### ****Project: Greedy Clustering Using Kruskal’s Algorithm for Medical Image Classification****

#### ****Objective:****

The goal of this project is to cluster **7 medical images** (representing **normal and tumor cells**) into **two distinct categories** based on their **similarity scores**. We achieve this by implementing a **greedy clustering approach** using **Kruskal’s Algorithm** for **Minimum Spanning Trees (MST)**. This ensures that we efficiently group the images in **O(n log n) time complexity** using a **Union-Find (Disjoint Set) data structure** for optimal merging.

### ****Problem Statement:****

Given a **distance matrix** that represents the **similarity scores between different medical images**, we need to **form two clusters (k=2)** that group the most similar images together while separating distinct groups (normal vs. tumor cells). The challenge is to do this in an **efficient manner (O(n log n))** while ensuring that the clustering accurately reflects the given distances.

#### ****Distance Matrix (Similarity Scores)****

Each row and column in the **7x7 matrix** represents an image, and the values denote the **similarity (distance) between them**. The **lower the distance, the more similar the images are**. The goal is to group **highly similar images into the same cluster**.

*distance\_matrix = [*

*[0, 38, 17, 28, 88, 59, 13],*

*[38, 0, 52, 49, 83, 91, 59],*

*[17, 52, 0, 46, 34, 77, 80],*

*[28, 49, 46, 0, 5, 53, 62],*

*[88, 83, 34, 5, 0, 43, 33],*

*[59, 91, 77, 53, 43, 0, 27],*

*[13, 59, 80, 62, 33, 27, 0]*

*]*

### ****Approach: Kruskal’s Algorithm with Union-Find****

To cluster these images efficiently, we use **Kruskal’s Algorithm**, which is a **greedy method** for constructing a **Minimum Spanning Tree (MST)**. The algorithm **sorts the edges** and **merges the closest nodes first** while avoiding cycles. Since Kruskal’s method naturally groups **closely connected nodes together**, we can **stop merging** once we have exactly **k=2 clusters**.

#### **Steps Involved:**

1. **Extract all edges** (similarity scores) from the upper triangular portion of the matrix to avoid duplicates.
2. **Sort the edges** in **ascending order** (smallest distances first).
3. **Initialize Union-Find** data structure for efficient cluster merging.
4. **Apply Kruskal’s Algorithm**:
   * Start with **each node as its own cluster**.
   * **Merge the closest pairs** until only **2 clusters remain**.
   * Use **Union-Find** to track cluster memberships efficiently.
5. **Output the final clusters**, where each cluster contains a **set of image indices**.

### ****Implementation Requirements & Constraints****

#### **✅ Complexity Constraint:**

* The algorithm must run in **O(n log n)** time.
* **Sorting edges takes O(n log n)**, and **Union-Find operations take nearly O(1)**.

#### **✅ Union-Find Data Structure:**

* Do **not** use external libraries (must implement Union-Find in 10 LOC).
* Use **Path Compression** for efficient find() operations.
* Use **Union by Rank** for optimal merging.

#### **✅ Grading Criteria:**

* **(-2)** if the time complexity exceeds **O(n log n)**.
* **(-2)** if an external library is used for Union-Find.
* **(-2)** if the clustering is incorrect.

### ****Expected Output:****

After executing the algorithm, we expect **two clusters** that group similar images together.

**Final Cluster Output (Example Result):**

Cluster 1: Images {1, 3, 4, 5, and 7}

Cluster 2: Images {2 and 6}

(Note: Image indices are **zero-based in Python**, but human-readable indices start from **1**.)

### ****Conclusion****

This project effectively **demonstrates greedy clustering** using **Kruskal’s Algorithm** and **Union-Find** to efficiently partition medical images. The approach ensures that **similar images remain grouped together** while keeping the solution computationally efficient. By following this method, we achieve an optimal clustering in **O(n log n)** time without unnecessary overhead.