DATA STRUCTURES AND ALGORITHMS FOR DATA ENGINEERS - ASSIGNMENT 1

Bagirishya Rwema Dominique

CANEGIE MELLON UNIVERSITY bdominiq@andrew.cmu.edu

First and foremost, the problem tasked with involved developing a C++ program to address the processing of fixation points obtained from automatic eye trackers. These fixation points, which represented the user's visual attention, included identifiers, x and y coordinates, forming a scanpath. The primary goal was to create a unique identifier for each distinct fixation point based on its coordinates. According to what has been explained during recitation and in the document, the program had to handle multiple test cases, each terminated by the coordinates (-1,-1), and output a list of distinct fixation points with their corresponding unique identifiers. The uniqueness of a fixation point was determined solely by its x and y coordinates, and the identifiers started from 1, incrementing based on the order of encounter. I used sample input that was given in the question sheet, but it can be tasted with different one.

So, my code is intended to process fixation points from an input file (one that is located in the data directory of my zip file), identify distinct points, and write them to an output file. To test the functionality, a set of fixation points was provided in the "data/input.txt" file. The program read the input, identify distinct fixation points, and then write the results to the "data/output.txt" file. After executing the program, the output file contains the distinct fixation points for each test case. (you can also use different test cases).

Down here are my test cases & output:

Input	Output
2	bdominiq
1 382 353	1 382 353
2 484 328	2 484 328
3 995 641	3 995 641
4 715 242	4 715 242
5 995 641	5 710 245
6 995 641	**********
7 710 245	1 200 300
8 715 242	2 400 500
9 995 641	3 600 700
10 -1 -1	4 800 900
1 200 300	5 1000 1100
2 400 500	6 1200 1300
3 600 700	**********
4 800 900	
5 200 300	
6 600 700	
7 1000 1100	
8 800 900	
9 1200 1300	
10 -1 -1	

In my approach, I undertake the task of identifying distinct fixation points from this input file. The algorithm begins its journey by scanning through each fixation point listed in the input file. As I delve into each point, I remain vigilant for the telltale sign that marks the end of our data - the coordinates -1 for both x and y. With each fixation point encountered, I embarked on a quest to discern its uniqueness. Then scrutinize the new arrival's coordinates, comparing them to those of its predecessors. Should the newcomer bear coordinates unlike any seen before, it earns the honor of being branded as distinct. With a swift stroke, it is adorned with a unique identifier and welcomed into the esteemed league of distinct fixation points. This process unfolds tirelessly until every fixation point in our dataset is meticulously examined. Then, I transcribe the saga of distinct fixation points into the annals of an output file. Each point's unique identifier, along with its coordinates, is carefully inscribed, ensuring that its tale is preserved for posterity. This narrative unfolds seamlessly for each test case within the input file, ensuring that my output captures the essence of uniqueness while delineating each test case with termination markers. Thus, my approach encapsulates the essence of and precision, ensuring that my output serves as a beacon of clarity amidst the vast sea of data.

In my test strategy, I devised several test cases to evaluate the functionality and robustness of the fixation point processing algorithm. The first test case involves a scenario with a variety of fixation points, including duplicates, and terminates with the standard -1 coordinates. This case aims to assess the algorithm's ability to correctly identify and record distinct fixation points while handling termination conditions appropriately. The second test case presents a different set of fixation points, with larger coordinate values. This test evaluates the algorithm's scalability and accuracy in handling different input sizes and ranges. Both test cases include the maximum number of fixation points (10) to verify the algorithm's capability to process a full set of data.

Lastly, for complexity, the algorithm exhibits a time complexity of O(N^2), where N represents the number of fixation points in the input. This complexity arises from the nested loop structure used that I used to compare each new fixation point with all previously encountered fixation points to check for distinctness. As the number of fixation points increases, the algorithm's execution time grows quadratically due to the need to compare each fixation point with all preceding ones.