COP 5536: Advanced Data Structures

Spring 2015

Instructor: Dr. Sartaj Sahni

Programming Project

Dijkstra's Algorithm Implementation Using Fibonacci Heap
And
Implementation of Routing Scheme using binary Trie

Submitted By:

Name: Akshat Sharma

UFID: 94581499

E-‐mail: akshatsharma0106@gmail.com

Department of Computer and Information Science and Engineering

University of Florida

Compile Info

Compiler used: Standard Java Compiler (javac)

How to compile:

Part1:javac ssp.java
Part2:javac routing.java

How to run:

Part1:

java ssp file_name source_node destination_node

Part2:

java routing file_name_1 file_name_2

source_node destination_node

Structure of Program

Classes Used

- **Part 1 -**

  1. ssp class: To find shortest path between two nodes in an undirected graph using fibbonacci heaps implemented using Dijkstra's algorithm.

- **Part 2 -**
  1. routing class: Implementation of Routing Scheme using binary Trie
  2. NodeStructure class: Defines the structure of node.

Function Prototype

➢ **Part 1 -**

Function: public static void main(String[] args)
Description: The Main method which creates an object of class ssp. It stores the values of source node and destination node taken from the input arguments and then calls all the following method in sequence:-

- **Function:** public void readFile(String inputFile)
  Description: use readFile() method given to read the input file and store the values

- **Function:** public void createGraph ()
  Description: use createGraph() method given above to find the weights between two edges edge1 and edge 2 from the input file and store the values in graph.

- **Function:** public void My_Fibonacci_Heap(int x)
  Description: use My_Fibonacci_heap() method given above to initialize the values of a fibonacci heap tree with x

- **Function:** public int findCost(int totalNodes, int source, int dest)
  Description: use findCost() method given above
  to find the cost of the shortest path
  from the source node to destination node
  given in the input file.

- **Function:** public void push(int x, int c)
  Description: use push() method given above
  to insert values into heap

- **Function:** private void moveUp(int i)
  Description: use moveUp() method given above
  to manage heap

- **Function:** private void moveDown(int i)
  Description: use moveDown() method given above
  to manage heap

- **Function:** public void writeFile()
  Description: use writeFile() method to write
  the final result in the output
  file and show output on console.

➢ **<u>Part 2 -</u>**

Function: public static void main(String[] args)
Description: The Main method which creates an object
of class routing. It stores the
values of source node and destination
node taken from the input arguments and then
calls all the following method in sequence:-

- **Function:** public void readFile(String inputFile)
  Description: use readFile() method given above
  to read the input file and store the values

- **Function:** public void createGraph ()
  Description: use createGraph() method given above
  to find the weights between two edges
  edge1 and edge 2 from the input
  file and store the values
  in graph.

- **Function:** public void My_Fibonacci_Heap(int x)
  Description: use My_Fibonacci_heap() method given above
  to initialize the values of a fibonacci heap
  tree with x.

- **Function:** public int findCost(int totalNodes, int source, int dest)
  Description: use findCost() method given above
  to find the cost of the shortest path
  from the source node to destination node
  given in the input file.

- **Function:** public void push(int x, int c)
  Description: use push() method given above
  to insert values into heap

- **Function:** private void moveUp(int i)
  Description: use moveUp() method given above
  to manage heap

- **Function:** private void moveDown(int i)
  Description: use moveDown() method given above
  to manage heap

- **Function:** public static Hashtable<Integer,String> findIp(String filename)
  Description: use findIp() method given above
  to save the IP address of
  each vertex as read from the
  input file in args[1].

- **Function:** public static String toBinary(int number)
  Description: use toBinary() method given above
  to return String in binary form.

- **Function:** public void insertTo(String data)
  Description: use insertTo() method given above
  to return String in binary form.

- **Function:** public void removeFromTrie(String data)
- **Description:** use removeFromTrie() method given above
  to remove from the Subtrie with the same
  next hop.

- **Function:** protected String postOrderTraverse (NodeStructure root)
  Description: use postOrderTraverse() method given above
  to return String in binary form.

- **Function:** public String getString(String data)
- **Description:** use getString() method given above
  to return String in binary form.

- **Function:** public void writeFile()
  Description: use writeFile() method to write
  the final result in the output
  file and show output on console.

Program Result

- **Part 1 –**

```
$ ./ssp  input_1000_50_part1.txt 0 999

12
0 670 18 184 856 999
```

- **Part 2 –**

```
$ ./routing input_graphsmall_part2.txt   input_ipsmall_part2.txt   0 3

3
1100000000000010  11000000000000101010100000000001  110000000000001
010101000000001
```