59bd, 48feb, 6616d, 41072, 6389b, 7a64e, b50ca, 7a64e, 41072, 5fe9b, 814da, 3849e, 7713f, b61cf, e71b5, 19884, e6080, e71b5, b465f, ba3e8, a517b, e2d42, c4459, a517b, 0598c, 04f3f, f7f6a, 3ff9d, bd449, 824ba, ead04, b559f, da0ba, faea3, fdf68, c8da2, 5b318, da0ba, c7b1e, 8f6cc, 8d330, 95d5b, 78be7, 95d5b, b562, 03f24, 3db46, e228c, 527ca, e70a8, 63f62, 6742c, a7bac, 3ba9e, 7367c, cbe21, fd1d4, 144e9, 9e57f, 0b647, fd1d4, ff988, e2d42, 95f4d, e5200, 7a813, ceae0, f5d8a, 1a8db, 4dd65, 15c85, 4b98a, ae7ba, e924d, 7ac7b, 3f894, 5f5b4, 31a00, 5f5b4, 95ba0, 1a5df, e269e, 1a5df, 2cc86, 03c99, 0868c, 4a714, c0b73, d972b, 436ec, d972b, 0637b, 1f724, 33902, f8184, 79045, 3a16f, f47a9, 304bf, 92504, 67275, 67ce6, bac3d9, 95134, 1ff43, bf0de, 7e78e, 1e779, 6cb16, c726b, b1bf9, 97aa5, f4751, 88b5c, c80c3, 6cb16, 4dddd, 789d9, 88b5c, bd42e, 96dd4, 3361b, 29847, 7fcd4, 9c5a3, ac2e5, 6dc46, 372a3, 7be72, 7c283, d9ff6, 454af, 372a3, 68ff5, 57bc7, 7fcd4, 36aca, 46e26, 8d210, e8c5b, afcc4, 647af, 84393, b50c4, b51e8, 09e85, d3daf, 8b9eb, 57060, 10f92, 57060, 7a2b1, 46b61, 4ab5a, 846ba, 4ab5a, 8a2d7, af3c2, 5db1a, 7fe42, 10c55, 573ab, 10c55, 12ddb, 73b8a, 2cc42, 7fc2d, 49fc4, 29946, 810b6, 2cc42, 49fc4, 8e3a4, 7fa41, a0647, 63a49, ff1b8, 93619, 63a49, cc12, 3dd82, cc12, b1b9b, e8d70, c8783, eac9b, adc1f, c8783, eae57, da811, 8fb58, 6ed74, 780a4, 8fb58, 874e3, 681a5, 09250, 10777, cbe84, 3f416, 54520, a6688, d9ff6, 7c283, 44dd1, 3b420, ae31c, 3b420, a4cdd, 5291f, 99d29, 394f9, 69f18, 486aa, 61a84, 90125, a3808, 4b98e, ccae0, 15c85, 22e48, 22e22, 395c9, 22e22, c232d, dc716, 9410e, c2856, 5042a, 0e16a, 98adb, 31fc0, 604fc, 71547, b7969, eae5e, a7f6f, 3e026, 0e16a, 98adb, f068b, 0c7b9, 7cb4d, 97951, 6e85a, 9be80, cad0c, 7cb4d, d70ae, 144e9, 2972a, 02fe1, 4e77b, 35be3, 68d52, a5dd1, 6e8c2, 8df36, a5dd1, f02c2, 68d52, 480e3, 30d1a, b560c, c543e, 8c1e3, 79146, b560c, 70700, a14f0, f4a14, bc97f, 957cf, 15ff6, d58f6, f6f39, 73c88, 63cb4, c80d0, d15fd, 9a790, a02b4, 916a4, 22d14, 59add, f0ed5, 59add, fb56c, 80a9e, 05905, 8df36, 458d6, c7f7f, af3c2, 8a2d7, 19308, 67403, 08806, 80a9e, fb56c, 677a4, 9182f, 7d25b, 6b010, f384c, 916a4, ea746, 47823, Total distance covered: 740861.1328146548

