***Distance matrix of the problem***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | City01 | City02 | City03 | City04 | City05 | City06 | City07 | City08 |
| City01 | 0 | 46 | 35 | 80 | 89 | 56 | 84 | 57 |
| City02 | 46 | 0 | 66 | 64 | 54 | 87 | 76 | 85 |
| City03 | 35 | 66 | 0 | 75 | 20 | 43 | 21 | 76 |
| City04 | 80 | 64 | 75 | 0 | 33 | 16 | 68 | 81 |
| City05 | 89 | 54 | 20 | 33 | 0 | 16 | 52 | 10 |
| City06 | 56 | 87 | 43 | 16 | 16 | 0 | 68 | 22 |
| City07 | 84 | 76 | 21 | 68 | 52 | 68 | 0 | 57 |
| City08 | 57 | 85 | 76 | 81 | 10 | 22 | 57 | 0 |

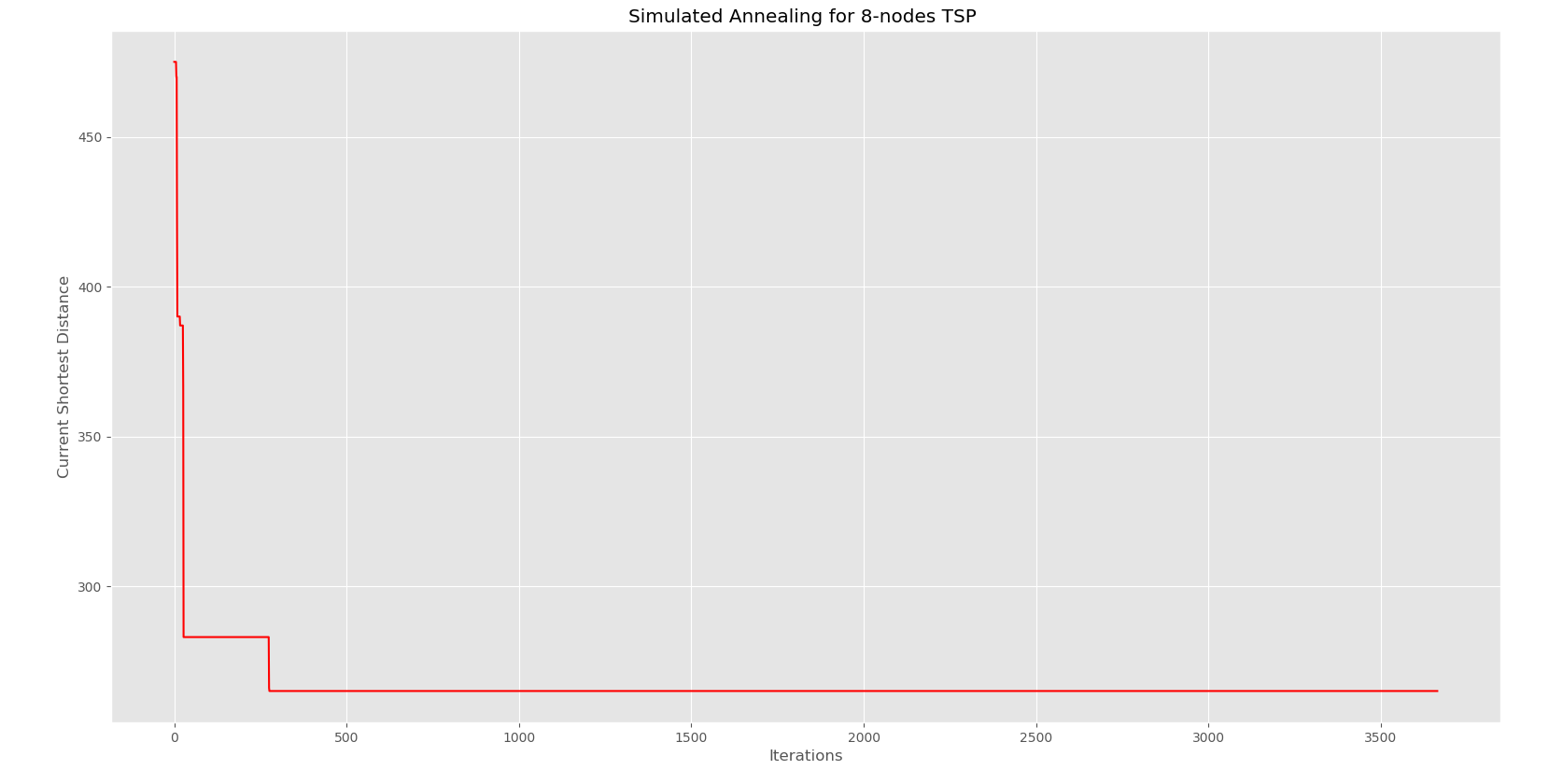
***Simulated Annealing for the problem***

|  |  |
| --- | --- |
| Parameter Setting & Mechanism Selection | |
