Assignment 6

Submitter's Details :

Name: Karanpreet Singh

Enrollment Number : 18114044

Class : B.tech , CSE 2nd year

Email: ksingh2@cs.iitr.ac.in

GithubLink : https://github.com/karan0299/CSN-261-ASSIGNMENT

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

DataStructures :

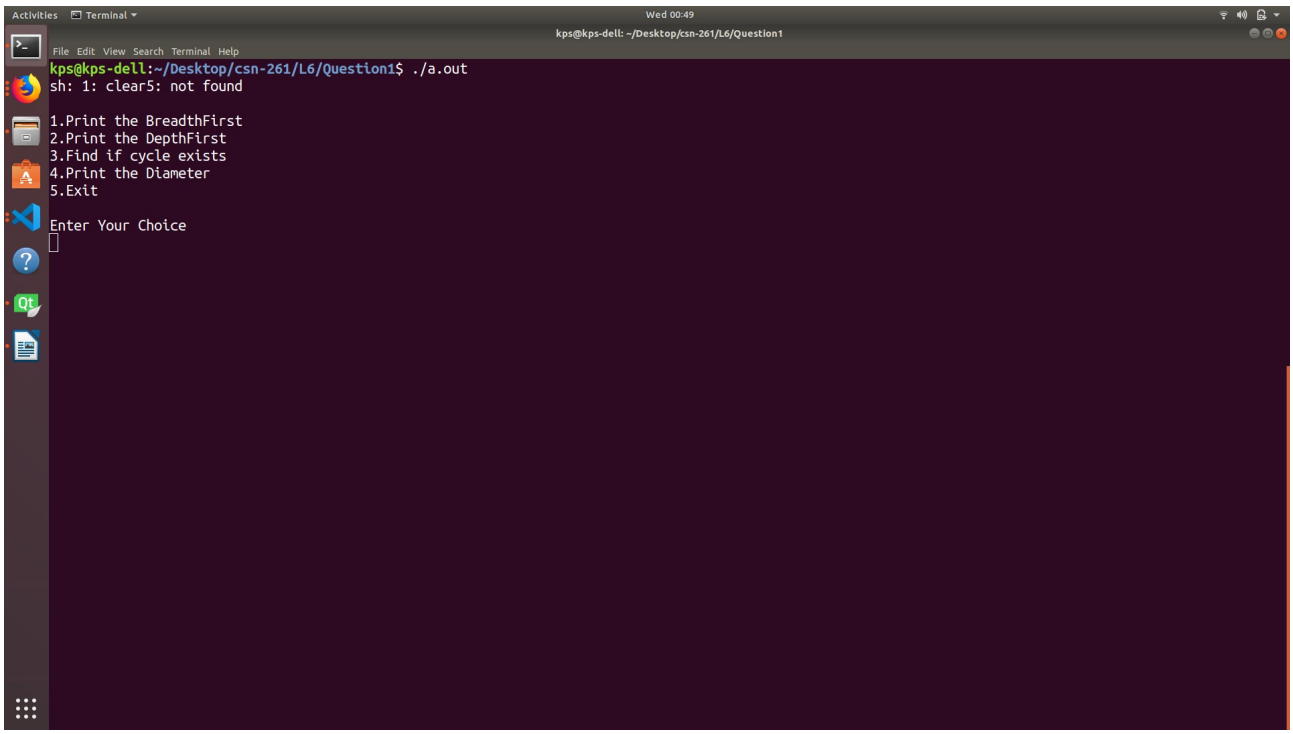
i) Linked List
ii) Vector

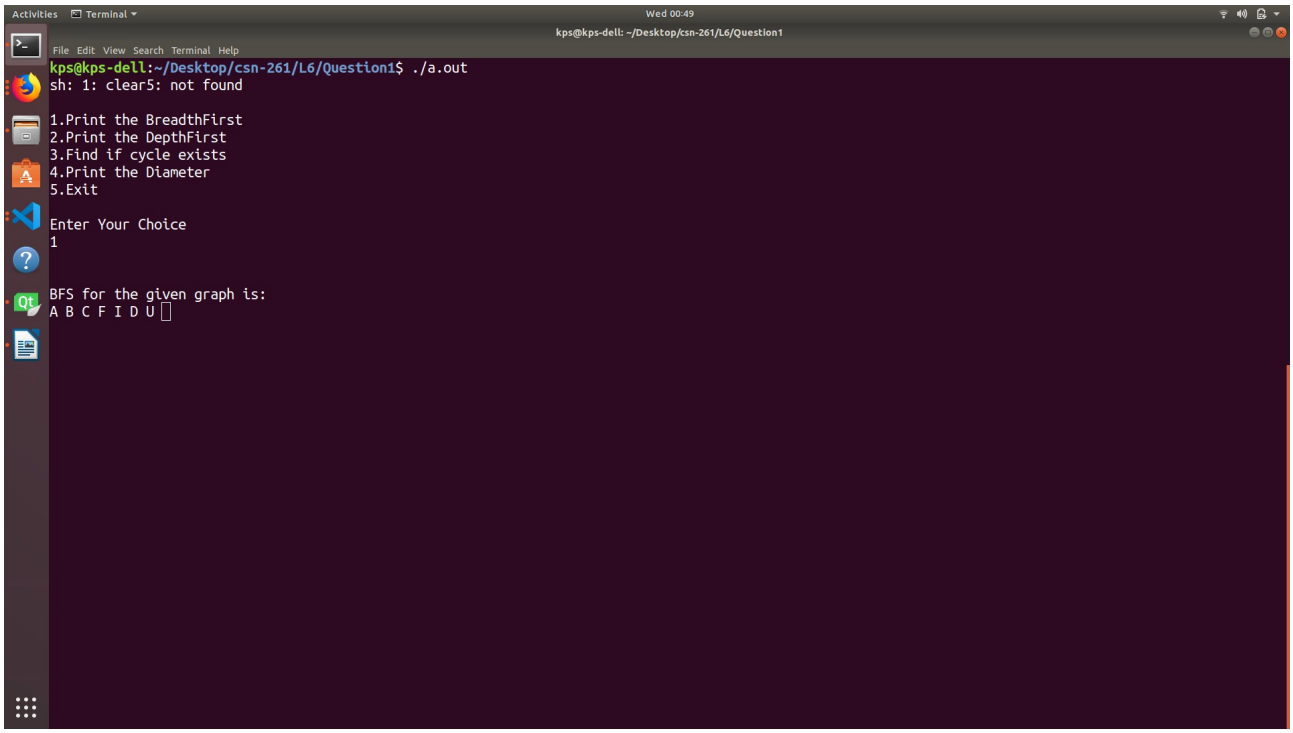
Algorithm:

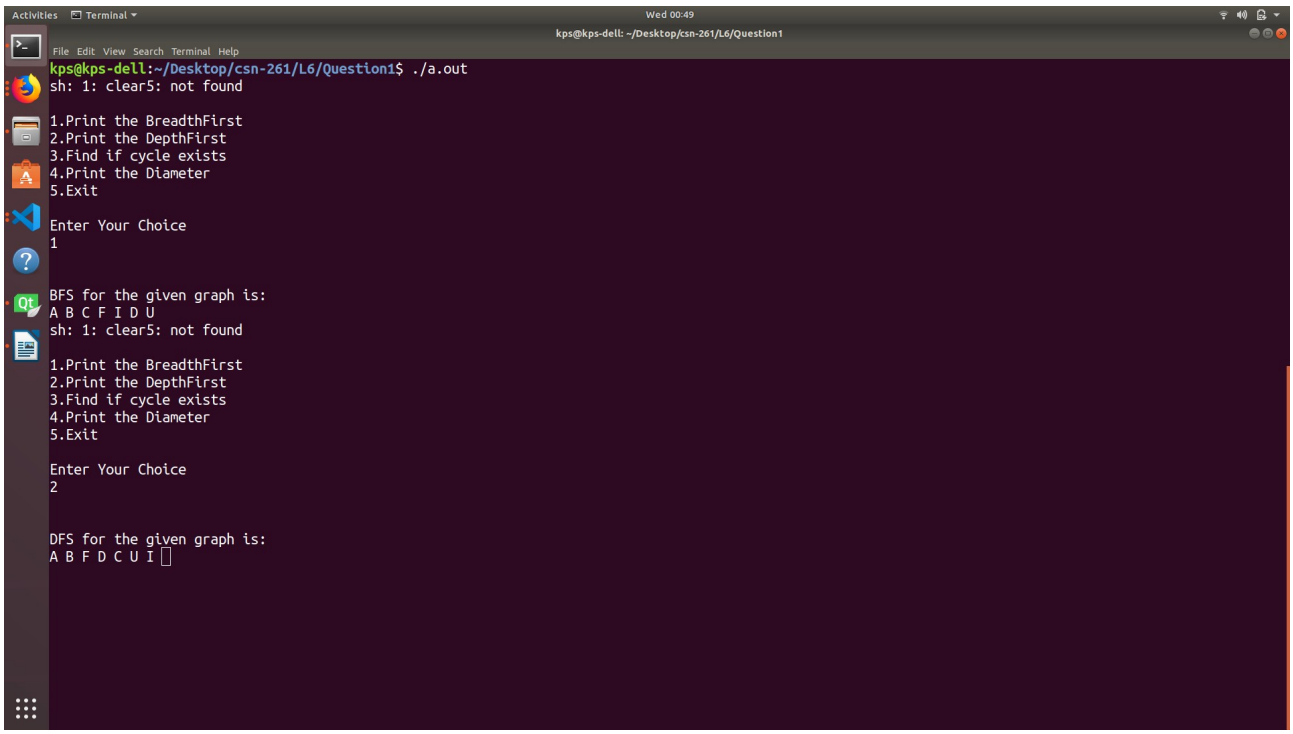
i)BreadthFirstSearch Traversal :
BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layerwise thus exploring the neighbour nodes.