b0f3e,31838,d66bf,b3765,937c0,110be,937c0,110be,93956,e0ca5,06491,f9816,ec3b1,c3298,f9916,6af49,5ad5f,a309f,29645,38c6e,4fc27,38c6e,e0bc5,cc9f3,63b73,110be,a1727,3927f,6b650,
 d2f0a,a1727,99956,70d3a,4546d,9654a,f7d41,594af,f2783,2dbd1,d260f,60f32,9e51c,e859e,44f62,36cae,cad5f,fa4e7,6a7d5,426a0,4c6bd,e61cc,a7e7b,4dc4c,9e51c,70d3a,2dca6,fcdf4,f5b0f,
 83178,d4b97,3ee9e,bab3a,56354,53257,bab3a,5d8a7,e8333,2d20a,2b65c,34272,e8354,91c63,c5b1c,f68bb,bab67,2cf9b,409d8,4bcdb,9b7cd,19c93,4f41d,1c765,551e0,ab7,9dc5a,f7d41,
 53593,b646f,60262,7773f,d4615,a0ae4,6a3b8,01c6c,bab6a,8a7a9,f8e105,f98e7,8428b,71eab,155e,bh937,3ac3f,b5b73,99085,02057,29943,9951a,71eab,0428a,bab63,b7681,b1859,6c267,faaea,
 7283b,b71c9,a6271,33553,b85d9,70af4,af4c1,f1907,c1560,a2427,f8fe9,d0eb,f9b1,deed,f0b1,deed,f42cb,c0598,4d02c,f51c1,adbf,f6697,92798,6298b,0c956,39a4b,b6679,9bdee,d21c1,
 140bb,0c46b,9801c,8d247,fe7ae,50ed1,fe7ae,662e9,af637,f1e9f,af637,2b613,3070a,b4562,b5397,b82a2,09e9b,18474,ffef9,2d349,f78b1,8dbdf,c5f98,e701b,cdbdf,2ccf2,c15c0,cecdc,595f8,
 803cc,3d168,efc5e,f1507,e4c1c,7a2b4,42b6a,d4299,dh667,f1c2f,9e112,7cd8b,f2f67,656eb,c2a5f,58d22,c726b,a6eb1,f1a4e,f6f1e,5780b,bf16e,57aa7,1b369,dadbf,3584b,18369,
 ae1a1,29943,98ae6,e9a50,c3355,a75a5,c3355,cb884,3f416,1eb2b,8e81e,50735,fdc3c,2bf09,90125,2bf89,7dfe2,c0987,7d2d7,d24e7,3f4fb,11dd6,67e12,94049,f4d9e,e0a86,9302e,1f4d0,ae4d0,
 99d29,ca49f,67612,f3884,53f6c,67275,b8a8d,c3f8a,bab8d,c9b4,36c1d,e71b5,9d936,31512,74e15,1512a,caeb2,b47ae,cfec3,c8392,86164,79845,ae47d,33902,8637b,8db30,5f8be,elcd5,f4a0b,
 2cc8e,d102d,ac347,f5094,b73af,80b87,26d4a,b735f,bf19f,97f15,0f199,af47f,b3892,d0993,54647,15f4c,1c795,39589,1e259,fd0e8,f48d,f4d0e,4790f,558f,fd0e8,488d,55f0f,
 bf103,bad43,15e8a,d860c,292c6,36116,9e08,1276c,292c6,d860c,12325,e543c,bbb28,8e6y3,b4b6a,96939,b5b65,9b3bd,5e408,9b3d0,61f08,3f1b2,56bdc,90290,87975,8ee6f,91f08,e28ff,ac25,
