COP 5536: SPRING 2015
PROGRAMMING PROJECT
FINAL REPORT

SUBMITTED BY:

SRAVYA PASUPULETI

UFID: 21229828

PART 1

In first part we were required to implement Dijkstra’s Single Source Shortest Path (ssp) algorithm for undirected graphs using Fibonacci heaps. Adjacency list representation for graphs was to be used.

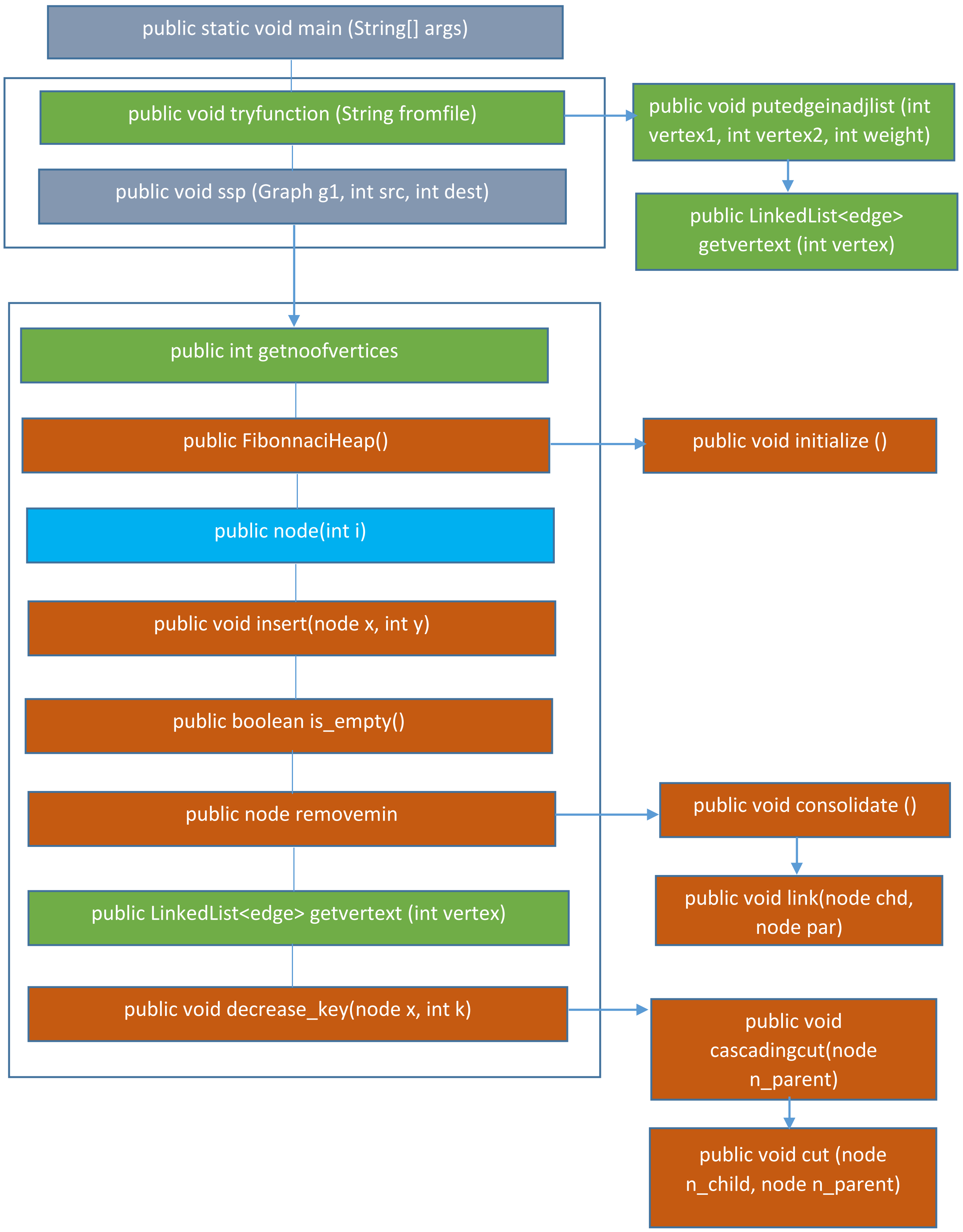
Compiler used: Java

Number of files: ssp.java

All the different classes are in this file itself.

