Image Quantization Project

**Team No. 49**

**Team Members:**

**AbdelRahman Mohamed Ismail sec(11)**

**AbdelRahman Mohamed Ashraf sec(11)**

**AbdelRahman Ahmed AbdelAziz sec(10)**

**AbdelRahman Hassan Mohamed Ali sec(11)**

**Abdallah Ashraf Abdallah sec(11)**

**Introduction**

The project should make quantization to image using some algorithms that take every pixel of the image and do some operations on it which is called "Clustering".

The goal of this project is to minimize the space or capacity of the image, but the quality still the same by using Prim Minimum Spanning Tree Method and Linkage Clustering.

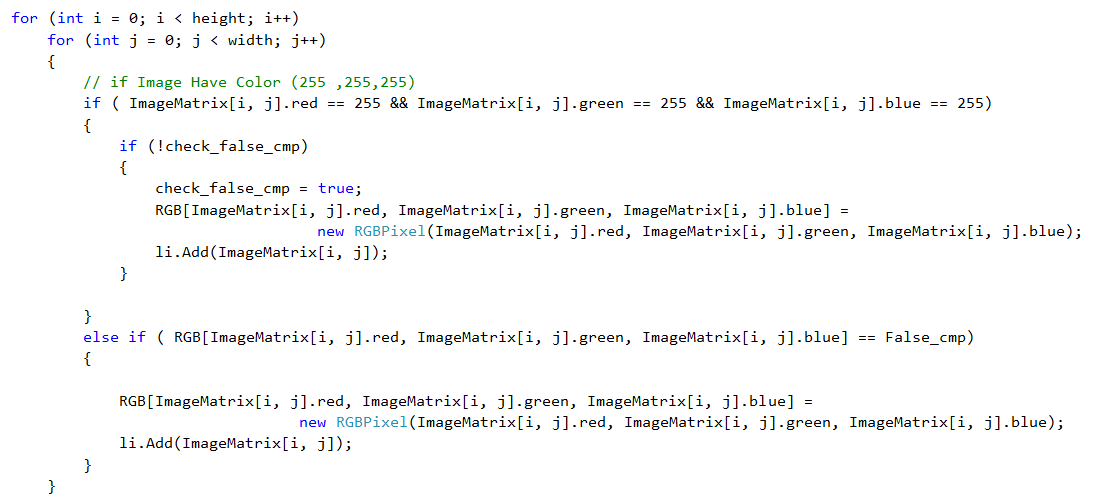
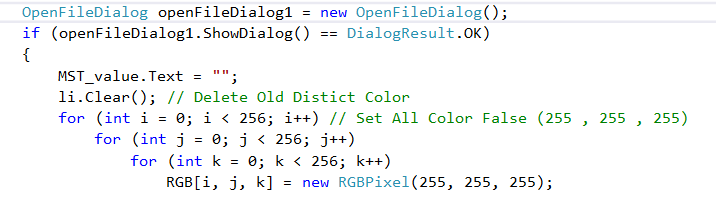
**Milestone Requirements:**

* Creating list of distinct colors of the image (RGB).
* Calculating the distance between each colors of the pixel color and the other pixels.
* Create MST Algorithm "Prim (Using Queue Binary Heap) that calculates the shortest path of the graph of pixels.

**Distinct List of Colors:**

At first, it is implemented that there is a template that takes the path of image and show it in a picture box.

So it has been created a Multidimensional Boolean array "RGB Array" (3 Dimensions) with all the possible RGB values. then we take the image matrix (2 Dimensional array of RGB Pixel type), then if the value of image matrix does not marked in the "RGB array" as visited we mark it as true and continue otherwise.



The analysis of this code is O(N2):

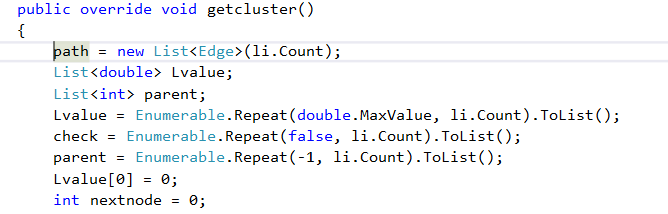
* The outer loop is O(n).
* The inner loop is O(n).
* the statements inside the inner loop O(1).