 6ad97,25239,d4b6a,96939,a7be,9f9ae,af279,ae9ea,c0d48,5d6eb,f4456,99bdc,ac347,5212a,c59bd,55a4b,5c9bd,48f6b,6161d,4172,6389b,7a64e,b58ca,7a64e,41072,5fe9b,6148a,3849e,7113f,
 b61cf,e71b5,b988a,e0860,e71b5,b465f,bab38,a517b,e2d42,c645c,af557b,b598c,9443f,7f61a,3f9fd,04b49,824ba,ae04b,b559f,dab4a,faea3,fdd8b,cdd2,5b31a,dab4a,c7b1e,848dc,395d8,95d5b,
 7b78,95d5b,b5582,03f24,3db4e,c2628,527c8,e70a8,63ef2,c672e,e7bca,3ba9e,7367c,ce2d1,fdd1d,144e9,9ef5f,0b647,fdd1d,ff98b,e2d42,95f4d,e5280,7881b,ceae0,fd5ba,18f6b,4d655,15c85,
 ab98e,aa7ba,e924d,7ac7b,3f894,5f5b4,31a0f,5f5b4,95ba0,1a5df,e269e,1a5df,2cc8e,03c99,0868c,4a71a,c0b75,d972b,436ec,0972b,0637b,f172a,33902,f8184,79845,3a16f,f47a9,304bf,92504,
 67275,67c6e,b454f,9513a,1f4f4,bf0de,7686,1e779,6c61b,c726b,b1b7f,97aa5,f471f,88b5c,c80c3,c651e,ad4d,7899f,88b5c,b454f,92d4e,96d4d,3361f,29847,f747d,9c5a3,ae265,d6c4e,37a2b,7e72,
 7c283,0f67c,ad39f,35213,68ff5,f57b6,7fcd4,36a4e,6e2b6,d1210,e8c5b,afcc4,647a,8349c,b501a,e9885,d348c,8b9ef,57046,f0192,57060,72a1b,466b1,a4b5a,8a64a,c264d,af4d7,af3c2,
 5db1a,7e4e2,10d51,873b8,10c55,12d2d,73b3a,c2c42,7f2cd,af47c,29946,810bb,c2c42,af47c,8e34a,f7a61,a0467,63a4f,ff11b,93619,63a40,afcf2,3dbd2,c1f1b,1b19b,e078,c8783,ae9cb,afcd1,
 c8783,ae9cb,afcd1,87b58,6e7b4,7808a,8f5b8,87453,681a5,99250,10777,cb8a,3f416,54520,a6698,0f97f,7c283,44d1b,3b420,ae13c,3b24d,ac2d0,529f1,99d29,3f94f,69f18,486aa,61a9b,9a125,
 a38db,0b98e,cac4a,15c85,22e48,22e22,395c9,22e22,c232d,c271b,9410e,c285e,5042e,ae11a,98ab3,31f60,6804c,7514f,b796f,ae59e,at76f,3e02e,8e1a5,7808b,f068b,0c7b9,7c7d4,97951,6e85a,
 9be80,cad0c,7c04d,c078a,144e9,2972a,02f9e,1ae77b,35b63,60d5e,asdd1,6e8c2,8df3e,asdd1,f0c2c,60d52,6804c,3051a,b560c,ae43e,8e1e3,79146,b560b,78080,014f0,fa1a,b,c9b7f,95f5f,15ff6,
 d58f6,fd639,73c88,63cb4,c8bd0,d197a,9a79a,ae02a,916a4,22d1a,59add,f9e05,94d8f,5b6c8,0849a,05985,8df3e,458d6,c77ff,af3c2,842d7,1938b,67403,0808b,80a9e,fb56e,677fa,9182f,7d25b,
 db018,f384c,916a4,ea74e,47823,
 Total distance covered: 633321.840379663

