Image Quantization Analysis

T055

# GetDistinctColorsList(ImageMatrix):

private static List<int> GetDistinctColorsList(RGBPixel[,] ImageMatrix)

{

HashSet<int> distinctColors = new HashSet<int>();

foreach (RGBPixel pixel in ImageMatrix)

distinctColors.Add(RGBPixel.Hash(pixel)); // O(N^2) if C# handles resizeing well

// Can we do better?

List<int> colorsList = distinctColors.ToList();

colorIndices = new Dictionary<int, int>(distinctColors.Count);

for (int i = 0; i < colorsList.Count; i++) // O(D)

colorIndices.Add(colorsList[i], i);

return colorsList;

}

Final Order: O(N^2)

# ReduceImageColors(ImageMatrix, ColorPallette, ClusterIndices):

// EXPECTED TO CHANGE

private static void ReduceImageColors(RGBPixel[,] ImageMatrix, Dictionary<int, RGBPixel> ColorPallette, Dictionary<int, short> clusterIndices)

{

int rows = ImageOperations.GetHeight(ImageMatrix);

int columns = ImageOperations.GetWidth(ImageMatrix);

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

RGBPixel currentColor = ImageMatrix[i, j];

int currentColorIndex = colorIndices[RGBPixel.Hash(currentColor)];

int currentColorClusterIndex = clusterIndices[currentColorIndex];

RGBPixel newColor = ColorPallette[currentColorClusterIndex];

ImageMatrix[i, j] = newColor;

}

}

}

Final Order: O(N^2)

# Bonus 1 (Finding K using MSDR):

// EXPECTED TO CHANGE.