Assignment 6

Name: Radhika
Enrollment Number: 18114060
Batch: O3

Problem Statement -1:

Write a menu driven C++ program to implement a graph using adjacency list (linked list) without using STL. Perform following operations on the graph.

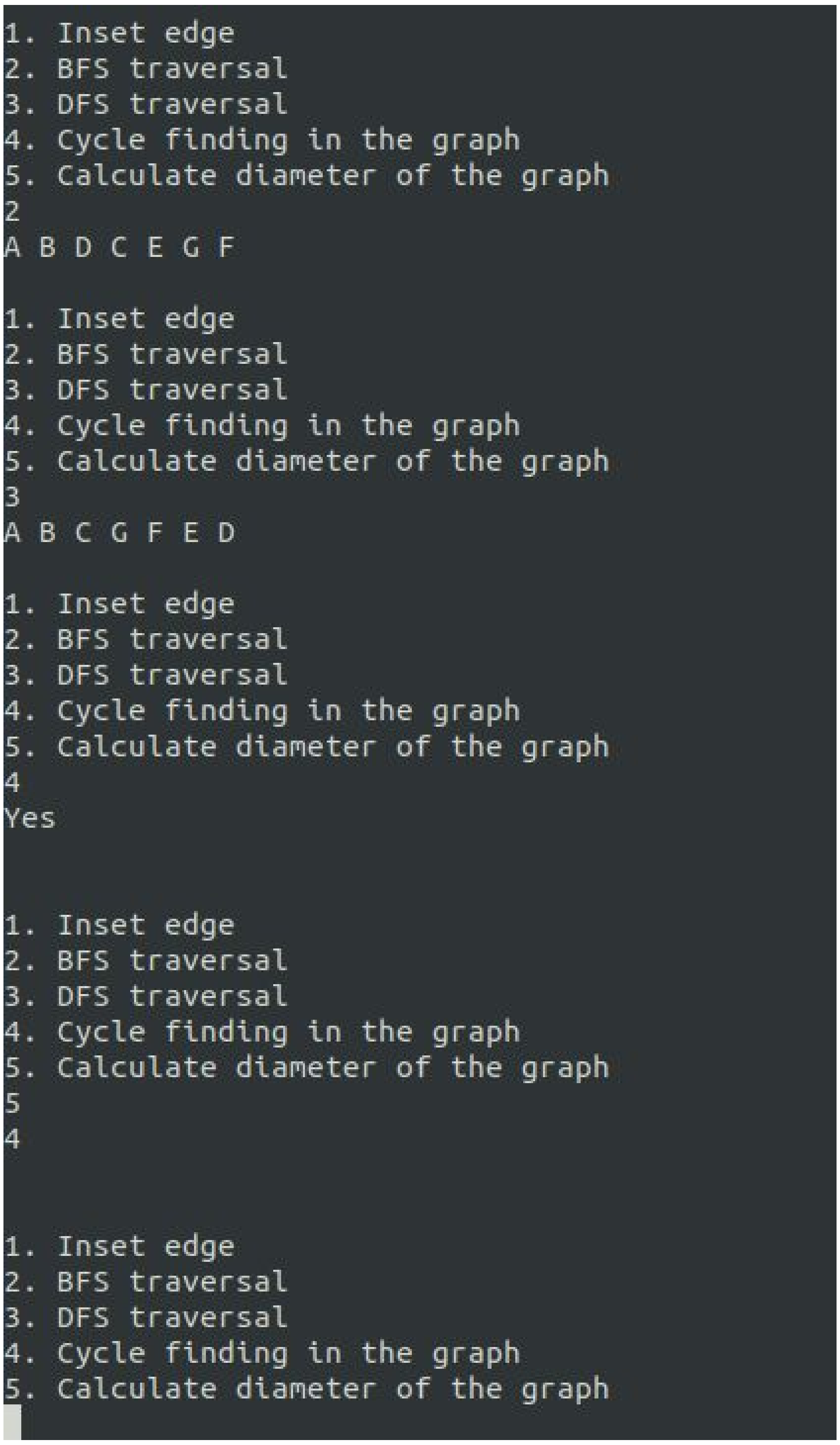
1. BFS traversal
2. DFS traversal
3. Cycle finding in the graph
4. Calculate diameter of the graph

Data Structure:

1. Adjacent List build of Linked List for every node.
2. Colour Array for maintaining status visited or not.
3. Present array for maintaining node is present or not.
4. Bool isCyclePresent for cycle and diameter int for diameter of graph.