***The minimum Spanning Tree "Prim MST":***

***The algorithm start with a specific node as a start node, then it gets the minimum cost to the next node till we reach the last node in the graph.***

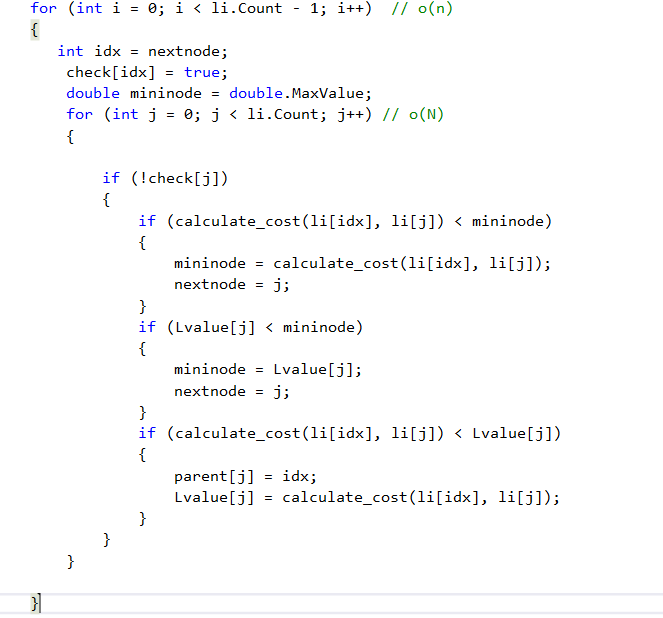
***The algorithm runs as we take all the distinct colors from the list, calculate cost of every color and the other colors, taking the minimum color constructing the MST on the minimum cost of each color step by step.***

***The cycle is not allowed so we created a backup value to check if it smaller than the minimum value so we take it.***



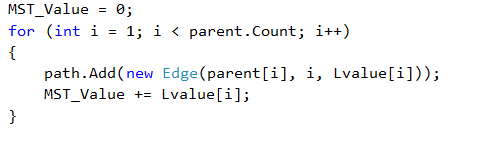
***Defining the variables we need and set an initial value to, moving all the elements with null before we get the distinct colors.***

***After that, we check every cost of color with other colors, taking the minimum, checking the parent and the new child cost, if it smaller with cost, take the new edge with parent and child, ignoring the edge that is calculated.***



***The complexity order of this code is O(N2).***

***After we calculate the minimum cost we set the parent index and cost value in a list "parent" "L value", then we add it into the path list of edges "The final MST".***



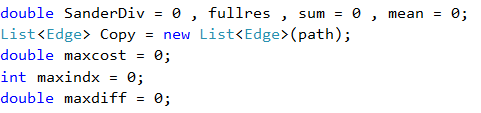
***Then the cost of the MST value is calculated. The complexity Order of this part is O(N).***

***The total MST function order is O(N2).***

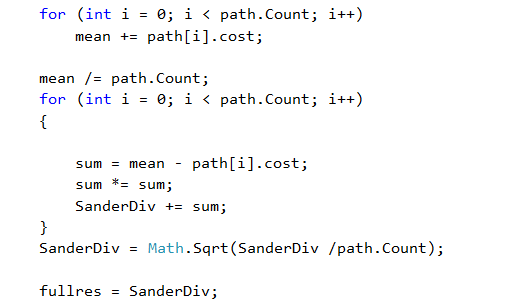
***The Auto-detect clustering:***

***As it is shown in the documentation the method is to calculate the standard deviation of all the distinct colors from the list "weights" and store it, then we remove the most efficient reduction in standard deviation, checking for every iteration if the difference between the current iteration and previous iteration is very small, then we are done at the optimal cluster.***

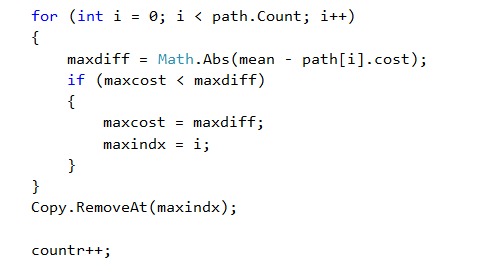
***We define the standard deviation, mean, copy of the MST and other variables that we will explain later with initial values or null.***



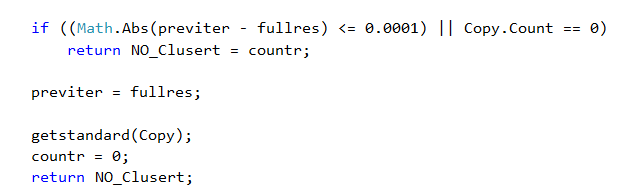
***So at first we calculate the mean and the standard deviation.***



***Then we choose the biggest difference of mean and weight in the MST and make a counter increase every time we remove edge.***