| Initial Temperature | 500 |
| Final Temperature | 0.1 |
| Cool Schedule | Lundy and Mees (1986)  T(k) = T(k-1) / (1+γT(k-1) ，γ= 0.001 |
| Number of Iterations | Linearly Increasing  niter = α(t0-t)+β; α = 2, β = 15 |
| Permutation Neighborhood | one-third opportunity to do Inversion / Transposition / Displacement respectively |
| Reheating | When a move accepted:  t = t / (1+β)；β = 0.2 |
| When a move rejected:  t = t / (1-γ) ; γ = 0.1 |

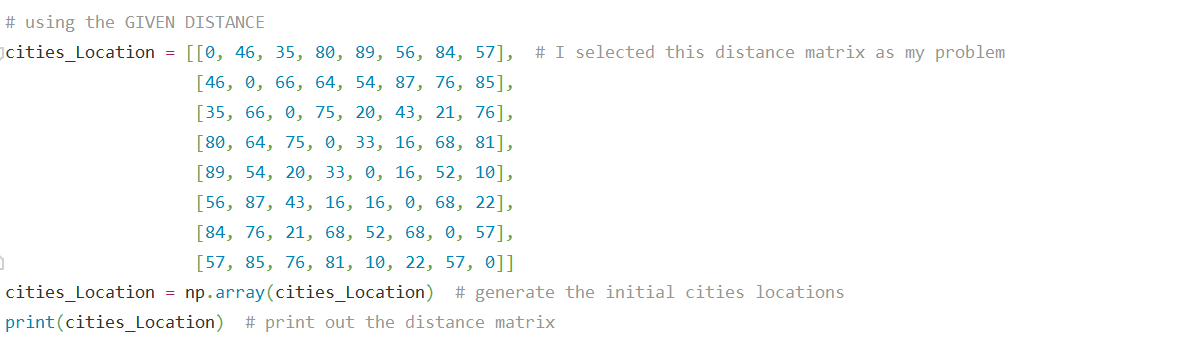
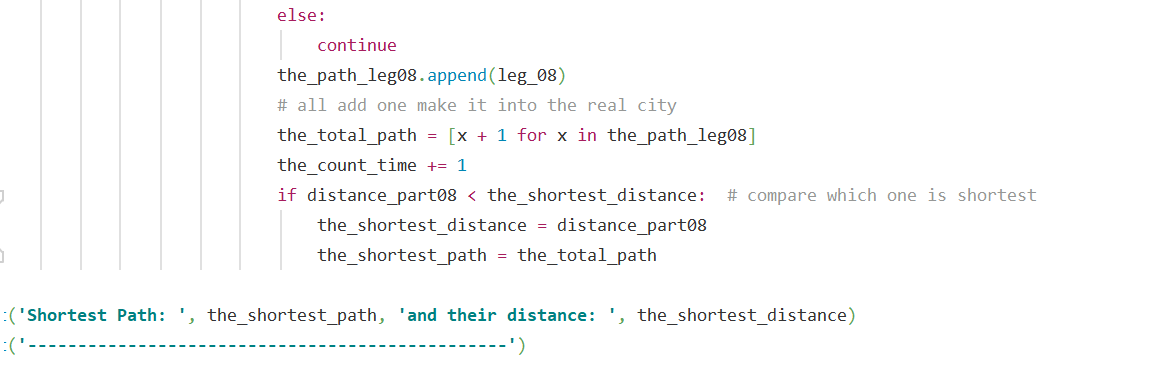
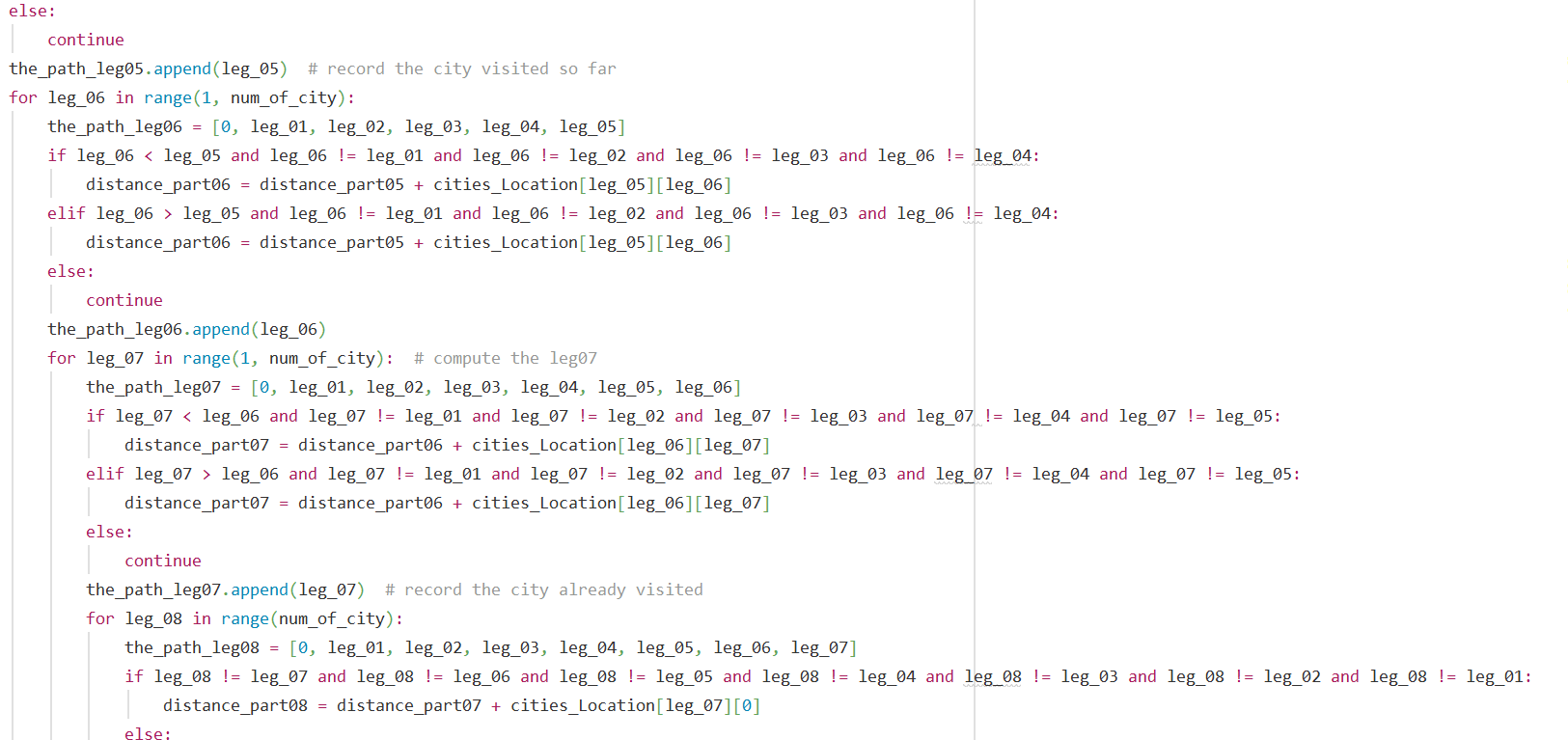
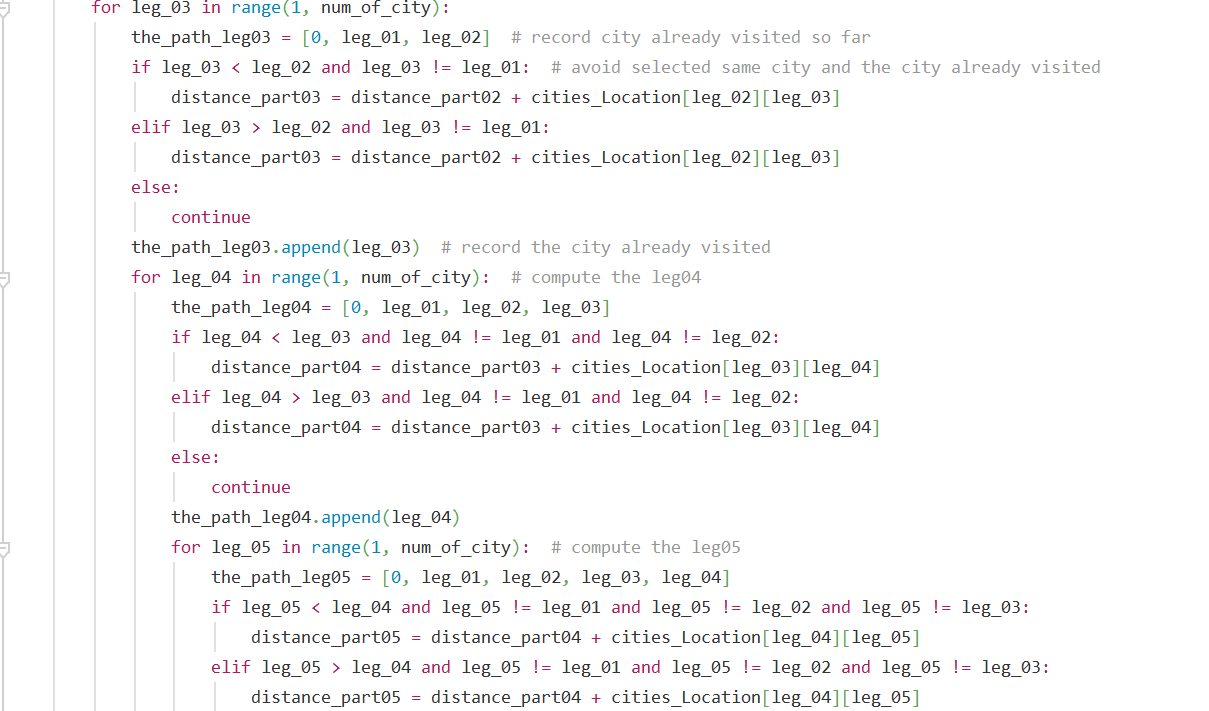
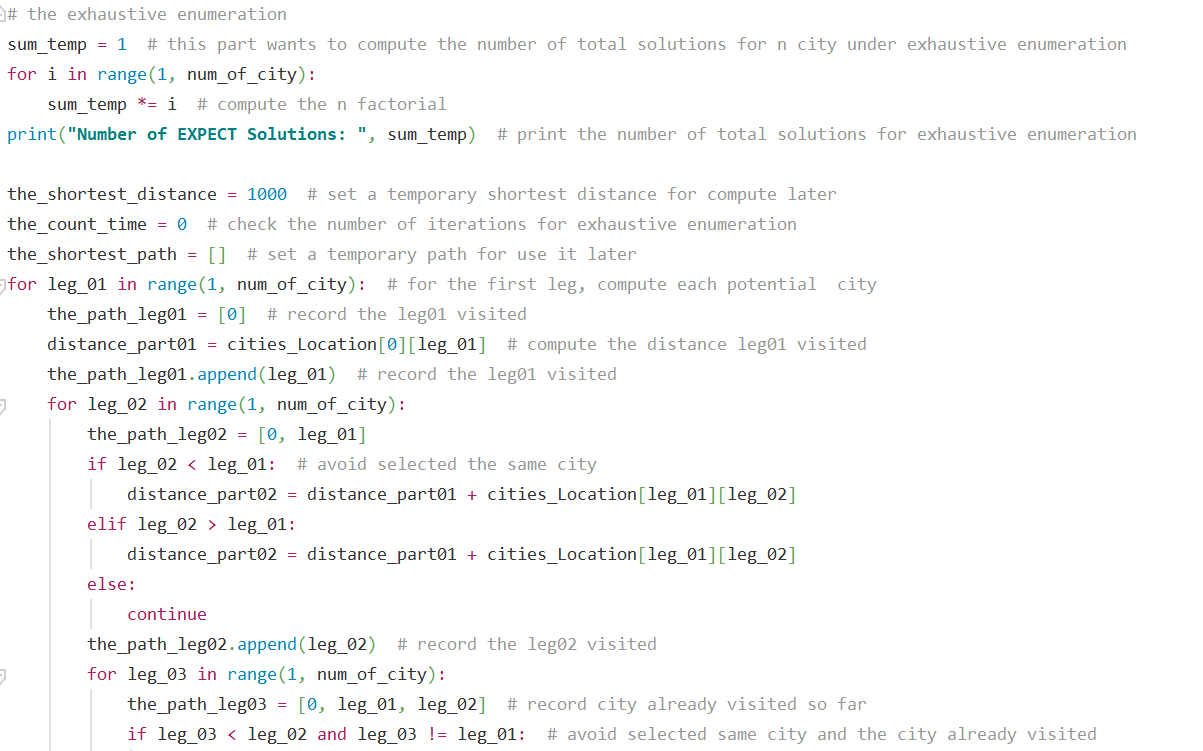
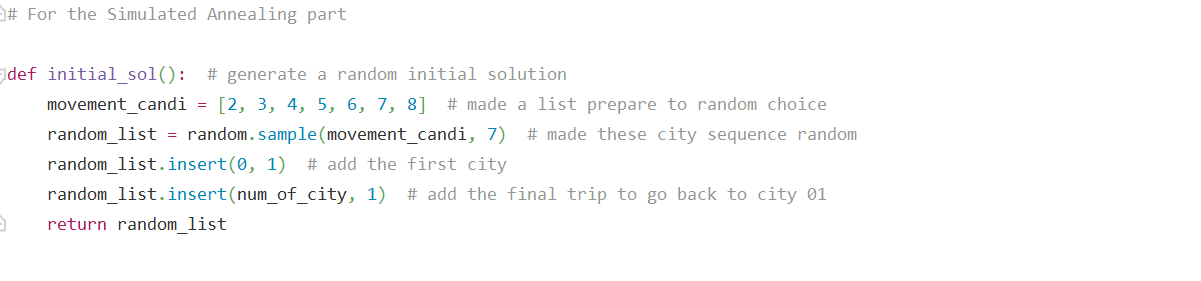
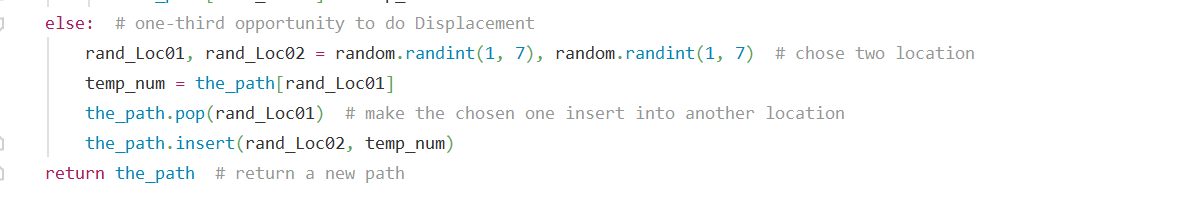
