Due date: February 18, 2025

Problem 1: Distance & Metrics. Which of the following distance functions covered in the class are metrics: (a) L_p-distance, (b) Minkowski distance, (c) Mahalanobis distance, (d) cosine distance, (e) edit distance, (f) Fréchet distance, (g) dynamic time warping.

Justify your answer. If a distance function is a metric under some conditions, state that condition and justify it.

Problem 2: Polynomial-time 1D k-means. In general k-means problems are **NP**-Complete, if you still remember what that means from CS330. Therefore, we currently do not know any polynomial-time algorithm that computes the *global* optimum.

However, it turns out if the input $\{X = x_1, \ldots, x_n\}$ is 1-dimensional, then this special case of k-means admits a polynomial time algorithm. Propose one such algorithm. [*Hint: sort the input. Hint 2: dynamic programming (other approaches may work too).*]

Problem 3: Huffman Coding Implementation. In this problem you will implement Huffman coding. You are provided the short novel *The Strange Case Of Dr. Jekyll And Mr. Hyde* by Scottish author Robert Louis Stevenson. A plain text version is provided under `Modules/Assignments/CS390_HW2_JH.txt`.

Your final deliverable consists of a write-up (like any other questions) as well as the source code. You may choose any programming language you prefer, as this problem will be graded mostly on the write-ups (we'll still briefly check the code to ensure that no one is encoding everything by brute force!). Please implement the following four modules. You may choose any function signature and return type.

- **ComputeFrequency**. Takes a text corpus as input and computes the frequency distribution of each character.
- **BuildHuffmanTree**. Takes the character frequency distribution as input and converts it into a Huffman encoding tree. You may find it helpful to implement a Huffman class in order to organize your code. You will be using the novel to build this tree.
- **Encode**. Takes a Huffman encoding tree and a string message consisting of existing characters in the Huffman encoding tree as inputs. Outputs a binary string that uniquely represents this message according to the Huffman encoding tree.
- **Decode**. Takes a Huffman encoding tree and a binary string as inputs. Outputs a message as a string of existing characters. This decoding should be unique.

In your write-up, please answer the following:

(1) Compute the frequency distribution for all characters that appear in the novel. How many unique characters appear, and how many nodes does your Huffman tree require?

(2) Build the Huffman tree for the novel. Display the first three layers of the tree in a structured format (e.g. describing the tree top down). For each node in these layers, list the corresponding character and its binary encoding.

(3) Encode the entire novel, then decode it. Show a few example outputs and verify that the decoded text matches the original input. If done correctly, after decoding, you should re-obtain an exact copy of the original file. Verify this.

(4) Measure the runtime of each module. If you identify a bottleneck, what are some potential improvements?

(5) Calculate the average number of bits per character in your Huffman encoding. Compare this against the theoretical entropy of the character distribution.