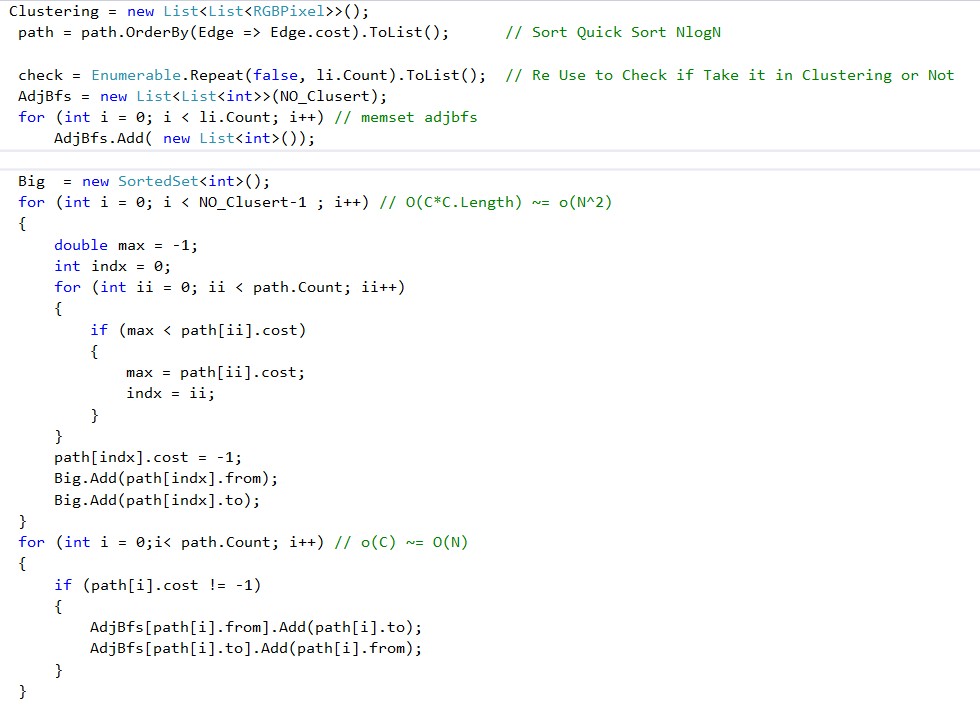
***After that we make the check of convergence at every iteration except the first one as we move it to previous iteration variable so it does not count.***



***The complexity of the code is O(N) "Non recursive" \* O(N) "Recursive".***

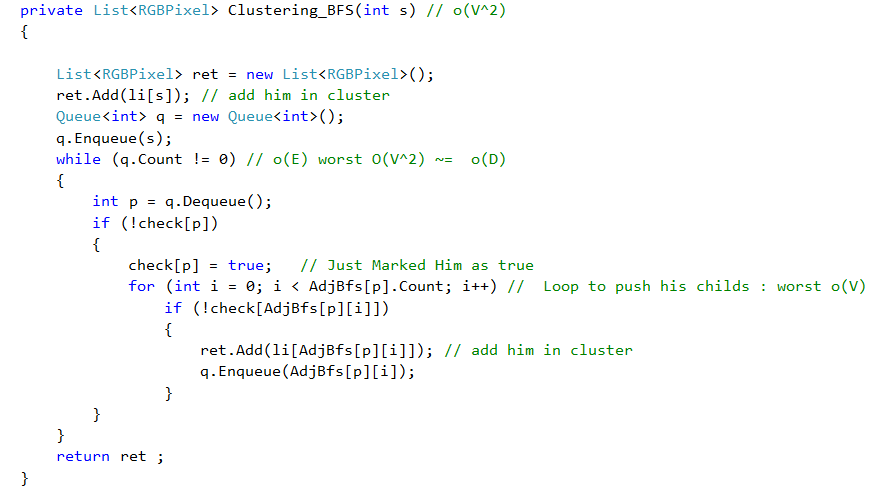
***The BFS Clustering:***

* ***The idea is to get the maximum edge, setting it to"-1"in the big "sorted set".***
* ***The "ADJBFS" is a list that has all the node and its relation with the other nodes.***
* ***The clustering is the final list that includes the clusters that is satisfy the numbers of clusters entered by user.***
* ***The path list has MST that should have the clusters from.***
* ***The check list is a Boolean list that check if the node visited or not***



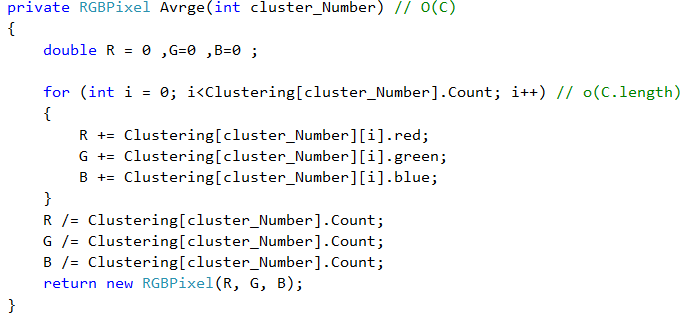
***The complexity of this code is O(N2).***

