PROJECT BASED LEARNING ASSIGNMENT

Course : Advanced Data Structures

Contributor Roll Number : 50 & 51

Problem Definition :

A graph consisting of edges and vertices is given as input. Implement randomized graph contraction algorithm and generate new graph of 25% size. The approach should allow expansion of graph as per the user input to size 50%, 75% and 100%. The suitable data structure can be used during contraction and expansion process.

Problem description :

A contraction graph is basically an application of the clustering concept. **Contraction** is an operation which removes an edge or vertex from a graph while simultaneously merging the two vertices that it previously joined. In a graph or any group of connected 'n' nodes, a graph can store intermediate results during contraction in a cluster wherein multiple nodes can get grouped to form an other node. Whilst during expansion, such same node which were clustered gets the opportunity to get de-clustered and thus, the nodes regain their original positions.

This graph can be implemented using the concept of logical grouping or can also be grouped with the help of randomization. The basic motive behind the given problem statement is to implement a graph contraction algorithm with the help of randomization. Suppose we have 20 nodes, so while we perform clustering at 50% factor, lesser nodes will be shown because they are clustered. In the same way, during expansion, more nodes become visible. The working of clustering and grouping of nodes depend on the zoom level.

Technology stack :

1. Google Maps API for using Map Preview within android application
2. Java JDK+ JRE Environment Version 1.8 for backend development
3. XML for UI development
4. Google Utilities

Functions used :

1. private void setUpClusterer(GoogleMap mMap) :

Parameters :

The setUpClusterer method takes the instance of GoogleMap fragment as a parameter and uses the same for performing operations on the mapFragment.

Within this method, the object of clusterManager class gets initialized with the context of the activity and the map instance. The cluster Manager stores the map items and groups them into a cluster.

Each item which is grouped into a cluster stores the latitude-longitude position points, title string and a snippet string. The same is used as a parameter for the clusterManager.

2. public void startClustering(View view)

Parameters :

This method takes in the instance of the currently active view as a parameter in order.

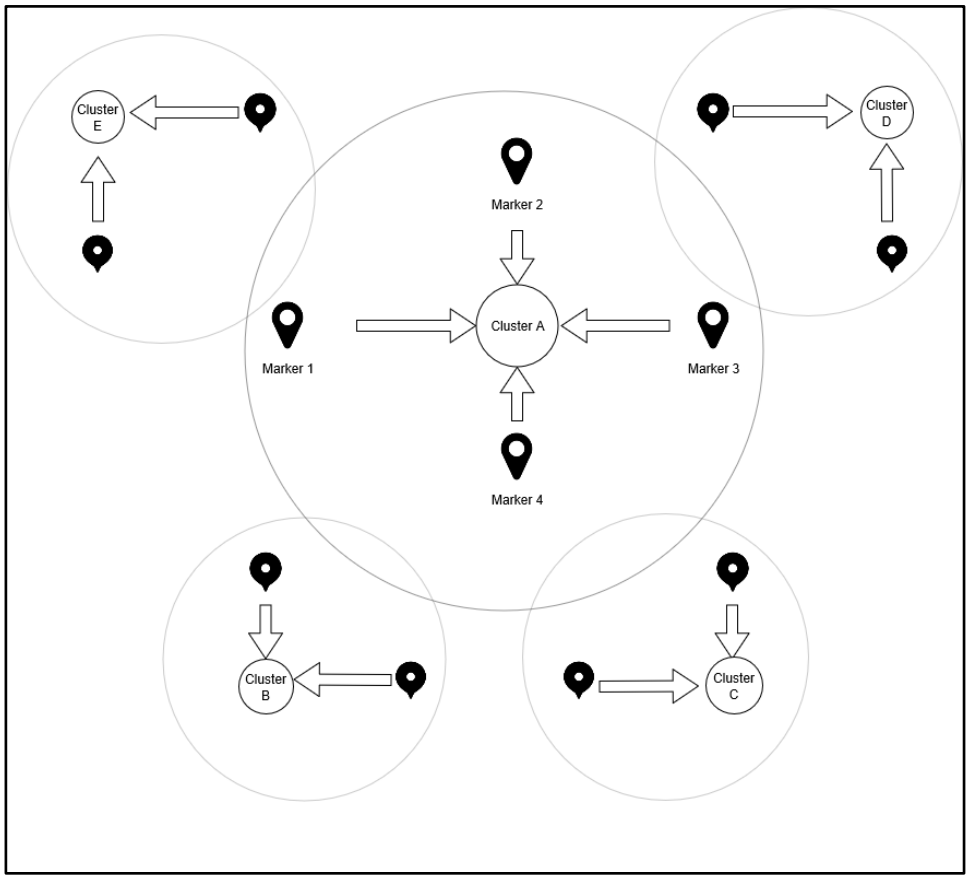
In this method, all the desired latitude longitude points are added into the list which is passed to the cluster manager object. The cluster manager groups all those points and makes a cluster of the same which gets expanded or contracted based on the zoom level.

Key Functionality :

Our application performs clustering in a unique way of its own. Suppose there are 10-12 items or markers mapped in the graph, the clustering takes place on the basis of consideration of the markers present in the vicinity. And our algorithm considers a marker which can take the form of a cluster or vice versa, to and fro. This too is done with the help of a design pattern “Composite Design Pattern” which allows to make the marker change its forms dynamically at the runtime. The more markers are mapped, the more is the dynamicity.

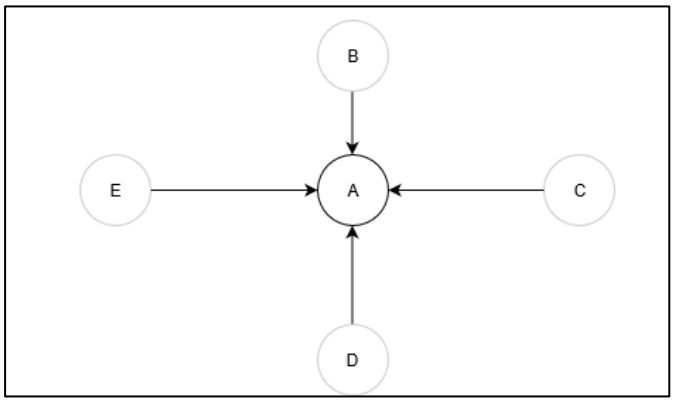
Wireframe Diagram :

Marker based Clustering :



In the diagram above, clusters A, B, C, D and E are formed on the basis of the markers which are in their proximity range. The circle represents their respective ranges.

Cluster based Grouping :



Markers B, C, D and E are further grouped in the cluster A.

All this contraction and expansion takes place on the basis of zoom levels and the selected cluster item and their proximity range.