[illegible]

9d29b, d1b0e, 801c9, 8d027, 66209, 4e7a6, fe7ae, fe7ee, fdeed, 619e2, af637, af637, 2b603, 3070a, b0542, 6b397, db82a, 09e90, 18474, ff9fe, f9fe9, 2d549, cdfcf, 2ccr2, 421bf, c15c0, c15e0, cec4c, 595f8, 803cc, 3d165, 31587, 53555, ccd01, 3a074, fe2ba, d4299, dbd67, 9e912, 1fc21, a661b, fl4ae, f6625, 198db, d674f, af4c4, c878c, ed783, ed878, b1b9c, cdfc2, ccf12, 61639, 9361f, b1f5b, 7880a, 4e7d4, 9eae7, da013, 87e43, 8fb58, 8fb58, c3355, c3355, a75a5, cb8e4, cb8e4, 3f416, 3f416, 54520, a6688, 7c283, 44dd1, f4d40, ff1ff, cfb0c, ba8bd, ba8bd, 7e78e, 1e779, c80c3, 6cb16, 4d4dd, 789d7, 4a751, 97a5a, b1bf9, c726b, c726b, 58d22, 62a5f, 6ec0b, f2f07, f2f07, 7b415, 1512a, 1512a, 36c1d, c9b4d, f7b55, 0953e, caeb2, ba47e, cfc3c, ccc3c, 8e164, 3a41f, f4749, 304bf, a474, 0886c, 03f99, 2cc8c, 2c8c0, 4afb0, 1a5df, 1a5df, 95ba0, 31a80, 98d87, 26d6a, cb3f7, f5094, 70357, 70357, 9715f, 0f199, 0f199, c3b43, 99bdc, ec311, ab474, 15f4c, 1c795, 50999, 1d25f, f8e64, fd0de, f8d0e, c489f, c489f, c489f, c489f, 55fdd, 55fdd, b1f03, b1f03, b1f03, 19e84, d0c6d, d0c6d, 36116, 292c6, 292c6, 00e1b, 270ec, 12325, e53f8, bbb28, e280f, abc25, c9d97, 3fbf2, 56bdc, 90290, 87975, 8ee5f, babb5, 96939, 96939, b4b46, b4b46, 8ae93, 23239, 61f08, 61f08, 9b3d0, 9b3d0, 5e408, 4e7be, 9fe9a, f2729, b8962, d0993, 4e9ea, 54eb6, 54eb6, f4456, 4521a, c9b9d, 5da55, 48feb, 6616d, 41072, 41072, 6389b, 76d4e, 76d4e, 650ca, 5fe9b, 814da, 3849e, 7713f, b1c1f, c7b5e, e71b5, e6900, 19884, 64ab6, b3a8f, 5a517, a517b, 95f4d, c4459, 0589c, a643f, f7f6a, 3f9fd, 82d4a, b6449, ead40, b059f, faa8c, cdda2, ddf68, b5318, da01a, dabba, c7b1e, 8dfdc, 8d339, 95d5b, 95d5b, 781b7, bb562, 83f24, 34d6b, d2258, 5272ca, e708b, 63f62, 6724c, a7bac, b3a8f, 7367c, cbe21, fd1d4, b4d7f, 14ae9, 0b647, ff9e8, a2d42, d2d42, e520b, 2803b, f818a, c9a0b, 5f80d, 8ddc, 4dd65, 15e8b, 15e8b, ccae0, 9244a, e924d, 7ac7b, 3f894, 5f504, 5f504, a269e, d102d, c4f86, c0b73, 436ec, d972b, d972b, 0637b, 0637b, 5f9eb, e1cd5, d0b3b, a47d7, 1f724, 33902, 33902, f8184, 79045, 79045, 3f250, 67275, 67275, 67275, 67275, 95134, e0a8e, 9302e, f4d9e, c4a99, c4a99, 67612, 67612, 99e29, 99d29, 48ba0, c9a79, c232d, 22a48, 22e22, 22e22, d0c16, 7afde, 2bf19, 2bf19, 08c1e, 1b82b, e9a50, 98ae6, 10777, 925d0, 68155, adcf1, eac9e, 36ac4, eac9e, d0210, e8c5b, b4c93, b50c4, 1b369, 0951a, 9ae6a, b3b6b, b3b6b, ba37a, 0428d, 2428d, f98e7, 18e05, 7a23b, 3170f, c6267, 6c267, 1b5d9, b8597, 7040f, a4cf1, 57aa7, bff1c, b50c4, b510e, 09e85, d33d7, 3d8d2, 6349e, a0e47, 7fa41, b9b9b, 8a67d, 8a2d7, af3c2, 73b8a, 12db5, c2c22, 2cc4c, 49d4f, 7fc2d, 49fcd, 8e34a, 8101b, 29946, 50735, df35c, 394fd, 11e6b, 3f884, 5f5f5, c9f42, 7e2d7, 7e2d7, 4d96f, ab38e, a38d8, a517b, 7b72, 372a3, 372a3, 8a2ff, 7b72, 29847, 96d04, 88b5c, 88b5c, bd42e, 3361f, 7f42d, 7f42d, c4f6c, ac2e5, 9c5a3, 45a4f, dfffc, dfffc, 7c283, 3b420, 3b420, a2c4d, a4cdd, 5291f, 69f1b, 6184, 9d125, 99125, 9410e, c285e, 8e164, bae5e, a7f6f, 3c02e, 0c7b9, f68b3, 98adb, 319f, 98ad, 58ad, 58ad, 395c9, 9795f, 1a65a, 9e80, cda0, 7cb4d, 7cb4d, d70ae, 14a29, 2972a, 02f61, 4e77b, 35813, 60d52, 60d52, 4080b, 9e16a, b560e, b560e, 9161c, c543e, 9161a, ea746, 47823, 47823, a02b4, 9a7b9, c8040, 957cf, 51f69, 9526d, 63cb4, 73c8e, 76f39, d58f6, 15ff6, bc97f, 4a41a, a14f0, 70709, 604fc, 71547, 07969, 573ab, 10c55, 10c55, 7fe42, 5db1a, c3c2f, c7c7f, 45846, 8df3c, 8df3c, 08080, 80a9e, 80a9e, 05905, 6e8c2, a5dd1, a5dd1, f0c2f, 7d25b, 60b10, 9182f, 6774a, f056c,