***The method takes the MST adding the related cost to a list by setting the max value of the MST to "-1"until the K clusters is satisfied by the number of clusters entered by user. The clusters added after checking the node not visited.***



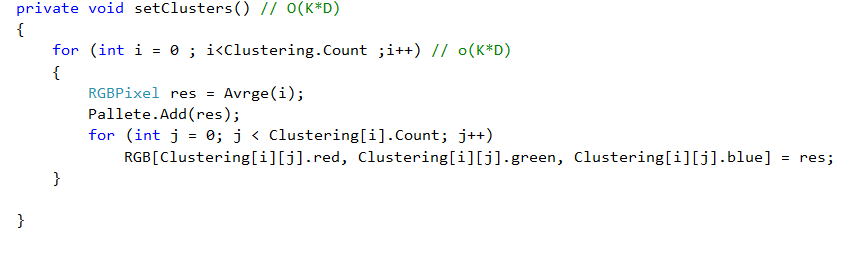
***The complexity of this code is O(N2) :N2 => D .***

***After getting the elements in each cluster we compute the average to return the average color to the colors in the cluster.***



***The complexity of this part is O(N).***

***After getting the clusters, we edit in the RGB matrix to load the picture pixel by pixel.***

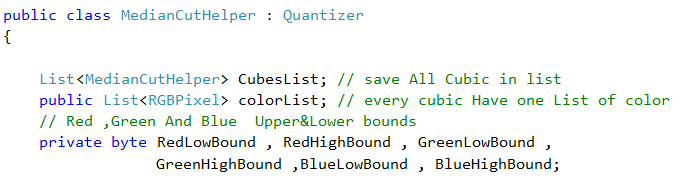


***The complexity of code is O(N2).***

***Median Cut:***

***The main idea of the median cut is to split the distinct colors we have into groups each time we split we make clusters.***

***the split process make the initial cut of the original list into two cubes. then each cube into so we have four cubes,...8,..16,...32,....2n.***