Algorithm:

1. Initially present is 0 and Adjacent List is empty for all node.
2. For Insert Edge input both char and create node for both linked List and insert in respectively List.
3. In BFS Traversal
    a. colour of all nodes is zero.
    b. Iterate on all nodes and check colour and present or not.
    c. If true then push that node into empty queue.
    d. Till queue in not empty check all its neighbour and check colour and insert into queue according to condition satisfied.
4. In DFS traversal
    a. colour of all nodes is zero.
    b. Iterate on all nodes and check colour and present or not.
    c. If true then call DFS.
    d. Change colour to 1 and call recursively DFS to all its colour 0 child.
5. In cycle Detection
    a. colour of all nodes is zero.
    b. Iterate on all nodes and check colour and present or not.
    c. If true then call findCycle.
    d. Change colour to 1 and call recursively DFS to all its colour 0 child.
    e. If any neighbour find colour 2, means that node is connected to previous visited node, cycle is present.
6. In Diameter
    a. Check Number of SCC present, if greater than 1 then Diameter is Infinite.
    b. Calculate greatest distance node from all node, and take maximum of all distance, that is diameter.



```
1. Inset edge
2. BFS traversal
3. DFS traversal
4. Cycle finding in the graph
5. Calculate diameter of the graph
2
A B D C E G F

1. Inset edge
2. BFS traversal
3. DFS traversal
4. Cycle finding in the graph
5. Calculate diameter of the graph
3
A B C G F E D

1. Inset edge
2. BFS traversal
3. DFS traversal
4. Cycle finding in the graph
5. Calculate diameter of the graph
4
Yes


1. Inset edge
2. BFS traversal
3. DFS traversal
4. Cycle finding in the graph
5. Calculate diameter of the graph
5
4


1. Inset edge
2. BFS traversal
3. DFS traversal
4. Cycle finding in the graph
5. Calculate diameter of the graph
```

Problem Statement -2:

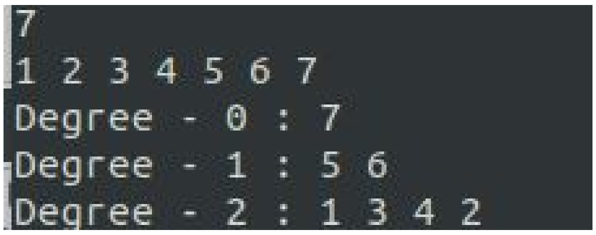
Write a C++ program to implement a binomial heap using heap data structures (without using STL). Print the order of each binomial heap and use Graphviz to show the forest of binomial heap.

Data Structure:

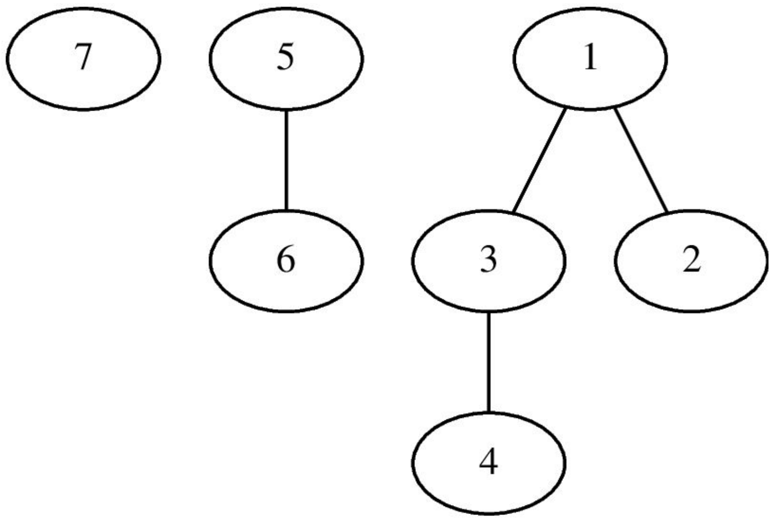
1. Struct node has int key,int degree,node *leftmost for child, node* next for siblings.
2. Degree saves the height of that heap.

