Homework 4 Write-Up

Alex Behm
NetID: abehm815

1. Adjacency List Representation

I represented the adjacency list using a `Map<T, Map<T, Integer>>`. The outer map stores each vertex as a key, and maps to another map which stores all its neighbors and their corresponding edge weights. This representation is efficient for checking vertex existence, neighbor lookup, and updating edges.

2. Edge Weight Assignment

When converting the image to a graph, each pixel becomes a vertex. I added directed edges to the pixel's four immediate neighbors (up, down, left, right). If the pixel brightness is above a threshold (240), it is considered white and the edge weight is 0. Otherwise, the edge weight is set to 1 to represent a darker pixel, indicating the presence of text.

3. Dijkstra Usage

To adhere to the requirement of using a constant number of Dijkstra invocations, I added a virtual source node connected to all white pixels on the left and top edges of the image with weight 0. I then ran Dijkstra's algorithm once from this virtual source node. A row or column is considered whitespace if all its pixels are bright and reachable from the source with distance 0.

4. Heap Implementation

For Dijkstra's algorithm, I used Java's built-in `PriorityQueue`, which is part of the standard Java Class Library. No external libraries were used, and no licensing or attribution is required.