Project Report
Advaced Data Structrue (COP 5526)

Name: Lavanya Chennupati
UFID: 11299429
Email: lchennupati@ufl.edu

1.Compile and Run

Using gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1)
Compiler: JDK 1.7.0_75

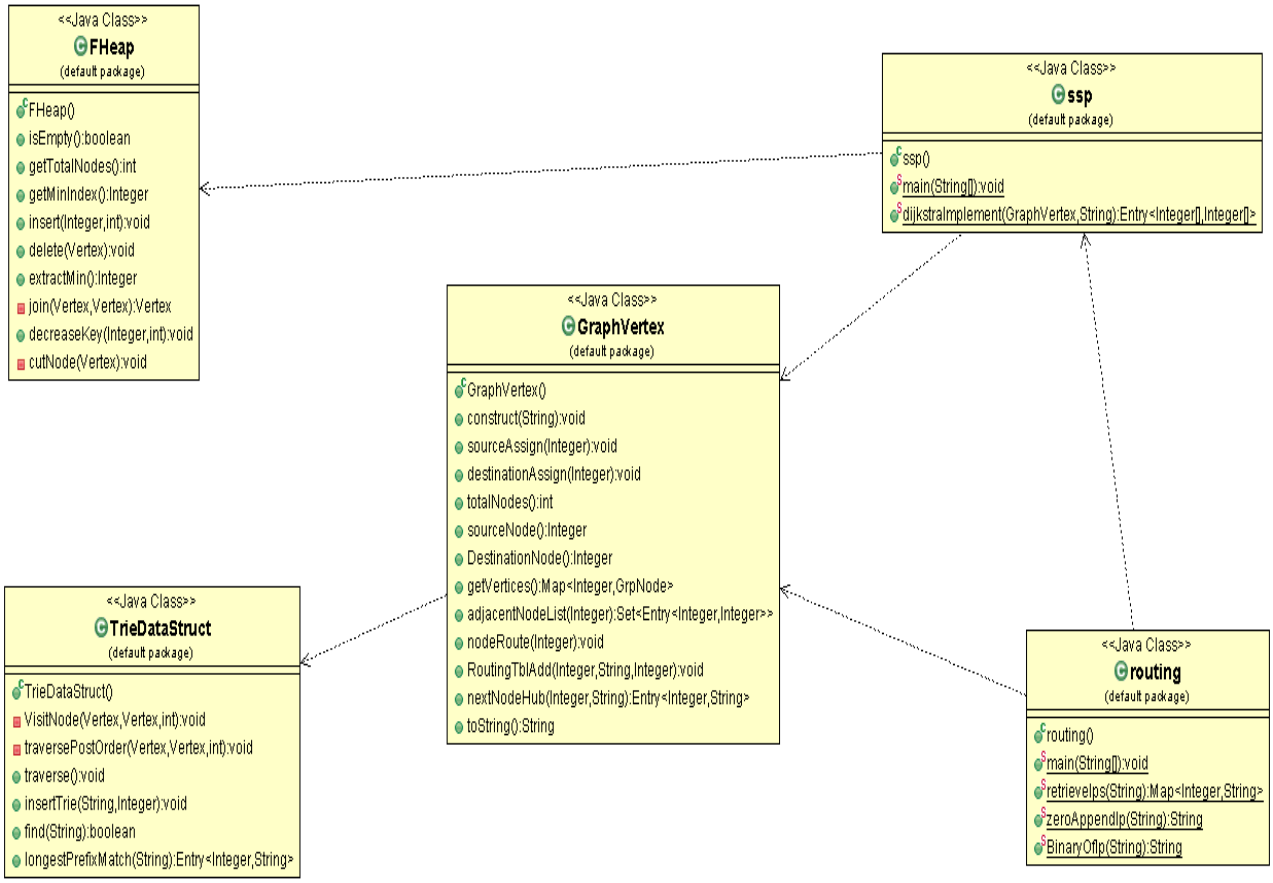
After unzip the Chennupati_Lavanya.zip, the folder will have 5 Java source file and a Makefile ssp.java, routing.java, GraphVertex.java, FHeap.java, TrieDataStruct.java and Makefile.