Total distance traveled: 704176 (3048488796

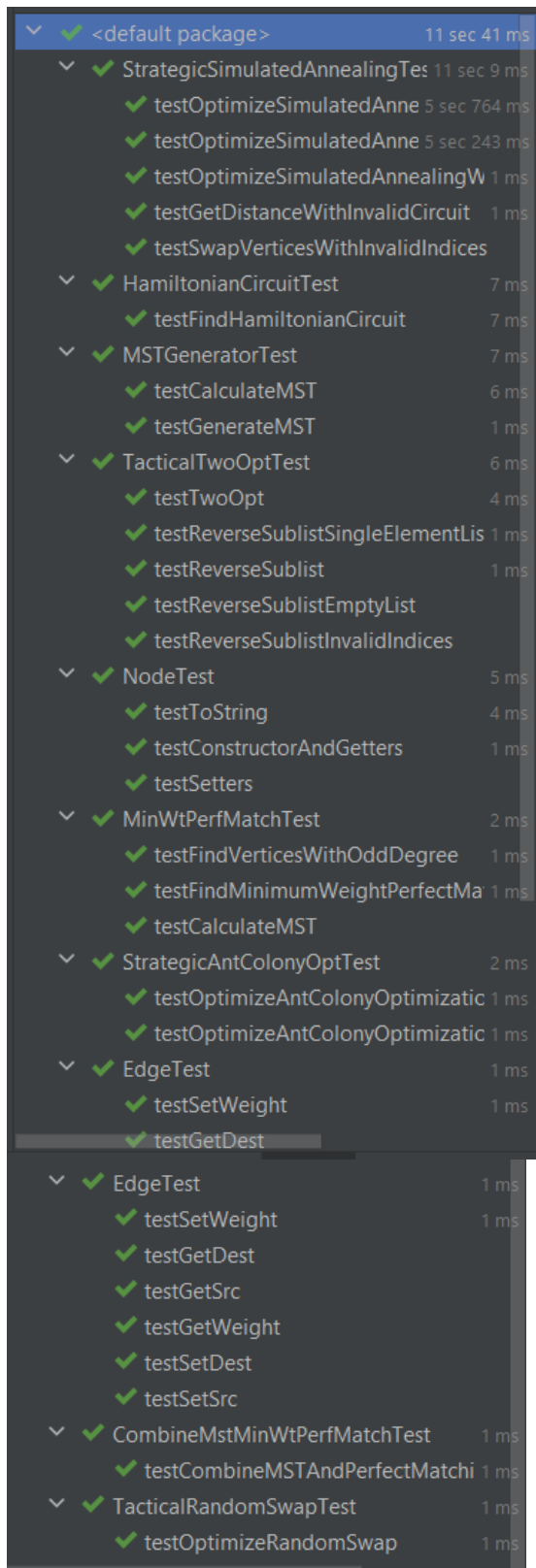
60d52, f02c2, a5dd1, a6ebc, 8df36, 05999, b0793, b0d5, 02b57, 3ce3f, bb937, 15ede, 71ab6, 0951a, ba063, b7b61, b5d59, 53135, ac267, b71c9, 728d3, faaea, 8e105, f98e7, 0428d, 0a7a9, 70d0f, at4ce, fa751, 88b5c, 789d9, adddc, 6cb16, c8063, 1e779, bd42e, 96dd4, 3361b, 29818, 7fcd4, 57bce, 68ff5, 37e23, 454af, df66e, a6088, 54520, 71b27, 7c283, 444d1, 3b420, ae31c, a54cd, 5291f, f4fd0, 6dc4c, ace25, 9c5a3, 8c1e, 50759, f4d3c, 2b097, 7dfe2, ce987, 90125, 61a8c, 48484, 69847, 99d2c, 2a9e5, 711d2, 116db, 3f41c, a2607, 7d2d7, d098b, aae74, 3b244, 7ac7b, 3f894, 5f5dc, 31a08, e269e, 1a56f, 95ba8, 5f5cc, 3f884, 95134, 3ab49, 67c6e, 67275, ba8bd, c9b4d, 36c1d, e71b5, d9536, 1512a, ceaab, cfec3, ce2c3, 861e6, 79045, f8184, 33992, 1724, 0637b, d972b, 436ec, cbb73, 4a714, 0868c, 03c99, 2cc8c, 64fab, d102d, c4f8c, b6c37, f5094, 76357, 26d4a, 09a87, 6616d, 41072, 7ab4c, 638b9, b510c, 5fe9b, 81d4a, 3849e, 7713f, b16c4, f71b5, b6808, 19884, b465f, ba3e8, a517b, c4459, e2d42, ff988, 95f4c, de520, 0b647, fd1de, cbe21, 7367c, 3ba9e, a7bac, 6742c, 63f6e, e70a8, 527ca, e228c, 3db46, 83f24, bb562, 95d5b, 8d330, 8f6cc, c7b1e, 78be7, da0ba, 5b318, faea3, fdf68, c0da2, b559f, ead04, 3ff9d, bd449, 7f6fa, 824ba, 04f3f, 0598c, 7a81c, ca6e, f5d8a, 1a8db, 4ad65, 15c85, ccae8, a3b8c, c232d, 222e, 2248, 395c7, c71d6, 9410c, c8556, 5042a, 1e6a, 98ab, 31fbc, 0626b, 067b7, 7cbdd, 47f0a, 144e9, 9e57f, 2927a, 6844f, 71547, 7b969, ea55c, a7f6f, 