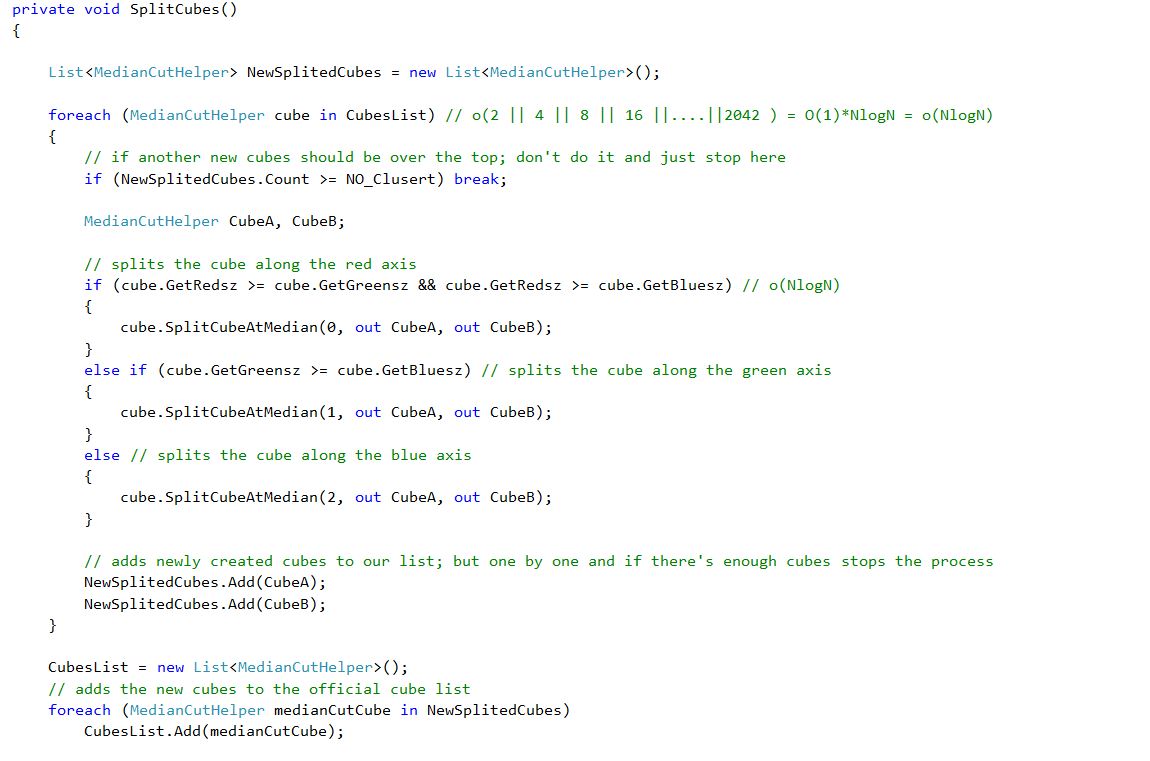


***The initial variables in the class is:***

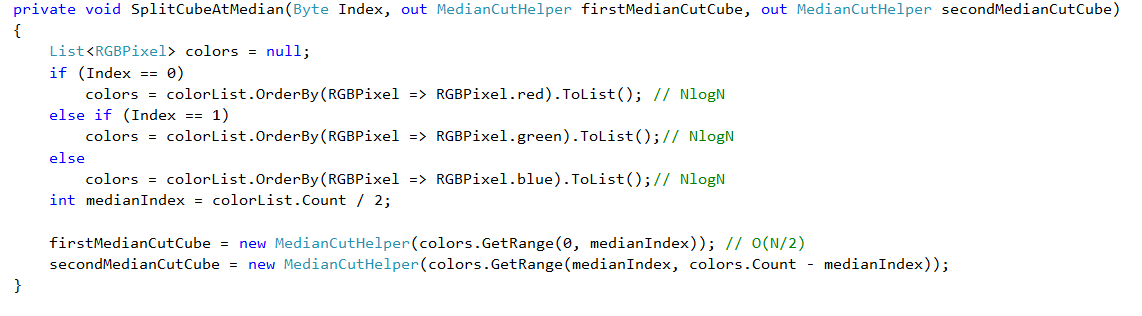
* ***Cube list: which have all the cube split and initially have the big cube that contains all the colors before split.***
* ***the color list : contains the color of each cube.***
* ***the low bound: The ground of each RGB, the minimum RGB the cube contains.***
* ***the high bound: the maximum RGB the cube contains.***

***Having declared the variables we take the big cube and split it into two cubes.***

***So we make a loop of each cube in the cube list we split it as we check the position of the cubes in Red section, Blue section or green section and set the index of the new cubes with 0,1 or 2 respectively to each RGB."O(N log N)".***



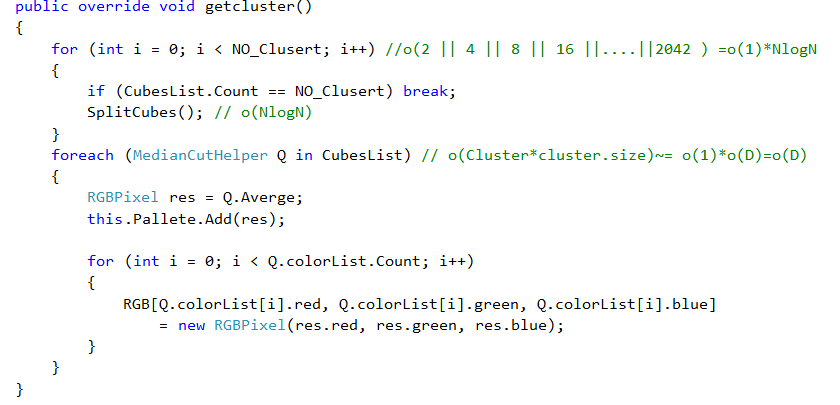
***Setting the index is the first step of calling Split at median function. The function cuts the cube after sorting it by the index that we set to before, then the cube is cut into two cubes added to the new split cubes, and finally added to cube list.***



***The complexity is (N log N).***

***So every time the cubes is cut until equals the desired number of clusters.***

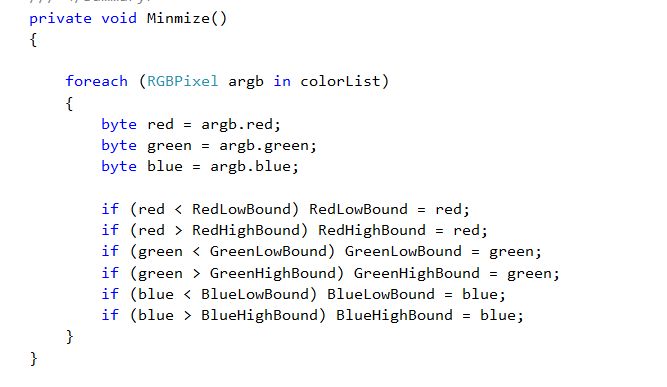
***After that the clusters is completed we just set each new cluster with the average color, changing the new clusters in the RGB array.***



