Image Color Quantization

Members

1-Mohamed Madney Hassib sec 15

2- Mohamed kamal Fawzy sec 14

3-Mustafa Aly Abd Meniem sec 17

4- Youssef Mohamed Elwahsh sec 20

**Distinct List**

public static List<RGBPixel> distinct\_arry(RGBPixel[,] ImageMatrix , int width , int height)

{

bool[ , , ] arry = new bool[256, 256, 256];

List<RGBPixel> Distinct = new List<RGBPixel>(width\*height);

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

{

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

{

int red = ImageMatrix[i, j].red;

int green = ImageMatrix[i, j].green;

int blue= ImageMatrix[i, j].blue;

if (arry[red, green ,blue] == false)

{

Distinct.Add(ImageMatrix[i, j]);

arry[ImageMatrix[i, j].red, ImageMatrix[i, j].green, ImageMatrix[i, j].blue] true;

}

}

}

return Distinct;

}

**Distinct list**

Function Paramter: -

Imagematrix[width,hight];

Function Return: -

List<RGPpixel>;

Function Analyze:- First loop = number of iteration \* order body

First loop = Hight(N)\*Order second loop

Second loop = Width(N)\*Order of body

List.Add() => O (1), Assigning of Array O(1)

Final Order => O(N^2)

Where N is Width and Hight (Number of color)

Minimum spraining tree

public class minuim\_\_spaining

{

public static bool[] marked;

public static SortedSet<Tuple<double,int, int>> Q = new SortedSet<Tuple<double, int, int>>();

static Tuple<int, int, int> [] colors;

static List<Tuple<double, int>>[] mst;

static int vertice;

public minuim\_\_spaining(List<RGBPixel> param)

{

int size = param.Count;

marked = new bool[size];

mst = new List<Tuple<double, int>>[size];

colors = new Tuple<int, int, int>[size];

vertice = size;

int i = 0;

foreach (RGBPixel x in param)

{

mst[i] = new List<Tuple<double, int>>();

colors[i]=(new Tuple<int, int, int>(Convert.ToInt32(x.red), Convert.ToInt32(x.green), Convert.ToInt32(x.blue)));

i++;

}

}

public static double cal\_weight(Tuple<int,int,int> x , Tuple<int,int,int> y)

{

double Euclidean = 0;

int red = x.Item1 - y.Item1;

int green = x.Item2 - y.Item2;

int blue = x.Item3 - y.Item3;

Euclidean = (red \* red) + (green \* green) + (blue \* blue);

Euclidean = Math.Sqrt(Euclidean);

return Euclidean;

}

public static void prim (int x ){

Tuple<double, int, int> y;

double cost = 0;

double max = -1;

Tuple <double,int, int> edge;

Q.Add(new Tuple<double, int , int>(0,0,x));

int count = 0;

while(Q.Count != 0)

{

// Select the edge with minimum weight

edge = Q.First();

Q.Remove(edge); //logd

x = edge.Item3;

// Checking for cycle

if (marked[x] == true)

{

continue;

}

marked[x] = true;

//form out the tree to inside the tree

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

{

if(marked[j] == false)

{

if(count == 0)

{

if (max < cost)

{

max = cost;

cost = cal\_weight(colors[x], colors[j]);

Q.Add(new Tuple<double, int, int>(cost, x, j));

}

}

if(cost > max)

{

continue;

}

cost = cal\_weight(colors[x], colors[j]);

Q.Add(new Tuple<double, int, int>(cost, x, j)); // Upper Bound of the Complexity

}

}

if (vertice-1 == count)

break;

y = Q.First();

int a = y.Item2;

int b = y.Item3;

double weight = y.Item1;

mst[a].Add(new Tuple<double, int>(weight, b));

mst[b].Add(new Tuple<double, int>(weight, a));

count++;

}

}

Minimum spraining tree

Constructor: -

Initialize Marked Array Visited

Initialize Mst list of minimum spainng tree

Initialize color Array of tuble Which contain the Distinct color in Distinct list

Foreach loop Number of Iteration (Distinct List) \* convert.to int() o(1)

O(D)

Function Cal\_weight: -

Return Distance between two colors

O(1)

Function prim: -

Construct minimum spainng tree of Distinct array

SortedSet.Add(); => O(Log E)

While loop => Number of iteration \* Order of Body

While Loop => Edges(E) \*Order of Body

SortedSet.First(); => O(1)

SortedSet.Remove(); => O(Log E)

For loop => Number of iteration \* Order of Body

For loop => Vertices(V) \* Order of Body

Note : First Iteration Only => Vertices(V) \*(Cal\_whight + SortedSet.Add(); )

Note : First Iteration Only => Vertices(V) \*(O(1) + O(Log Edges(E))).

Next Iterations

Number of iteration \* Order of Body

Vertices(V) \* Order of Body

Order of Body => Cal\_whight + SortedSet.Add();

Order of Body => (O(1) + O(Log Edges(E))).

Ending of For Loop

Remaining part of While

Sortedset.First(); => O(1)

mst[a].Add(new Tuple<double, int>(weight, b));

mst[b].Add(new Tuple<double, int>(weight, a));

Assigning of MST[] => O(1)

Adding of the List => O(1)

End Of While

Complexity Equation :-

Log(E) + E(Log(E) +V( O(1) + O(Log(E)))+O(1)) +V( O(1)+O(Log(E)).

Final Complexity => O(EV Log(E))

End of Prim Function

Milestone One

Algorithm Design And

Analyses

Project