3e02d, c4ac0, 9b8e0, 8658a, 97951, 29946, 810b6, 2cc4c, 49fc4, 7f24c, 73bba, 12db0, 1c285, 547d2, 5db1a, 35bea3, a7f6c, f82f1, 573ab, 8c3a4, 7fa4f, 14f0a, e0a9b, 9e036, 10433, bf0de, 7e78e, 92504, 3c04f, f47a9, 3a16f, c3f6f, 74a15, 6b397, b82a2, 4211b, ba3ae, 53257, 5da87, a833d, 2d0ae, d8bd1, 9e51c, adcaf, a7b0f, 6e11c, 4bcd6, 4244c, ccf93, eb0ca5, 38c6c, 4fc27, 29e45, 049d1, a30f9, f9816, 6afdc, 93c8b, ec3b1, 5aad7, b37e5, d56f, 937c0, 110bc, 6b7b3, a1727, 93934, 70334, f546d, 9654a, a7c1a, f7d41, 594af, c7d03, 7dd1d, db60f, 68f32, 85357, b46af, 1c7e4, 4dff6, 5510e, 3f27f, 6b450, d2f84, 018c, 93956, 5138c, 50f63, 4ab7f, b5d5f, f0b49, f8f8b, d3788, 528a2, 0b8cd, 055e9, fcf1f, cdfb9, d15db, 0880d, 35cd3, 4d843, 5f169, 9256d, 4ab83, 06f63, 3a633, f0621, d5826, 6f639, 15f66, 957cf, bc7f4, 47a14, a14f0, 70780, 7388c, 63c04, 3a8ef, c99bd, 50a45, 4521a, c3b11, 99bdc, f4456, 546e, cdb83, 4ee9a, f08de, 55f6d, f1083, bad43, 158e4, d0dc8, 292c2, 270e8, 0e10, 36116, 5099b, 1d259, d9f8c, 9e67a, f489f, 1c795, 15f40, 4ab47, d0993, 36f87, 01f99, 97f15, a7279, 9f69a, e7a7e, 96939, b6b85, 9b3d8, 5e408, 86f09, b1308, 3ff02, 9029b, 87975, 447d7, d0b30, 5f6eb, ec1e3, 68646, 8a693, bbb28, e53f8, 12325, e280f, abc25, 0ad97, 23239, 56bcd, ba74e, ba542, 30704, 2b683, af63a, 61e92, 662e9, fe7ae, 8b2d7, 801c9, d1ba6, 140b0, 3bdee, 76697, 3a496, 0c956, 43aa8, 02928, 44fb2, e85ee, 36cae, cd05f, fa4e7, 6a7d5, 50ed1,

Total distance: 7116184.8324200534

The complete output is present in the Output.txt file in the project files.