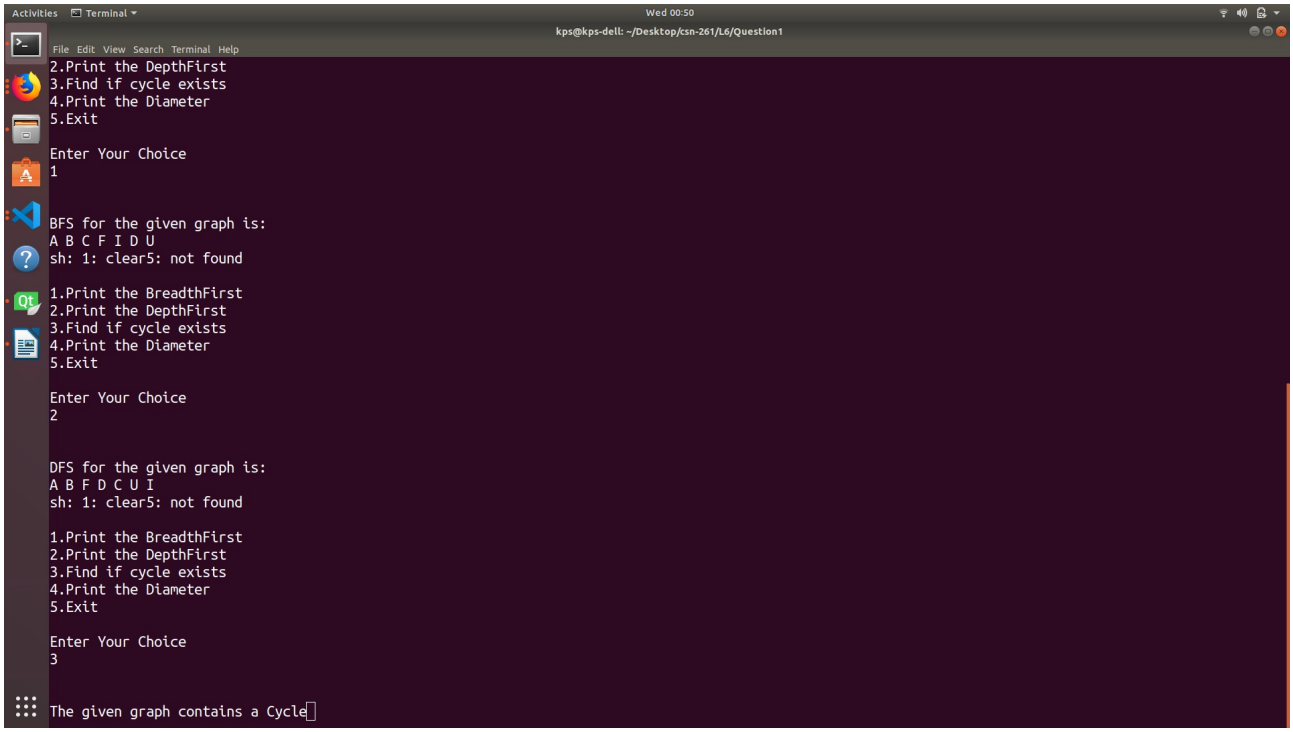
ii)DepthFirstSearch Traversal:Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

