Machine Problem 1 MST Algorithm

Object Description:

Edge:

Contains source, destination and weight of an edge

Has getter and setter methods for all its contents

Color/Pixel:

Contains RGB values

Method for getting average RGB difference between two Colors/Pixels (returns float aveRGB)

Method for getting weight between two Colors/Pixels (returns float pixWeight)

WQU/Disjoint Sets:

Contains predecessor array and size of tree (initially each node is a predecessor of itself and each tree has size 1) each with V elements

Method for determining which set a node belongs or simply getting root of a node (returns int root); implemented with path compression

Method for combining two sets (effectively updates predecessor array and size of subtree)

Graph:

Contains an adjacency list (has V nodes) of Edge objects; note that two distinct nodes can only be neighbors if they are exactly one pixel away from each other

Contains pixel list (has V nodes) of Color/Pixel objects

Elements from any lists can be conveniently accessed

Has method for conversion of image to graph and vice versa

Has method algorithm 1 which modifies the graph into an MST

Note: text in bold and italics means that a class method/container was used

```
algorithm modifiedMST()
initialize Graph G with v Pixels
initialize Disjoint Set D with v component trees, one for each vertex in G
while v is greater than parameter k:
    create/reset flag: no decrease in component tree count is true
    initialize the shortest edge from any node to its neighbor to INFINTY
    for every node n:
        for every neighbor of node m:
            ignore if m and n have the same root in D
            find minimum crossing edge e from m to n
            if shortest edge of n is greater than e
               set e as shortest edge of n
            if shortest edge of m is greater than e
               set e as shortest edge of m
    for every component tree whose shortest edge e is not INFINITY:
        ignore edge if e.source and e.destination has same root in D
        ignore edge if e.weight is greater than or equal to parameter T1
        ignore edge if average RGB difference between Color of e.source and Color of e.destination
        color is greater than or equal to parameter T2
        get updated Color value c using weighted averaging
        //Explained at a later part
        change Color of the root of e.source in D and the Color of the root of e.destination in D
        to updated Color value c
        combine set of e.source's root in D and e.destination's root in D
        //This is the same as adding e to the MST
        change flag: no decrease in component tree count to false
    for every node n:
        reset the shortest edge to INFINTY
        change Color of node n to the Color of n's root in D
    for every edge e in G:
        ignore if e.source and e.destination has same root in D
        new edge weight is the weight between the Color of e.source's root in D and the Color of
        e.destination's root in D
        updated e.weight is original edge weight multiplied by the new edge weight
    if the flag is true break the outermost loop
```

Additional note regarding weighted averaging:

In this formula, the containers pixel list in Graph and size of tree in Disjoint Sets was used. Given any two nodes, u and v, the updated color value is equal to p divided by t where:

p = (u^{th} pixel RGB values * u^{th} tree’s size) + (v^{th} pixel RGB values * v^{th} tree’s size)

t = u^{th} tree’s size + v^{th} tree’s size