Group project: Benchmarking Heuristic and Nondeterministic Approaches in Traveling Sales-

man Problem (TSP) and Its Variants

**Dataset URL**

Tiny / Small (<10) / medium (10-50) / large (>50)

Kaggle dataset:

<https://www.kaggle.com/datasets/mexwell/traveling-salesman-problem>

TSPLIB:

<http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>

TSP Challenge:

<https://github.com/acu192/fun-tsp-challenge/tree/master>

Baseline algorithm

**Exact search (output time)**

# (1) Brute-force search (c++) wu

<https://github.com/AbrarJahin/travelling-salesman-problem-brute-force>

(2) Dynamic Programming (Held-Karp algorithms) (c++) Zhang

<https://github.com/ishanjogalekar/TSP-using-Dynamic-Programming-?tab=readme-ov-file>

(3) Integer linear programming (python) huang

<https://github.com/Arthod/LP-tsp-gurobi>

**TSP heuristics algorithm (output time and path)**

(1) Nearest neighbor (python) wu

<https://en.wikipedia.org/wiki/Nearest_neighbour_algorithm>

<https://github.com/chingisooinar/KNN-python-implementation>

<https://gist.github.com/mkocabas/feb9c4431e552aa33ffe2dc95d58c76c> (matlab)

(2) Lin–Kernighan heuristics (python) zhang

<https://en.wikipedia.org/wiki/Lin%E2%80%93Kernighan_heuristic>

<https://github.com/kikocastroneto/lk_heuristic>

(3) Christofides algorithm (python) huang

<https://github.com/Retsediv/ChristofidesAlgorithm>

对比方法：

1. 总路程（检查阶段）
2. 同一个算法，不同dataset的时间（测试复杂度O（n））
3. 相同的dataset，不同的算法的时间

2025.3.22

10 city dataset (The initial point is the first data coordinate of the data set)

<https://www.kaggle.com/datasets/mexwell/traveling-salesman-problem/data>

Compute the time of the three accuracy algorithm

Target: input: dataset, output: algorithm time

Next meeting time: after class on Tuesday

Comparative studies

Latex:<https://www.overleaf.com/9344939596wwpdvptbfcqh#022ad7>

Fanli Wu

# Brute-force search : this algorithm considers all possible routes and computes their distance to find the minimum distance.Therefore, this algorithm will always give the optimal solution.

A→B→C→D→A

Nearest neighbor : we start at the first point and consider the nearest unvisited point until all points are visited.