Algorithm:

1. While
    a. inserting create one node.
    b. Merge with head node.
    c. Union head node.
2. In Merge new node into head node in increasing order of degree.
3. In Union max 3 same degree heap possible.
    a. If only one unique degree heap possible present, ignore it.
    b. If two heap present of a unique degree, then merge both into degree+1.
    c. If three heap present of a unique degree, then merge last 2 heap into degree+1, and insert that position.
4. Then show heap by call showHeap of next head node only.All head node call childShow function and recursively call all nodes.



```
7
1 2 3 4 5 6 7
Degree - 0 : 7
Degree - 1 : 5 6
Degree - 2 : 1 3 4 2
```



Problem Statement -3:

Write a C++ program to implement Bentley-Ottmann Algorithm to find and print all the intersection points of n given lines. Use of STL is allowed. The specific type of data structure that must be used include Priority Queue and BST. Using least square method find the linear fit of the M found intersection points and print the line in the form ax+b. The student should demonstrate this on a GUI using QT library. The input should be given in following format:

1. Input number of line segments, N
2. N lines where 2N points are provided, i.e., 2 points in each line

Data structure :

1. Store line in lineSet pair of two pair which contain x and y coordinates of end points.
2. sortByX map x coordinates of end points and intersection points with type and line Number.
3. sortByY maintains active line segments.
4. Intersection point stores all intersection points for Linear Fit.

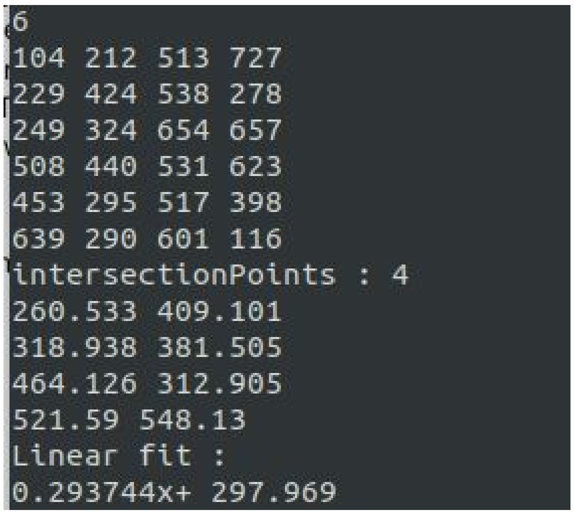
Algorithm:

1. Let there be n given lines. There must be 2n end points to represent the n lines. Sort all points according to x coordinates. While sorting maintain a type to indicate whether this point is left point of its line or right point.
2. Start from the leftmost point. Do following for every point
    a. If the current point is a left point of its line segment, check for intersection of its line segment with the segments just above and below it.
    b. If the current point is a right point, remove its line segment from active list and check whether its two active neighbors (points just above and below) intersect with each other.
    c. If intersection point then swap both lines of intersection point.
3. For linear Fit by all intersection point use

