Hyder Shad
hs25796
EE 360C
4/2/2019

Programming Assignment #2 Report

a) From a space-complexity perspective, an adjacency list would be the best method because while each node (pixel) is connected via an edge to its adjacent nodes, of which there can be a maximum of 4 (above, below, left, right) and a minimum of 2. Creating an adjacency list for N pixels of which there are at maximum 4 edges per pixel and at minimum 2 would result in less than Nx4 memory locations to represent the adjacent vertices to a particular vertex. This can be significantly less that the N^2 memory required to create an adjacency matrix for any image of size N. A large image (large N-pixels) would be considered a relatively sparse graph if each node has a max of 4 edges, so an adjacency list is the best method in terms of memory usage.

   From a time-complexity perspective, you need to go through each node (pixel) and find the edge weights associated with it and sum up the total for all nodes in the image graph. You would have to go through either the linked list of each pixel for an adjacency list or the array of each pixel for the adjacency matrix to determine the corresponding edge weights. In this case, a large image would be considered a relatively sparse graph, so an adjacency list would technically be quicker for larger images of N-pixels since you wouldn't waste time iterating through a whole array of say size 100 when each pixel can only have 4 possible adjacent edges.

b) I first created a matrix called "mstSet" of the same dimensions as the input graph/image and initialized all values in the matrix to zero except the origin at mstSet[0][0], which I set to 1. This matrix represents the pixels that have been "added" to the minimum spanning tree, even though I do not create or return an MST and am just interested in finding the sum of the edges that would be part of the MST if created.

   In a set of nested for-loops corresponding to the row and column of the mstSet matrix, I go through each location in the mstSet matrix and if the location is a 1, I search all neighboring pixels (nodes) in the image graph to either the left, right, above, or below that pixel which are not yet in the mstSet matrix (the zeroes in the mstSet matrix) and analyze the edge weights. I then find the minimum edge weight between these adjacent pixels and if it is the smallest edge weight out all the adjacent pixels not yet added to the MST, the location of that adjacent pixel is corrected in the mstSet matrix from a zero to a 1 to indicate it has been added and the corresponding edge weight is added to the total edge weights for the MST.

   This search through the mstSet matrix continues until all locations in the matrix have been turned from zero to a 1, indicating that every pixel/node has been added to the spanning tree. The total edge weights for this process are then returned.

c) The program goes through the entire MST matrix for each pixel. Therefore, the O(n) time complexity in terms of the number of pixels, p, is O(n) = p.