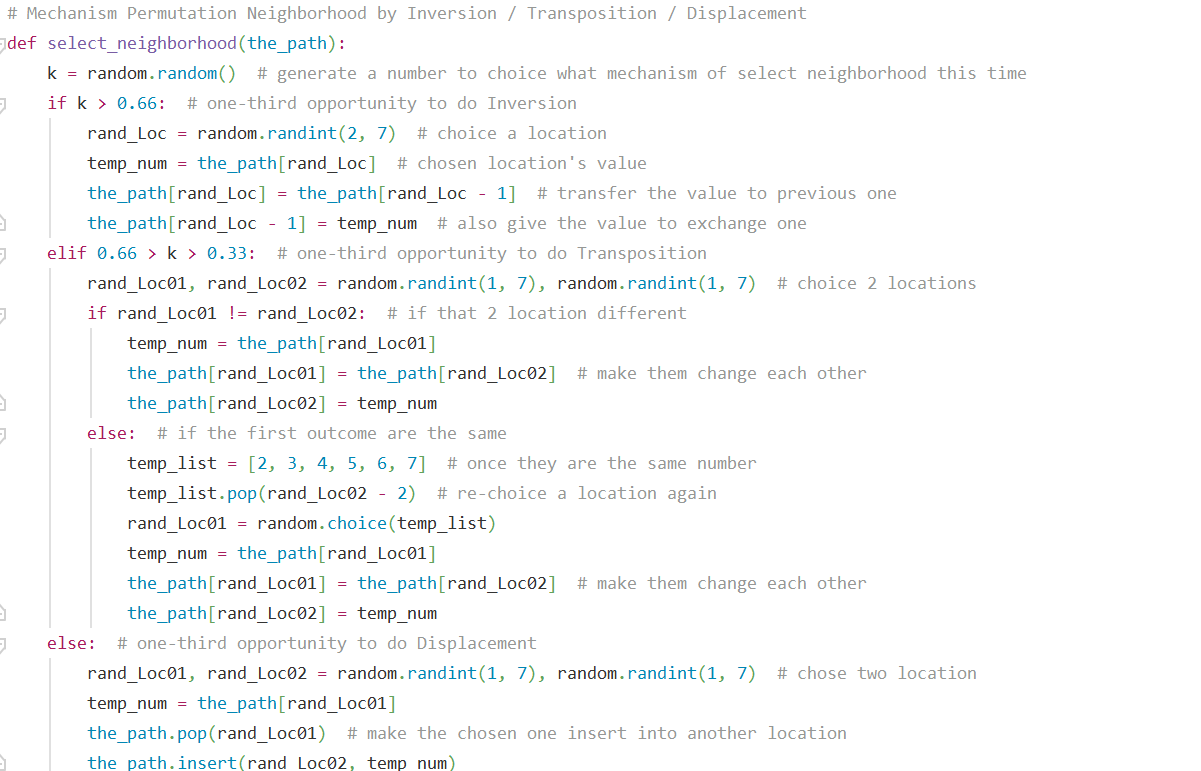
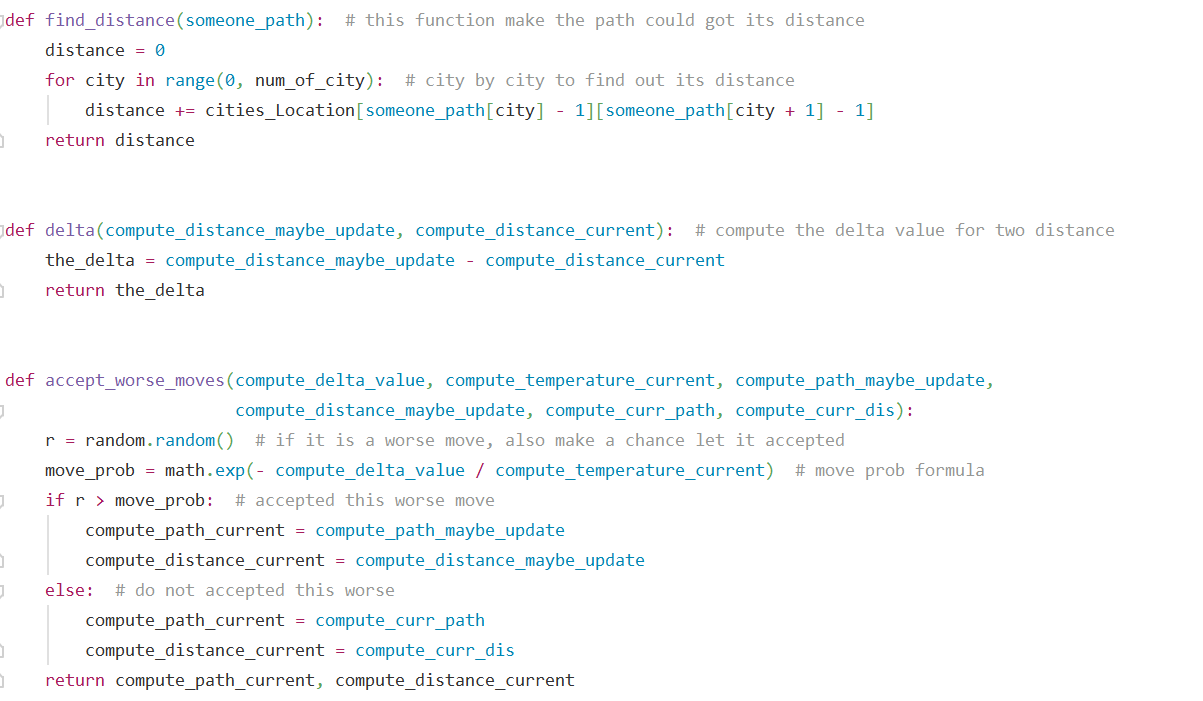
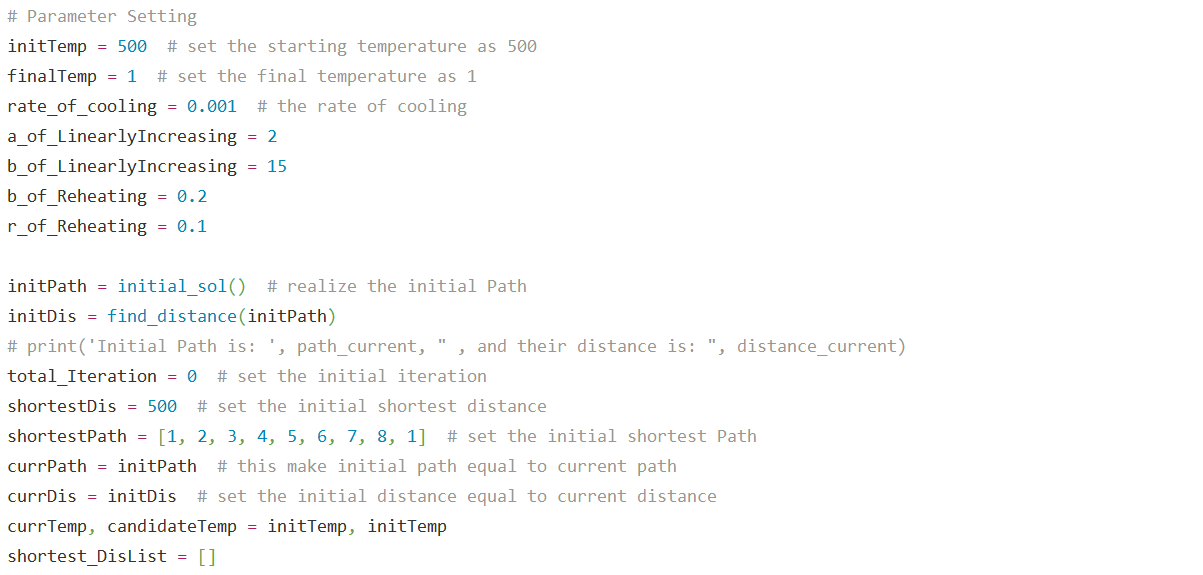
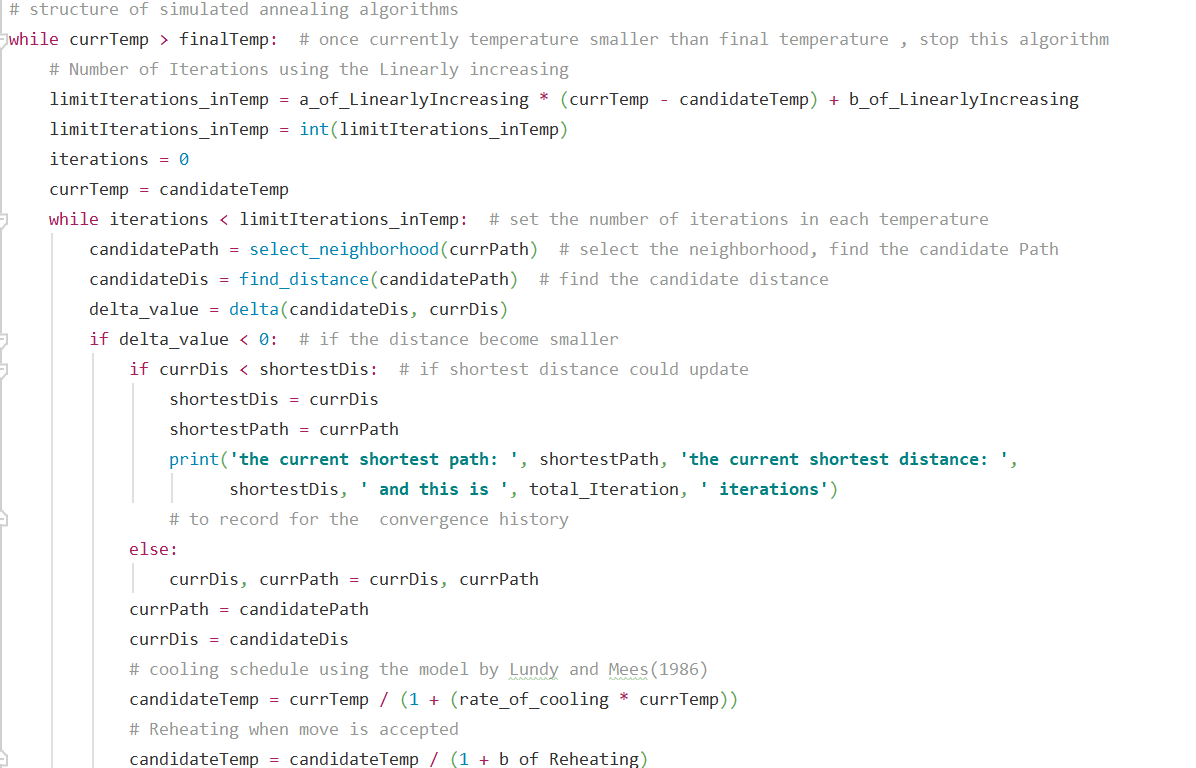
***Optimization result***

|  |  |
| --- | --- |
| Solution Under Exhaustive Enumeration | |
| Shortest Path: | [1, 2, 4, 6, 5, 8, 7, 3, 1] |
| Shortest Distance: | 265 |
| Number of Total Solutions | 5040 |
| Solution Under Simulated Annealing | |
| Find the shortest path:  [1, 2, 4, 6, 5, 8, 7, 3, 1], with shortest distance 265  On the 276th iterations | |
| Number of Total Iterations: | 3665 |

***The evolution history***



***Program Code:***

1. ***Generate the distance matrix for problem***
2. ***Determine a distance matrix for my problem.***
3. ***Write the exhaustive enumeration***To find the optimal shortest path and their distance solution for my problem.  
   
4. ***Start the Simulated Annealing Algorithm***First, establish a function to generate an initial path randomly.
5. ***Permutation Neighborhood by Inversion / Transposition/ Displacement.***
6. ***Established three functions.***find distance: make a path could show their distance in my problem.  
   delta: compute the distance current one and previous one.  
   accept worse moves: once occurred a worse move, compute the move probability and generate a random number.
7. ***Parameter Setting, Generate initial path and its distance.***
8. ***Start the structure of Simulated Annealing***
9. ***Visualization***