UNIT TESTS

Screenshots:



✓ <default package>	11 sec 41 ms
✓ StrategicSimulatedAnnealingTest	11 sec 9 ms
✓ testOptimizeSimulatedAnnealing	5 sec 764 ms
✓ testOptimizeSimulatedAnnealing	5 sec 243 ms
✓ testOptimizeSimulatedAnnealingWithInvalidCircuit	1 ms
✓ testGetDistanceWithInvalidCircuit	1 ms
✓ testSwapVerticesWithInvalidIndices	
✓ HamiltonianCircuitTest	7 ms
✓ testFindHamiltonianCircuit	7 ms
✓ MSTGeneratorTest	7 ms
✓ testCalculateMST	6 ms
✓ testGenerateMST	1 ms
✓ TacticalTwoOptTest	6 ms
✓ testTwoOpt	4 ms
✓ testReverseSublistSingleElementList	1 ms
✓ testReverseSublist	1 ms
✓ testReverseSublistEmptyList	
✓ testReverseSublistInvalidIndices	
✓ NodeTest	5 ms
✓ testToString	4 ms
✓ testConstructorAndGetters	1 ms
✓ testSetters	
✓ MinWtPerfMatchTest	2 ms
✓ testFindVerticesWithOddDegree	1 ms
✓ testFindMinimumWeightPerfectMatch	1 ms
✓ testCalculateMST	
✓ StrategicAntColonyOptTest	2 ms
✓ testOptimizeAntColonyOptimization	1 ms
✓ testOptimizeAntColonyOptimization	1 ms
✓ EdgeTest	1 ms
✓ testSetWeight	1 ms
✓ testGetDest	
✓ EdgeTest	1 ms
✓ testSetWeight	1 ms
✓ testGetDest	
✓ testGetSrc	
✓ testGetWeight	
✓ testSetDest	
✓ testSetSrc	
✓ CombineMstMinWtPerfMatchTest	1 ms
✓ testCombineMSTAndPerfectMatch	1 ms
✓ TacticalRandomSwapTest	1 ms
✓ testOptimizeRandomSwap	1 ms

CONCLUSION

Based on the given table, we can see that the MST algorithm has the lowest distance of 513326.09 meters, which means that it provides the optimal minimum spanning tree for the given problem. However, the Christofides algorithm, which is a heuristic algorithm for the TSP, has the lowest ratio of 1.443256332, indicating that it provides a solution very close to the optimal distance obtained from the minimum spanning tree.

The 2-opt algorithm has the third-best ratio of 1.233761253, which is better than the ratios obtained by the Random Swap, Simulated Annealing, and Ant colony algorithms. This indicates that the 2-opt algorithm is more effective in optimizing the given TSP problem than these other algorithms.

Out of the four TSP optimization algorithms, the 2 Opt algorithm has the best distance ratio of 1.233761253, which means it produces a tour that is only 23.4% longer than the optimal distance, on average. The other three TSP optimization algorithms, namely Random Swap, Simulated Annealing, and Ant colony, have a distance ratio ranging from 1.372167212 to 1.398117053, which means they produce tours that are 37.2% to 39.8% longer than the optimal distance, on average.

Therefore, based on the given table, the 2 Opt algorithm is the best optimization algorithm apart from TSP and MST for solving the TSP problem as it provides the most efficient and effective solution among the TSP optimization algorithms

REFERENCES

- Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.
- Arora, S. (1998). Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5), 753-782.
- Carpin, S., Černý, V., Pechoucek, M., & Kleiner, A. (2013). A distributed ant colony algorithm for solving the traveling salesman problem. *Swarm Intelligence*, 7(3), 267-295.
- Cook, W. J. (2012). *In pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press.
- Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1), 53-66.

- Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.
- Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(2-3), 293-306.
- Guan, J., & Nakamori, Y. (2009). Simulated annealing algorithms for the traveling salesman problem with pickup and delivery requests. *European Journal of Operational Research*, 198(2), 395-405.
- Helsgaun, K. (2000). An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1), 106-130.
- Johnson, D. S., & McGeoch, L. A. (1997). The Traveling Salesman Problem: A Case Study in Local Optimization. *Local Search in Combinatorial Optimization*, 215-310.