|  |  |
| --- | --- |
| Faculty of Computer & Information Sciences  Ain Shams University  Subject: Analysis and Design of Algorithms  Year: (3rd year) undergraduate (CS)  Academic year: 2nd term 2021-2022 |  |

**Image Quantization Project**

**Team ID-T038**

**Department-CS**

TA-Mohamed Magdi

**Team Members:**

|  |  |
| --- | --- |
| Member ID | Name |
| 20191700729 | هادي إيهاب رجاء أحمد |
| 20191700728 | هادي أحمد عبد السلام عبد الحميد |
| 20191700730 | هادي عاطف سيد محمد |

**colorCodingClass**

1. codeColors() function:

parameters: “pixel” of type “RGBPixel”.

body: the whole color code of “pixel” is stored inside an integer variable named “enCodedColor”. This is done by adding the red color (represented in 1 byte) after being shifted to the left by 16 bits to the green color (represented in 1 byte) after being shifted to the left by 8 bits. Then, the result is added to the blue color. Finally, “enCodedColor” variable holds the code of the three colors together and it is returned at the end.

Function’s Order:

1. decodeColors() function:

parameters: “codedColor” of type integer which carries the whole RGB code.

body: “res” variable of type “RGBPixel” carries three attributes of type “byte”; “red”, “green” and “blue”. We set red to “codedColor” after shifted rightwards by 16 bits casted to byte. The same applies to green but 8 bits not 16. Finally, blue is set to “codedColor” casted to byte and at the end “res” variable is returned.

Function’s Order:

**getDistanceClass**

1. getEculideanDistance() function:

parameters: “src” & “dst” of type “Vertex” Class which carries the vertex and its parent

body: the vertices of “src” and “dst” are decoded using decodeColors() function to use red, blue and green separately and calculate the Eculidean distance between “src” and “dst” and return “res” variable storing this Ecuildean distance.

Function’s Order:

**ClusteringClass**

1. generatePalette() function:

parameters: “dis” List of integers, “mst” List of edges that exist in the Minimum Spanning Tree and integer “k”

1. getMaxEdge() function:

parameters: “mst” List of edges that exist in the Minimum Spanning Tree

body: Firstly, “ind” & “max” variables are set to 0. Then, a for loop is entered to pass on each element in “mst” list. If the weight of the current element is greater than the max, the max is set to this weight and “ind” is set to the index of the current element. After finishing the loop, “ind” is returned.

Function’s Order: O (for loop) + O (the rest of function)

Let N=mst.Count

Order of for loop: # times \* loop body= N \* 1 =

Order of the rest=

Function’s Order=

1. removeEdge() function:

parameters: Edge “e”

body: Firstly, an object of “Edge” Class is instantiated named “e2”, so it carries the following attributes; source, destination and weight. The “src” of “e2” is set to the “src” of “e” and the same applies to “dst”. Then, the Weight of “e2” is set to -1 because setting it to a negative value means removing it because “MST” Algorithm isn’t applied on negative edges. Finally, “e2” object is returned.

Function’s Order:

1. getClusters() function:

parameters: “vertices” list of integers and “mst” list of edges

body:

1. Dfs() function:

parameters: “vertex” integer

body:

1. getCenteroid() function:

**ImageClass**

1. Edge Struct

Attributes: - “src” integer //source node

- “dst” integer //destination node

- “Weight” float //Edge’s weight

1. **Vertex inner class**

Attributes: - “vert” integer - “parent” integer

Methods: getters and setters for both attributes

1. getDistinctColors() function:
2. buildingMST() function:
3. getMSTsum() function:
4. makeClister() function:
5. getK () function:
6. standardDeviation() function:
7. getMean() function: