

Ideas Behind the Algorithm

First, we wrote a function that stores points in memory. The aim here was to provide accurate and direct access to the points. We did not use Array because the number of input points is not known, we decided that we would design it more easily with memory allocation. After doing these, we created an array function containing two dimensions. We made this array with a pointer to call it correctly in memory.

After these auxiliary functions, we used the greedy approach TSP algorithm, which is the main algorithm of our project. The algorithm first accepts the 0 and 1st indexes of the array as the starting point. Then he creates the half TSP tour with the greedy approach approach. We equated the outputs from 0th and 1th indexes to two pointers named testtour1 and testtour2. We created a for-loop within the greedyTSP function to calculate the lap distance will be shorter from whichever starting point we start.

Since the loop is not controlled from the 2nd index point, we exited the loop by starting the for-loop from the 2th index and advancing it to the last point but if we take every point as the starting point and check it, we added the "two-operation TSP" algorithm because the algorithm run time will be very high. The aim here is to ensure that new tours are found, starting from points that were not in the previous tour, rather than checking each point. In this way, we save time. If the point is in the tour, it doesn't need to be the starting point because it can be moved with the two_opt algorithm. After the implementation of these functions, we got 2 different tours. We got a new test tour by resetting whichever distance is longer in each iteration. After these, the greedy_tsp function returns the tour with the shortest distance and we get our output. After obtaining the shortest tour, we used the two operation algorithm so that both the distance was optimized. In addition, the points that we did not control as a starting point were checked.

Finally, we made the main method first priority text file reading operations. Then we call the necessary functions. After the functions were completed, we printed the distance and elements of the shortest path we obtained as output.

Important Functions Used and Their Explanations

```
point *createStruct(FILE *fp) {  
  
    //deallocating input pointer  
    void freeStruct(point *input) {
```

The createStruct function keeps all points in memory and syncs them to the pointer. The freeStruct function deallocates the memory used in createStruct.

```
//creating adjacency matrix for all points  
void createAdjMatrix(int rows, int cols) {  
  
    //deallocating adjMatrix to avoid memory error  
    void freeAdjMatrix(int rows) {
```

createAdjMatrix creates an adjacency matrix of all points. freeAdjMatrix deallocates the used memory.

```
//greedy tsp algorithm  
int **greedyTSP(int **adjMatrix, int n) {  
  
    //deallocating tour  
    void freeGreedyTSP(int *tour) {
```

greedy TSP algorithm modified according to the logic we explained on the first page. freeGreedyTSP deallocates the used memory at greedyTSP. The purpose here is to prevent confusion due to different values in memory.

```
//calculating tour distance  
int tourDistance(int *tour, int n) {
```

In this function, it calculates the total distance of the tour sent to the function and returns this value.

```
//two-opt algorithm for decrease the tour distance  
int* two_opt(int* tour, int **adjMatrix, int n_cities) {  
    //deallocating memory  
    void freeTwoOpt(int* tour) {  
        free(tour);  
    }
```

We used the two_opt algorithm to better optimize the tour we obtained. We used the freeTwoOpt function to deallocate the memories we used in the two_opt function.

In the main method, we have completed functions such as reading a file, obtaining output, writing the output file as txt, calling a function.

```
FILE* fp = fopen(fileName, "r");  
//writing file name  
FILE* fp1=fopen("test-output-3.txt", "w");  
  
//if file not found  
if (fp == NULL) {  
    printf("Error opening file.\n");  
    return 1;  
}  
  
//gen pointing the starting point of point struct array  
gen = createStruct(fp);  
fclose(fp);  
//creating adjmatrix  
createAdjMatrix(inputSize, inputSize);  
//half size tsp  
int halfSize = ceil(inputSize / 2);  
int* tour = greedyTSP(adjMatrix, halfSize);  
  
//optimizing the tour with two-opt algorithm  
int* optimize = two_opt(tour, adjMatrix, halfSize);  
  
//printing tour distance  
printf("%d\n", tourDistance(optimize, halfSize));  
fprintf(fp1, "%d\n", tourDistance(optimize, halfSize));  
  
//printing tour  
for (i = 0; i < halfSize; i++) {  
    printf("%d\n", optimize[i]);  
    fprintf(fp1, "%d\n", optimize[i]);  
}  
fprintf(fp1, "\n");  
//deallocating memory to avoid to memory error  
freeStruct(gen);  
freeAdjMatrix(inputSize);  
freeGreedyTSP(tour);  
freeTwoOpt(optimize);  
return 0;
```

INPUTS AND OUTPUTS OF THE PROJECT

Input-1:

Consists of 280 cities -> distance: 1357 // starting point: 184

Execution time:

```
Process exited after 1.754 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\doguk\Desktop\algo son>python half_tsp_verifier.py test-input-1.txt test-output-1.txt
Your solution is VERIFIED.
```

Input-2:

Consists of 1002 cities -> distance: 126073 // starting point: 518

Execution time:

```
Process exited after 31.86 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\doguk\Desktop\algo son>python half_tsp_verifier.py test-input-2.txt test-output-2.txt
Your solution is VERIFIED.
```

Input-3:

Consists of 33810 cities -> distance: - // starting point: -

Execution time: We tried to run input-3 but it took more than 12 hours. However, we could not get output. After selecting only one point as the starting point, we sent the tour we obtained to the two_opt algorithm, but although the process took more than 9 hours, we could not get any output.

Input-4:






Consists of 2924 cities -> distance: 5045 // starting point: 984

Execution time:

```
Process exited after 934 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\doguk\Desktop\algo son>python half_tsp_verifier.py test-input-4.txt test-output-4.txt
Your solution is VERIFIED.
```

RESULT SUMMARY TABLE OF THE
PROJECT

/ 	Input -1 	Input -2 	Input -3 	Input -4 
Cities	280	1002	33810	2924
Distance	1357	126073	//	5045
Execution time	1.754 sec	31.86 sec	//	15.56 min