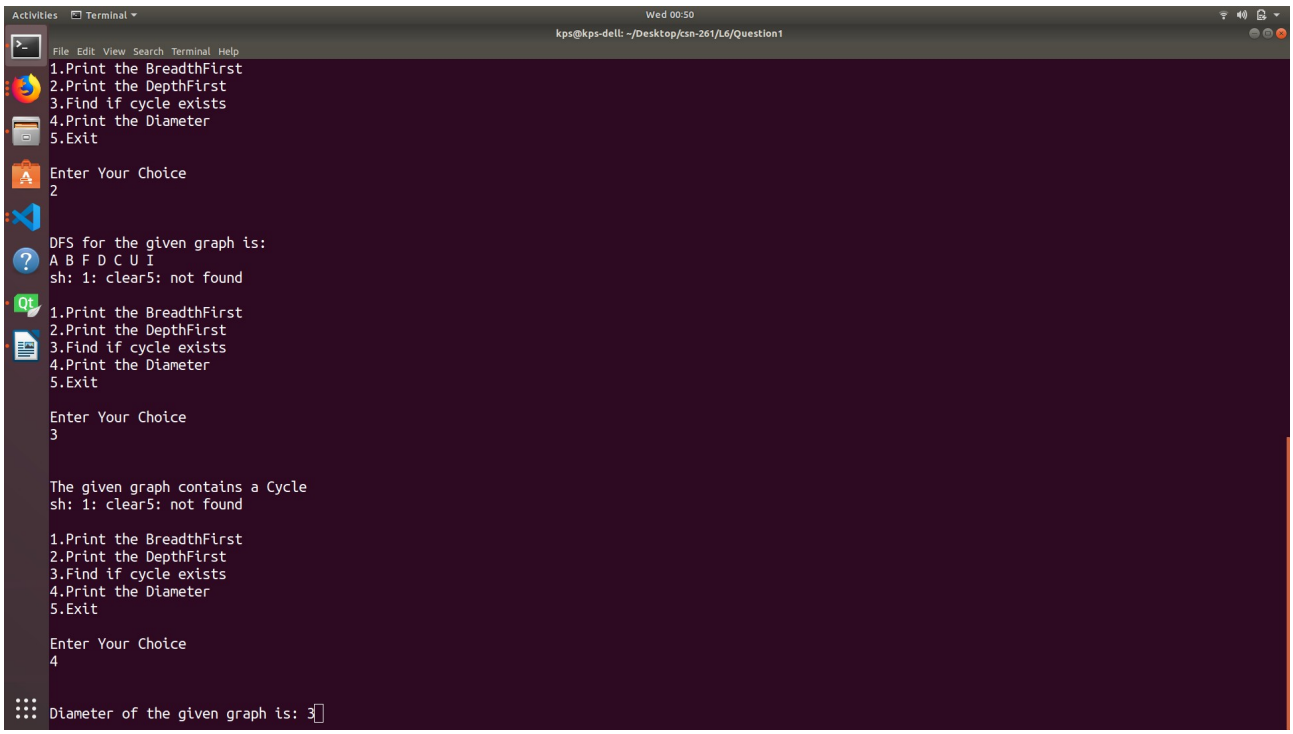
ScreenShots:


Activities
Terminal
Wed 00:49
kps@kps-dell: ~/Desktop/csn-261/L6/Question1
File Edit View Search Terminal Help
kps@kps-dell:~/Desktop/csn-261/L6/Question1$ ./a.out
sh: 1: clear5: not found
1.Print the BreadthFirst
2.Print the DepthFirst
3.Find if cycle exists
4.Print the Diameter
5.Exit
Enter Your Choice


Activities
Terminal
Wed 00:49
kps@kps-dell: ~/Desktop/csn-261/L6/Question1
File Edit View Search Terminal Help
kps@kps-dell:~/Desktop/csn-261/L6/Question1$ ./a.out
sh: 1: clear5: not found
1.Print the BreadthFirst
2.Print the DepthFirst
3.Find if cycle exists
4.Print the Diameter
5.Exit
Enter Your Choice
1
BFS for the given graph is:
A B C F I D U