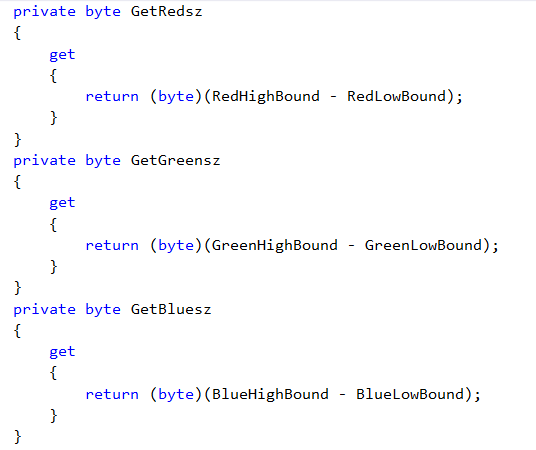
***The complexity is O(N2) :N=>V, N2=>D.***

***Additional functions in the median cut:***

***minimize: get the high and lower bound of the R,G,B "O(N)":N=>size color list.***



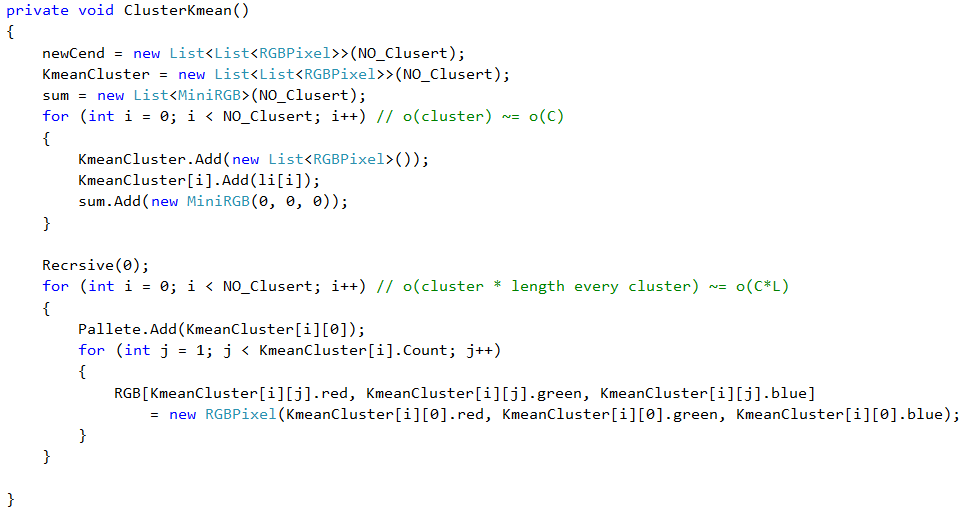
***get Reds, get Greens, get Blues: returns the difference between the high and low bound "The range of the cube".***



***Complexity: O(1).***

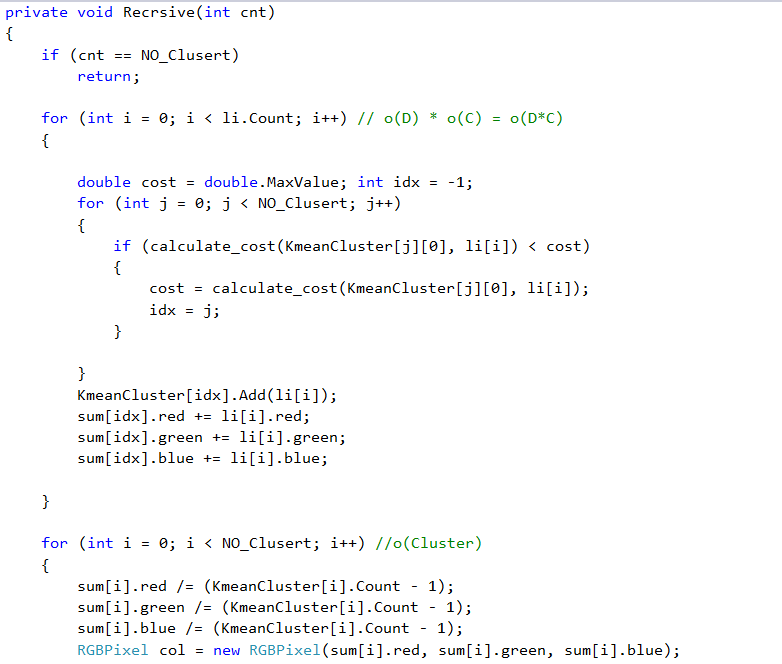
***K-MEAN:***

***The K-mean method is to grouping the colors into clusters taking the average of the colors in each cluster and check if it has the same average at the same index or the number of cluster extracted is greater than the desired one, it returns the clusters in the K mean cluster list.***

