IE 343- TERM PROJECT

The purpose of this code is to demonstrate algorithmic solutions for selecting a set of songs within a given time constraint and optimizing a tour route for a list of cities. The primary algorithms employed are the Knapsack algorithm for song selection and the Nearest Neighbor algorithm for the Traveling Salesman Problem (TSP). For the Nearest Neighbor Algorithm for TSP, our aim is to find an approximate optimal tour route by selecting the nearest city at each step. In our Knapsack Algorithm the purpose is to select a subset of songs with the maximum total popularity within a specified maximum duration. As our greedy heuristic algorithm for Knapsack, our chosen heuristic is song's popularity is divided by song's duration. Then we sorted the songs based on the popularity/duration ratio. We wanted to maximum popularity per unit duration. And then, our code does iteration over sorted song list and select the best songs until maximum duration is reached.

We think that using greedy heuristics can be advantageous. For instance, I can give some positive sides easily: Greedy algorithms are fast, allowing you to process large datasets efficiently. In addition, greedy algorithms can be advantageous for minimizing the total duration of computations. As they are efficient, they contribute to faster decision-making and processing times. It is obvious that the efficiency and quick decision-making of greedy algorithms make them suitable for scenarios where computational resources and time constraints are significant factors.

In addition to that, if we talk about runtime analysis of our codes, the code executed in 0.07959198951721191 seconds. The runtime efficiency is influenced by factors such as the number of songs, the duration constraint, and the number of cities.

Finally, if we want to give some information about the time complexity, as we mentioned before, in our Knapsack problem we used greedy heuristics. After sorting, the algorithm selects songs greedily, meaning it picks the best ratio for that one step and that one step only without minding the further steps, and does this until the maximum duration is reached.

Number of songs is defined as “n” and constructing the “song_densities” list takes O(n) time. Sorting this list based on total population/ (average population * duration) ratio takes O(n log n) time. The iteration through the sorted list takes O(n) time. In the worst-case scenario, it loops through all songs once. Therefore, the overall complexity is O(n log n). The algorithm sorts n songs based on their popularity /duration ratios and then selects the songs greedily in each step. This makes the time complexity mainly a sorting step.

Our Traveling Salesman Problem follows the Nearest Neighbor algorithm. This algorithm starts from a source city and iterates through each city and calculates distances to all other unvisited cities to find the nearest neighbor and has a time complexity of O(n^2) where “n” is the number of cities.

Inside this loop, there is another loop that iterates over all cities to find the nearest unvisited city. The inner loop runs “n” times in the worst case. This algorithm provides a fast heuristic solution for the Traveling Salesman problem but lacks accuracy as it might not yield the most optimal tour.

Deniz Şener S024436
Aslı Kartal S029215