Activities
Terminal
Wed 00:49
kps@kps-dell: ~/Desktop/csn-261/L6/Question1
File Edit View Search Terminal Help
kps@kps-dell:~/Desktop/csn-261/L6/Question1$ ./a.out
sh: 1: clear5: not found
1.Print the BreadthFirst
2.Print the DepthFirst
3.Find if cycle exists
4.Print the Diameter
5.Exit
Enter Your Choice
1
BFS for the given graph is:
A B C F I D U
sh: 1: clear5: not found
1.Print the BreadthFirst
2.Print the DepthFirst
3.Find if cycle exists
4.Print the Diameter
5.Exit
Enter Your Choice
2
DFS for the given graph is:
A B F D C U I


Activities
Terminal
Wed 00:50
kps@kps-dell: ~/Desktop/csn-261/L6/Question1
File Edit View Search Terminal Help
2.Print the DepthFirst
3.Find if cycle exists
4.Print the Diameter
5.Exit
Enter Your Choice
1
BFS for the given graph is:
A B C F I D U
sh: 1: clear5: not found
1.Print the BreadthFirst
2.Print the DepthFirst
3.Find if cycle exists
4.Print the Diameter
5.Exit
Enter Your Choice
2
DFS for the given graph is:
A B F D C U I
sh: 1: clear5: not found
1.Print the BreadthFirst
2.Print the DepthFirst
3.Find if cycle exists
4.Print the Diameter
5.Exit
Enter Your Choice
3
The given graph contains a Cycle



Question 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.

DataStructures :

i) *Vector*

ii) *Binomial Heap:*

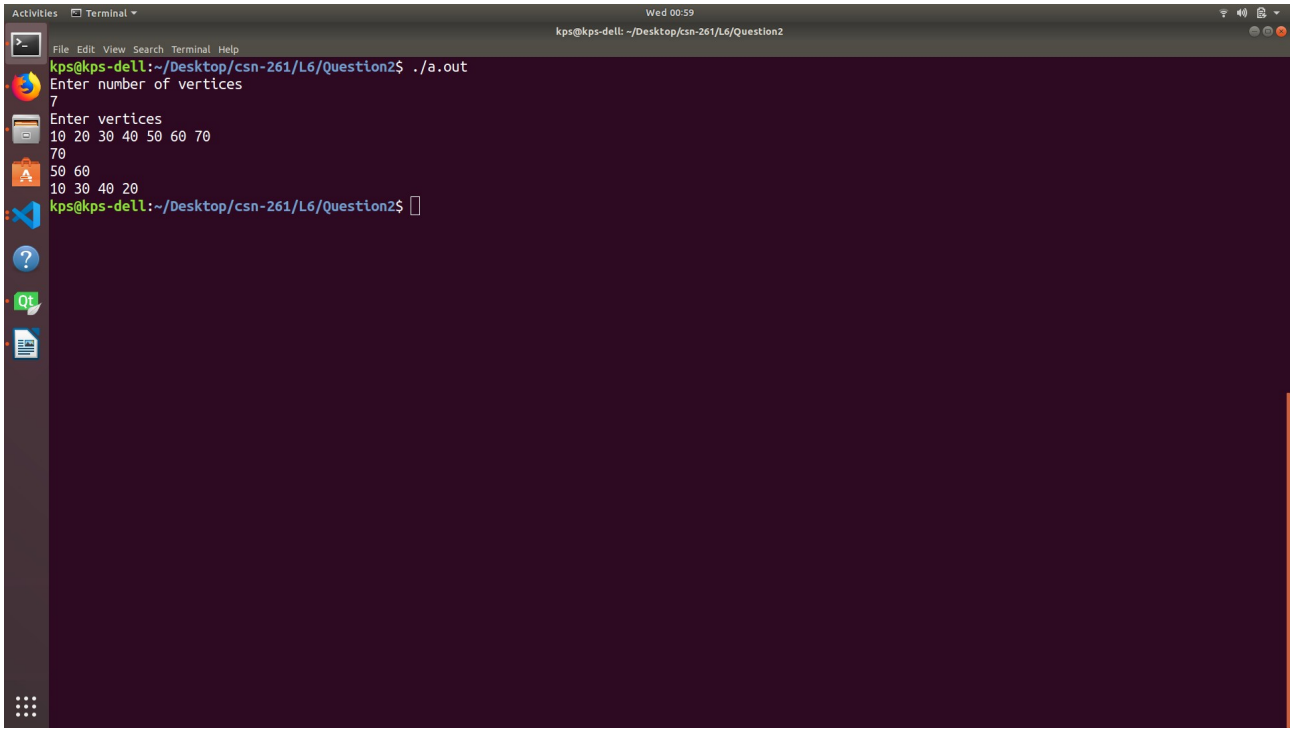
A binomial heap is implemented as a set of **binomial trees** (compare with a binary heap, which has a shape of a single binary tree), which are defined recursively as follows:

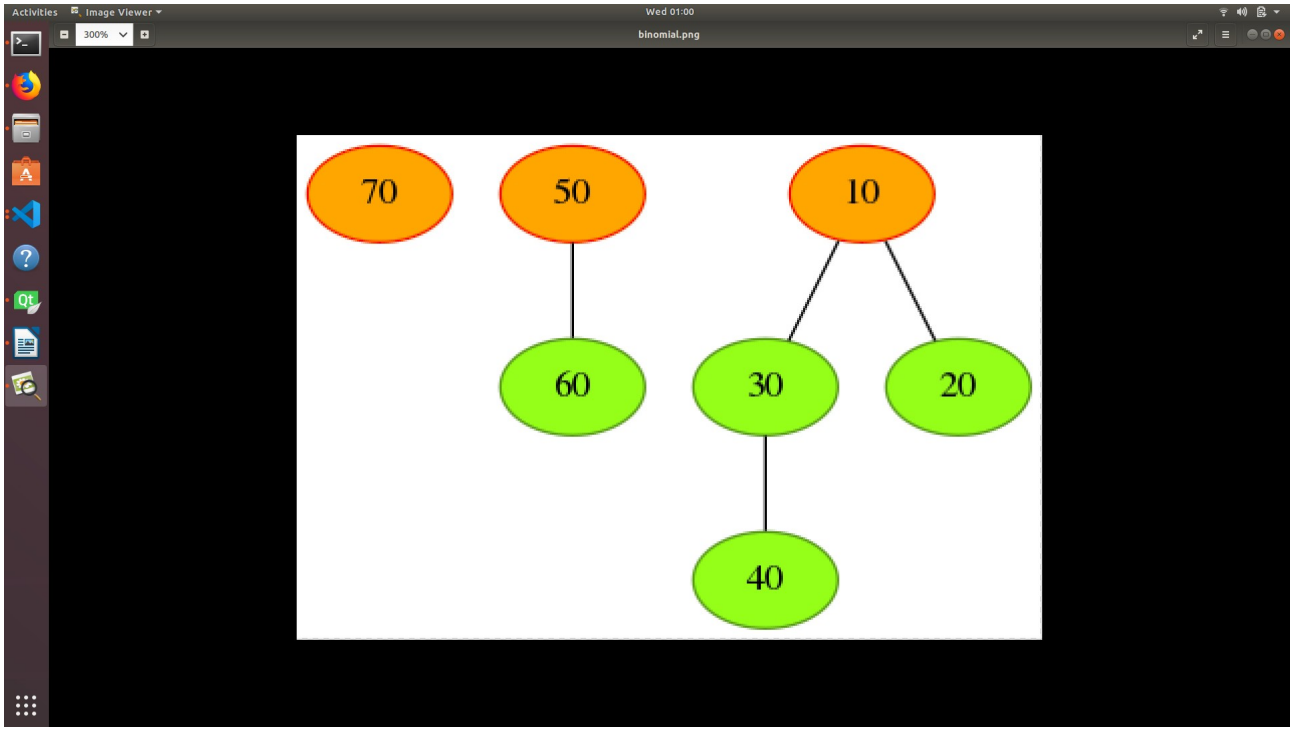
- A binomial tree of order 0 is a single node
- A binomial tree of order k has a root node whose children are roots of binomial trees of orders $k-1$, $k-2$, ..., 2, 1, 0 (in this order).