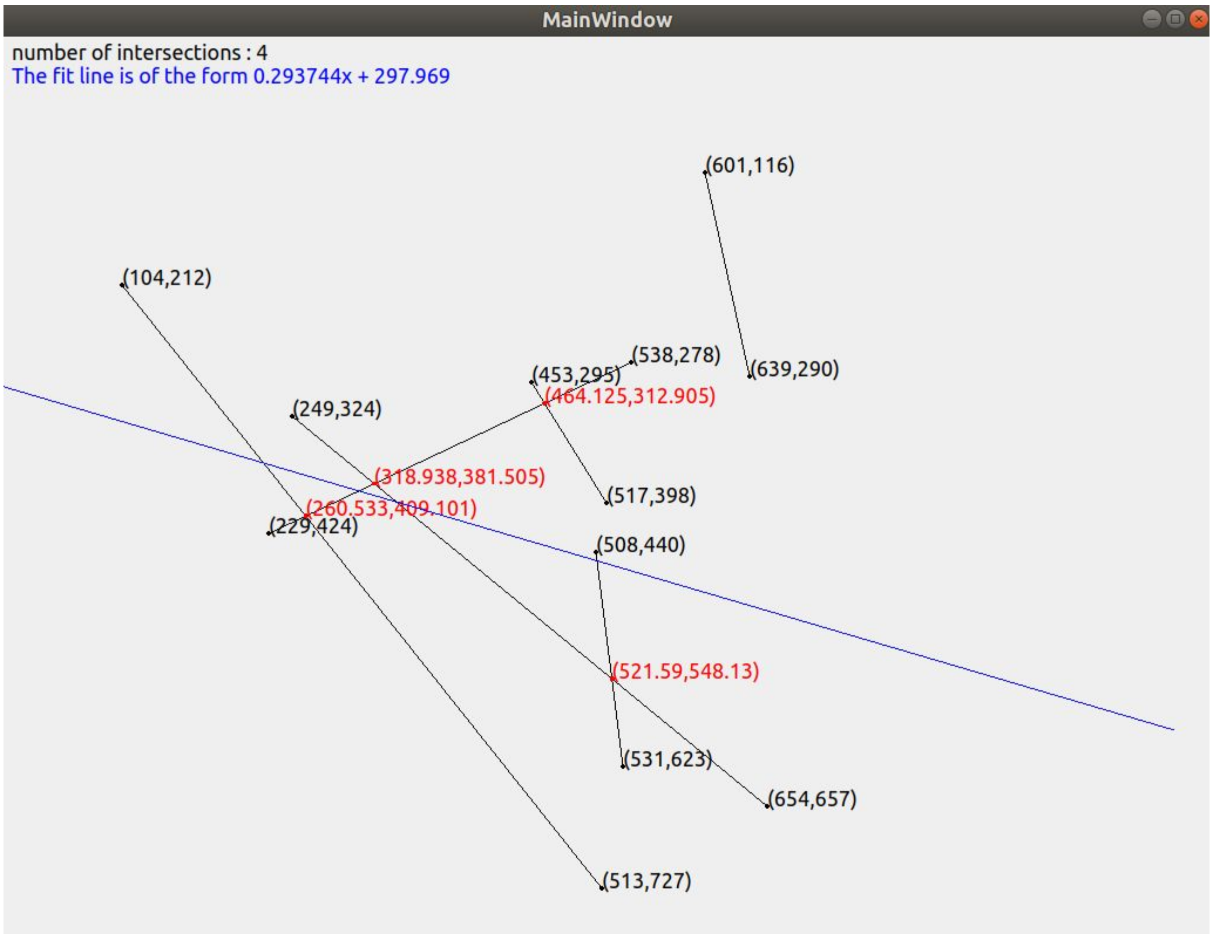
$$a = \frac{(\Sigma y)(\Sigma x^2) - (\Sigma x)(\Sigma x y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

$$b = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{n(\Sigma x^2) - (\Sigma x)^2}$$

Linear fit equation : y=ax+b.



```
6
104 212 513 727
229 424 538 278
249 324 654 657
508 440 531 623
453 295 517 398
639 290 601 116
intersectionPoints : 4
260.533 409.101
318.938 381.505
464.126 312.905
521.59 548.13
Linear fit :
0.293744x+ 297.969
```



MainWindow

number of intersections : 4
The fit line is of the form 0.293744x + 297.969