To compile the files in Unix environment, first move to the right directory, then use:
make ssp routing

After compiled, use following command to :
1) run the program for Part-1
java ssp <file-name> <source> <destination>

2)run the program for Part-2
java routing <file-1> <file-2> <source> <destination>

2. Structure Of The program



a) Class GraphVertex :

Private class GrpNode :
The GraphVertex class contains a private class named GrpNode which contains property variables of a graph node object like its label, its adjacency list etc. The adjacency list is represented using a Map. This private class contains set and get methods along with others which will be used in the second part of the project.

Methods :
construct () : A call to this method is made from the ssp class using the object of GraphVertex class. The file passed through the command line arguments is loaded and the data from it is read. The data in the first line gives number of vertices and number of edges and the next lines give the edges and their weights.

getVertices (): This method returns the number of vertices passed from the command line

traversingNode() : This method will be used in second part of project by getting data from Routing table

RoutingTblAdd () : Used to add new entries in the routing table

sourceNode (): returns the label of the source node

DestinationNode(): returns the destination node label

adjacentNodeList(): returns the adjacent nodes of the vertex passed to the method as parameter

destinationAssign (): this method sets the destination to the value passed in the command line arguments. If it is not passed properly an exception occurs

getVertices(): returns the vertices of the given graph

nextNodeHub(): this method will be used in the second part where we determine the next hop on the path from source to destination using the routing table

toString(): The toString method is overridden to obtain the output in a desired manner and print it

b) Class FHeap :

Private Class Vertex: This private class contain a parametrized constructor along with some get and set methods. These methods are getIndex(), setIndex(element), getValue().

Methods :
isEmpty (): checking if the Fibonacci Heap is empty. This methods returns true if the Heap is null.

insert(): This method adds all the elements to the Fibonacci heap

decreaseKey (): The decreaseKey operation of the Fibonacci Heap is performed here. This function changes the key value of a specific node to a new value given and based on the priority of the node and its parent we determine if a cutNode (cascadingCut) is to be performed or not.

CutNode() : Here the child cut operation is performed. A base check if the parent of the node is null is done. Intially the childCut value of the node is false. If the childCutValue of the parent is true then cutNode is performed recursively until it becomes false. If it is not true then it is made true. Here after the cut operation is performed for a node from the tree and it is reinserted to the root list. Changes to the previous and next are made accordingly.

extractMin (): This method removes the minimum node in the fibonacci heap ie the element with the minimum key. Its previous and next pointers are rearranged to retain the circular list after the removal. If the node that is removed had any children then they are added to the root list. Now, the roots which have the same degree are merging and the min points to the new minimum in the heap.

join (): This method merges the forest of trees that are created after the extractMinimum operation. The merging is done such that the roots of the same degree are only merged.

c) Class ssp:

Methods :
main() :The main method is in this class where we create an instance for the GraphVertex class. Using this instance we load the file that contains sample test data for the project.

dijkstraImplementation(): Here, the actual implementation of the Dijkstra Algorithm using Fibonacci Heap for the adjacency list representation of undirected graph is done. An instance for the FHeap class is created and operations on fibonacci are done.

d) Class routing:

Methods :
main() : Here the files are read from command line arguments. The first file is same as that of the file for part 1 while the second file contains the IP addresses of the nodes. We then create a routing table for all the nodes and then add entries.

retrieveIps () : The IP Address values are loaded from the file and stored in map

zeroAppendIp (): Append zeros such that the length of address is 32 bits .

BinaryOfIp () : Convert the IPAddress to binary format

e) Class TrieDataStruct :

Private Class Vertex: The private class Node contains the get and set methods for the nodes in the trie. The methods are getValue(), getKey(), getRight(), getLeft(), AssignRight and AssignLeft.

Private Class vertexElement : This private class extends the class Vertex. It contains the get methods to retrieve the key and values.

Private Class nodesInBranch: The get and set methods of the Vertex class are overridden here and given their own definition, since the next levelheight elements also have the same property

Methods :
insertTrie (): The node values are inserted into the trie based on the properties of the root node, parent and grandparent.

traverse() : The tree is traversed in post order defined in the method post

traversePostOrder() : The recursive definition of the post order traversal is given here where first left sub tree, then right and finally root is visited.

longestPrefixMatch() : The longest prefix which is same for the destinations are grouped together in the trie. The longest prefix string is returned.