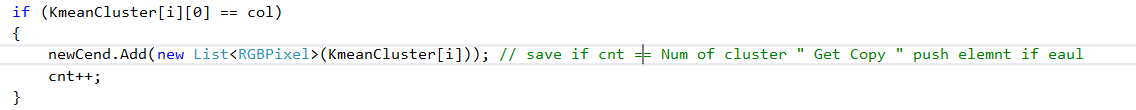


***The function sets the colors in the k mean list then calculates the average, putting the average in the first column of the list, then adding all the colors by the clusters after taking the values from the distinct color list.***

***Calculating the average of each cluster, put the new color in the first element of each cluster, then check the number of clusters reached the one entered by user, is called "Recursive" and return the clusters after setting it by the new average color.***

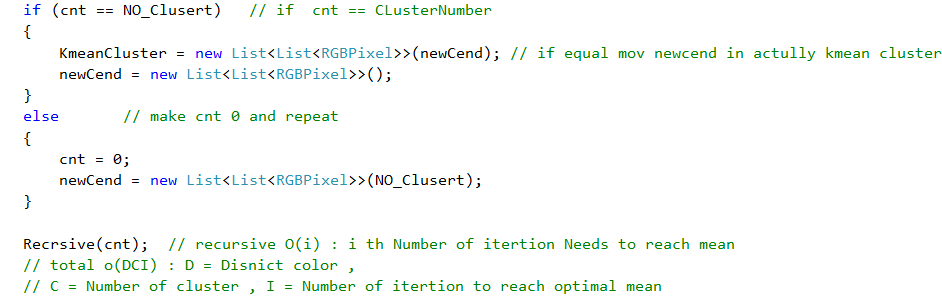


***The calculated average part of the previous code is O(D x C) that D is number of distinct colors and C is number of clusters.***



***Adding the clusters of the previous code after checking the average.***

***Complexity is O(1).***

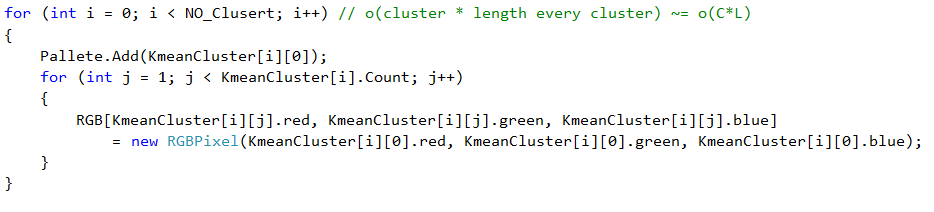


***Checking the number of clusters and returning the cluster list.***

***If the number of clusters is not the reached one, it begins the function again and compute clustering with the number of cluster.***

***The complexity of the function is O(DCI) : D=> Distinct colors, C => number of clusters , I => N (Number of iterations).***

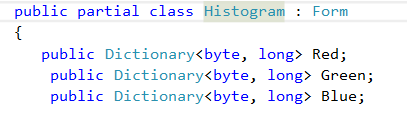
***After returning the clusters, the draw function is setting the RGB array with new values of the clusters "RGB".***



***The complexity of this part is O(C\*K): C=> number of clusters entered by user and K => number of computed clusters.***

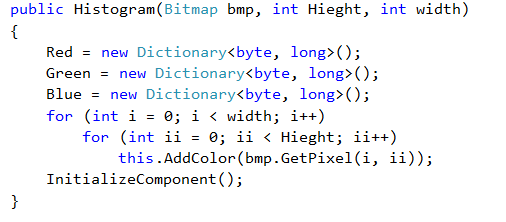
***Histogram:***

***The Histogram is a graph series that shows each R,G,B color of the picture whatever it is original or quantized.***

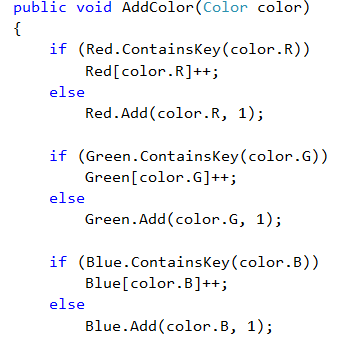


***We define the Red, Green and Blue as dictionary that contains the number of repetition the color in the image and the its position on the graph.***