Algorithm: Binomial Heap

Thebinomial treeBkis an ordered tree defined recursively. the binomial treeB0consists of a single node. Thebinomial treeBkconsists of two binomial treesBk−1that arelinkedtogether: theroot of one is the leftmost child of the root of the other.

ScreenShots:





Question 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:

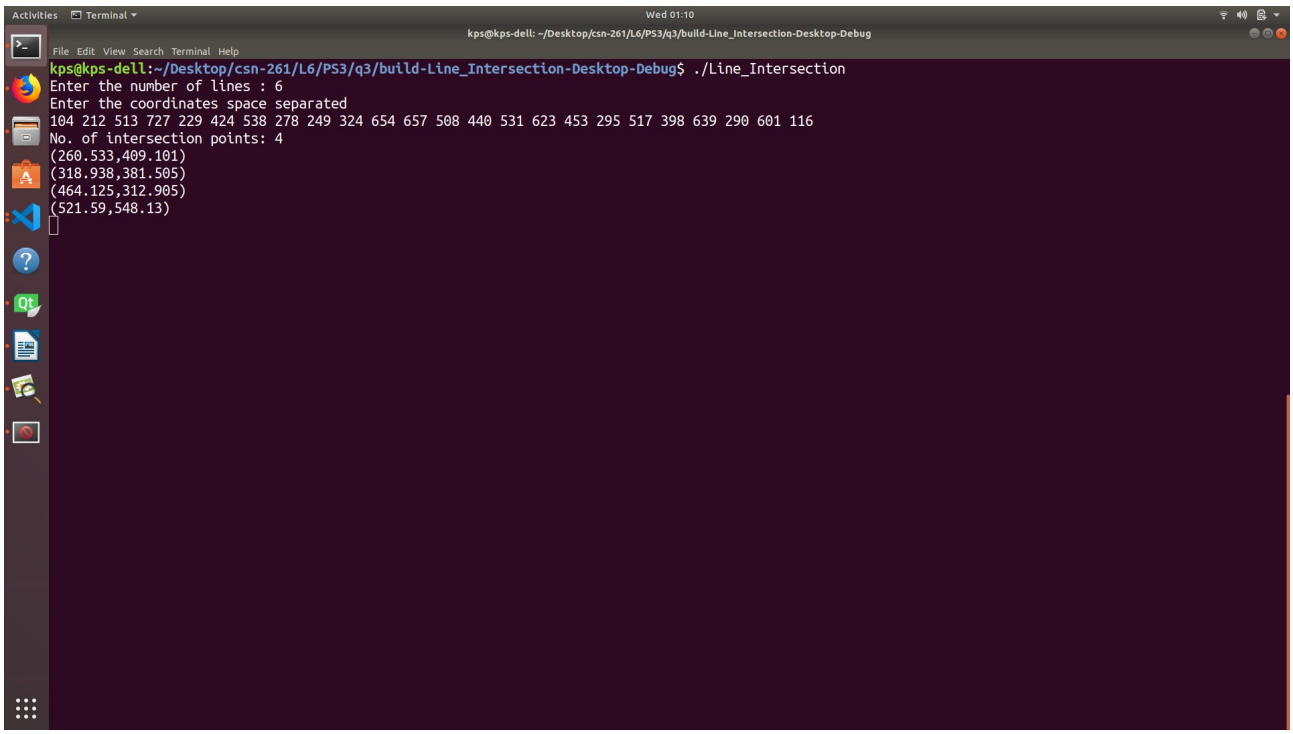
DataStructures :

