r/ThingsYouDidntKnow • 2 days ago
TheStocksGuy

# TSP P equals NP 1000 sorted in an array as proof

## Understanding the Sorting and Adjustment Process in TSP

In this example, we start with an array of 1000 cities, indexed from 0 to 999. After sorting, we adjust the array to form a closed loop by adding the first city at the end, resulting in a sorted array of length 1001.

## Timestamps

1. **Sorting Time**: This is the time taken to sort the array of cities. In this example, it took 0 seconds and 045 milliseconds. **Sorting Time: 0s 045ms**
2. **After Sorted Adjusted Last to First**: This is the time taken to add the first city to the end of the sorted array, ensuring the path forms a loop. This step also took **0 seconds and 045 milliseconds**. After Sorted Adjusted Last to First: 0s 045ms $^{\text{(Note}}$ "To be clear this is 0 ms it uses end time - start time")

## Example Starting Points

Let's look at the specific cities involved in this process:

- **First Starting Point**: The first city in the sorted path. **{'name': 'City229', 'x': 500, 'y': 125}**
- **Second-to-Last Starting Point**: The city that was second to last in the sorted array before adding the first city again. **{'name': 'City732', 'x': 899, 'y': 169}**
- **Last Starting Point**: The first city that was appended to the end of the sorted array to form a closed loop. **{'name': 'City229', 'x': 500, 'y': 125}**

## Why This Matters

By including the first city at the end of the sorted path, we ensure that the path is a complete loop, which is essential for many algorithms solving the Traveling Salesman Problem (TSP). The timestamps provide insight into the efficiency of these operations.

This explanation provides a clear context and detailed breakdown suitable for a testing env.

### Why It's Related to P vs NP

The correlation between the time taken for sorting and the concept of **P vs NP** can be understood as follows:

- **P-Class Problem**: Sorting an array is a problem that falls into class `P`, meaning it can be solved efficiently in polynomial time. The time taken to sort and adjust the cities in our example (0s 045ms) demonstrates a task that a computer can perform quickly.
- **NP-Class Problem**: Verifying the sorted path and ensuring it forms a closed loop is a task that can be verified quickly. If we were to introduce a more complex problem, such as finding the shortest path that visits all cities

## Time Correlation

In this context:

- **Initial Function Execution**: Gathering the initial timestamps and sorting the array reflects how efficiently the algorithm performs a known polynomial-time task.
- **Post-Processing**: Adding the first city to the end of the sorted array is a quick verification step, analogous to checking a solution in an `NP` problem.

If the array of cities was already pre-loaded, the time taken to process the data would be faster because the computationally expensive sorting step would be skipped, leaving only the verification and adjustment steps.

This explanation should provide a comprehensive understanding of the **P vs NP** problem and how the time correlation in sorting and verifying city paths relates to it. If you have further questions or need additional clarification, feel free to reach out!

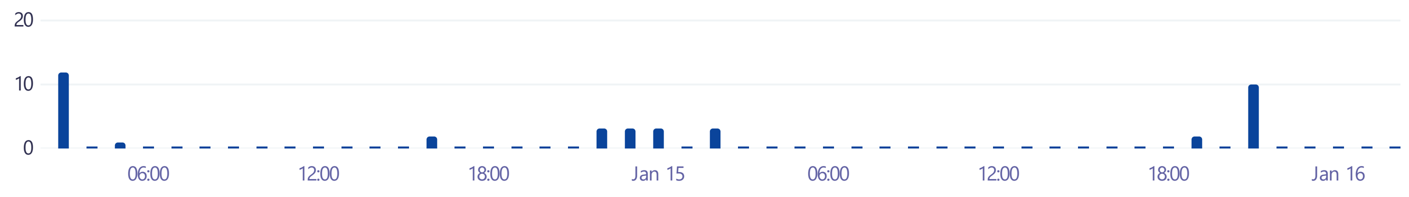
Find the code on my github called zap in folder TravelingCitiesProblem, don't forget to support me.
https://github.com/BadNintendo/tsp/blob/main/TravelingCitiesProblem/zap.py

Post Insights

Only the post author and moderators can see this

| | | | |
|---|---|---|---|
| 39 | 100% | 0 | 0 |
| Total Views | Upvote Rate | Comments | Total Shares |

Hourly views for first 48 hours


06:00
12:00
18:00
Jan 15
06:00
12:00
18:00
Jan 16