***As we initialize the histogram, we put the colors in the bit map.***



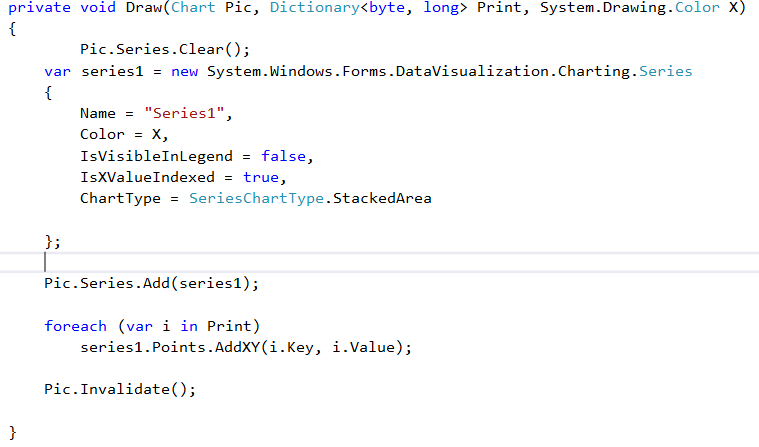
***Having initialized the color in bitmap, we check the R,G,B then whether the value of R is repeated then it increase the counter. It is also applied in the G and B colors.***



***Then, we draw the chart in the series. The series at first is cleared for any previous iterations. After that we set the color to one of R or G or B color.***

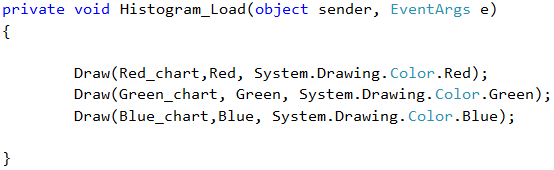
***The visible in legend is false as we do not want to define each chart, and X value indexed by true as we ignore the approximated values automatically.***

***Then we add the X,Y of the graph after setting it to the color that is repeated, then we draw the chart and load it into the form.***



***Well, the function "draw" needs the color chart, the values repeated of the color and its position.***

***The load function calls the function draw with all the above parameters of each R G B chart.***



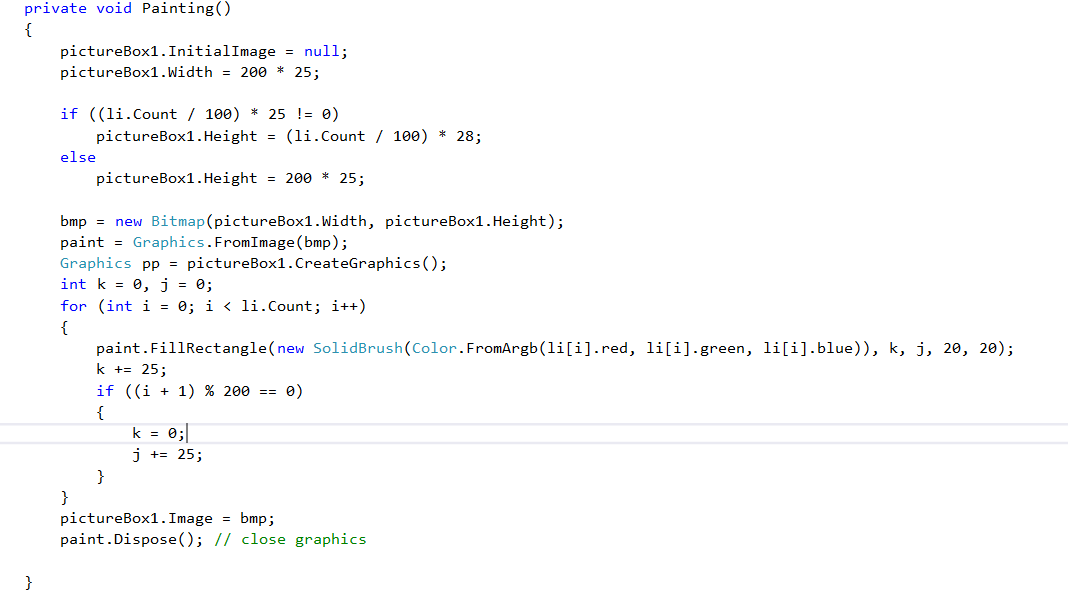
***Palette:***

***Palette is a form that shows all the distinct color from the picture. The palette is a tool to help the user extracting the color.***

***How to save display the colors:***

***The colors is set to the picture as a back color and the R,G,B is taken as a text.***

***The palette takes the list of distinct colors then create a graphics "paint" and fill the bitmap "BMP" then load it to the picture box.***



***The complexity of this code is O(D)= O(V2)***