# README: COL106 Assignment 6

Zuhaib Ul Zamann

November 2019

## 1 Introduction

In this assignment I implement Graph Data Structure for the case of triangulation of surfaces so as to be able to answer some queries on the graph. The assignment requires us to use the data of triangulation to return the results of various queries like neighbours of a triangle, Maximum diameter, Centroid of component, centroid,Point neighbours, triangle neighbours, Incident triangles etc;

## 2 Methods: Implementation and Time complexities

The main implementation that was the core of this assignment was that of shape. In shape I maintained fields of Points(A Red Black Tree which stores all the Points created), Triangles(A Red Black Tree of Triangles),Edges(A Red Black Tree of Edges),D(A dynamic Array of triangeles),DS( A disjoint set which implements quick union using rank heuristic for keeping a track of number of components in the graph) ,BoundaryEdges( A Dynamic Array storing Boundary edges) and two integers Num1Edges and NumG2Edges respectively keeping track of boundary edges and improper edges.

1. **ADD_TRIANGLE**:
   In this method I created a triangle and checked if it is a valid triangle. If it is then I do the necessary pre-computations else I return false. In pre-computation I maintain the Adjacency List and also the various datastructures.

2. **BOUNDARY_EDGES**:
   In this method I return the boundary edges in an array if any else I return null. This is accomplished using BoudaryEdges DynamicArray;

3. **COUNT_CONNECTED_COMPONENTS**:
   In this i return the number of components present in the Graph. This is an **O(1) retreival from Disjoint set Data Structure**

4. **IS_CONNECTED**:
   This is close to **O(1:: log*n) retrival form DS disjoint set as I used path compression and Rank Heuristic to create the Data Structure**;

5. **MAXIMUM_DIAMETER**:
   This I solved by Applying BFS on each and every triangle and keeping the track of number of triangles visited along with the farthest triangle from it. I then return the Farthest distance Possible between two triangle in a component with maximum Triangles . If two components have same number of triangles then i return the one with the maximum (MAximum diameter). The whole process is acheived with the help of a Data Structure Tuple::: O(n*(n+m))

6. **CENTROID**:
   The query is answered by first finding all the point in each Component and storing in a RedBlackTree of the corresponding component. The RedBlack trees are then converted in DynamicArrays and the minimum distance between two points in Different DynamicArrays is return. O(n*n+n(n+m))

7. **CENTROID_OF_A_COMPONENT:**
   The whole process is similar to that of the CENTROID query.O(n*n+n(n+m))

8. **NEIGHBOURS_OF_A_TRIANGLE**:
   These queries are simply answered by the underlying DataStructures in each triangle and the implementation is self understood. O(n)

9. **BinaryMaxHeap**:
   Although the name will suggest that it is a max heap the actual implementation is that of a min heap. I just intially created a maxHeap but then felt the need of a min Heap **so essentialy changed a few lines of code to create a minHEap** although the name remained as that of earlier given(JAVA DOES NOT ALLOW ME CHANGE FILESNAME EASILY WITHOUT AFFECTING THE NAME INSIDE)

10. **RBTRee**:
    This implementation is same for Assignment 4,5 and 6

11. **TYPE_MESH**:
    The query is answered by the maintained field variables (Num1Edges and NumG2Edges) and the implementation is self understood.

12. **OneSourceMaxima**:
    This is just BFS

13. **Zuhaib**:
    This is helper function for centroid query essentialiy helping in finding mean. This used another helper function ADD
    All Other implementations are basic and are self understood.
    **Note: The pseudo code for the Disjoint set union was known to me because i did a course on Graphs and Data Structures on Coursera where it was taught by Professor. Alexander Kukkilov of University of San Diego.**