Structure of program along with function prototypes is shown on the next page. Function explanation is presented as the last topic.

Structure of the Program


public static void main (String[] args)
public void tryfunction (String fromfile)
public void putedgeinadjlist (int vertex1, int vertex2, int weight)
public LinkedList<edge> getvertext (int vertex)
public void ssp (Graph g1, int src, int dest)
public int getnoofvertices
public FibonnaciHeap()
public void initialize ()
public node(int i)
public void insert(node x, int y)
public boolean is_empty()
public node removemin
public void consolidate ()
public void link(node chd, node par)
public LinkedList<edge> getvertext (int vertex)
public void decrease_key(node x, int k)
public void cascadingcut(node n_parent)
public void cut (node n_child, node n_parent)

In the structure of program, the function which belong to the same class are represented by the same colors.

ssp class function are represented by **grey color.**

node class function is represented by **light blue color**

fibonacciheap class is represented by **brown color**

Graph class is represented by **green color**

Functions Explanation:

public static void main (String[] args)

- This is the first function which is called when the arguments are passed on the command line.

public void tryfunction (String fromfile)

- tryfucntion is method used to parse the input text graph text file and then create the adjancey list bases on the scans performed over the input file

public void ssp (Graph g1, int src, int dest)

- Dijkstra algorithm: This is main algorithm which implements the shortest path for a given source and destination based on the adjacency list and using Fibonacci heap as means to select the minimum distance next hop vertex and then repeatedly perform this till we reach the destination

public int getnoofvertices

- Method used to access the number of vertices in the graph given

public FibonnaciHeap()

- Constructor call for creating FibonacciHeap obj

public node(int i)

- This method is basically a constructor to initialize a node with the its node identity

public void insert(node x, int y)

- Insert into Fibonacci: Whenever a node is to be inserted into FibonacciHeap this function is called.

root list; minimum node pointer is

public boolean is_empty()

- Check to see if the heap is empty or not

public node removemin

- This is one of the main and important functions for the FibonacciHeap. It removes the minimum node element from the heap. If the node being removed has children all are sent to the root list of FibonacciHeap and then consolidation is called

public LinkedList<edge> getvertext (int vertex)

- Get the neighbours of any particular vertex say 0 or 1 etc.

public void decrease_key(node x, int k)

- Decrease key :
  Case1: When the property of FibonacciHeap that key of parent is less than children is not violated.
  a) Decrease key of x
  b) Change heap minimum pointer (if necessary).

  Case2: When the property of FibonacciHeap that key of parent is less than children is violated.
  a) Decrease key of x.
  b) Cut the tree rooted at x. Then meld into root list, and mark its child cut as false.
  c) If parent of node x has child cut value false, mark it as true.
  d) If parent of node x has child cut value true. Cut parent meld into root list, and mark its child cut false (and do so recursively for all ancestors that lose a second child).

public void putedgeinadjlist (int vertex1, int vertex2, int weight)

- This function puts edge into the Adjacency list for a given vertex number

public void initialize ()

- Initialize the heap with zero nodes and minimum node set to null as no elements present

public void consolidate ()

- Consolidate is called in FibonacciHeap to pair wise combine all the node based on the degree of nodes such that after consolidation functions no two nodes in [illegible] have the same degree

```
public void link(node chd, node par)
```

- Link is called to to make a same degree node child of another

```
public void cascadingcut(node n_parent)
```

- Cascading cut is used to check if the node being cut has ancestors whose child cut value is true, if this is the case then cut and cascading cut are called for those nodes

```
public void cut (node n_child, node n_parent)
```

- Child cut is used to remove the node from sibling list at the level at which it is parent, Also the same node is inserted into the top level root list and minimum is updated if necessary