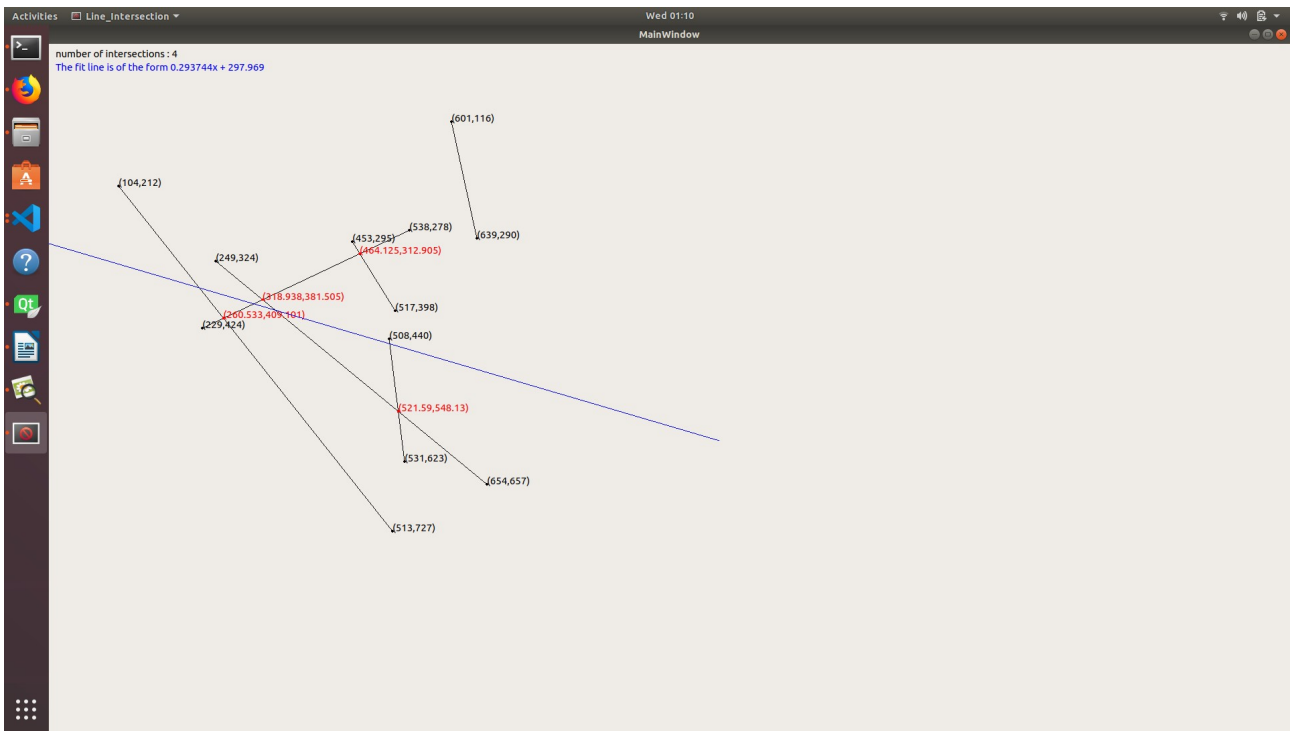
i) *Vector*
ii) *Set*
iii) Queque
iv) Array

Algorithm:

i)*Bentley-ottman:*
In computational geometry, the **Bentley–Ottmann algorithm** is a sweep line algorithm for listing all crossings in a set of line segments, i.e. it finds the intersection points (or, simply, intersections) of line segments. It extends the Shamos–Hoey algorithm,a similar previous algorithm for testing whether or not a set of line segments has any crossings. For an input consisting of line segments with crossings (or intersections), the Bentley–Ottmann algorithm takes time
ii)*Least Sqaure Method* :
The method of **least squares** is a standard approach in regression analysis to approximate the solution of overdetermined systems, i.e., sets of equations in which there are more equations than unknowns. "Least squares" means that the overall solution minimizes the sum of the squares of the residuals made in the results of every single equation

ScreenShots:


Activities
Terminal
Wed 01:10
kps@kps-dell: ~/Desktop/csn-261/L6/PS3/q3/build-Line_Intersection-Desktop-Debug
File Edit View Search Terminal Help
kps@kps-dell:~/Desktop/csn-261/L6/PS3/q3/build-Line_Intersection-Desktop-Debug$ ./Line_Intersection
Enter the number of lines : 6
Enter the coordinates space separated
104 212 513 727 229 424 538 278 249 324 654 657 508 440 531 623 453 295 517 398 639 290 601 116
No. of intersection points: 4
(260.533,409.101)
(318.938,381.505)
(464.125,312.905)
(521.59,548.13)


Activities
Line_Intersection
Wed 01:10
MainWindow
number of intersections : 4
The fit line is of the form 0.293744x + 297.969
(601,116)
(104,212)
(538,278)
(453,295)
(639,290)
(464.125,312.905)
(249,324)
(318.938,381.505)
(260.533,409.101)
(517,398)
(229,424)
(508,440)
(521.59,548.13)
(531,623)
(654,657)
(513,727)