School of Computing
National University of Singapore
CS2010 Data Structures and Algorithms 2
Semester 2, AY 2015/16

Lab 4 – Empirical Complexity Analysis of Minimum Spanning Tree Kruskal vs Prim!

Assigned – March 23 (Wednesday), 2016
Lab – April 4th (Monday by 11.59pm), 2016

Objective
Analyze the time complexity between two Minimum Spanning Tree (MST) algorithms – Kruskal's Algorithm and (Lazy) Prim's Algorithm.

Preparation
Lab 4 zip file contains a main function, two algorithms' implementations and other auxiliary classes:

- KruskalMST.java: implementation of the Kruskal's Algorithm
- LazyPrimMST.java: implementation of the (Lazy) Prim's Algorithm
- Lab4Main.java: the class containing main function
- g1.txt: a very dense graph
- g2.txt: a relative sparse graph

Download the file lab4.zip from IVLE into your working directory with three Java files above. Uncompress the zip file and you should find the following document, Java source files and data files: `CS2010-Lab4.pdf, Lab4Main.java, KruskalMST.java, LazyPrimMST.java, Bag.java, Edge.java, EdgeWeightedGraph.java, In.java, MinPQ.java, Queue.java, Stack.java, UF.java, sample.txt, g1.txt, g2.txt`

Please go through the codes before starting to this lab.

Task:

Your task is to empirically measure the complexity of two algorithms (Kruskal's Algorithm and Lazy Prim's Algorithm) for computing MST on different input graphs. Both of these methods rely on Priority Queues (binary heaps) to sort the edges. However, Kruskal has additional overhead in performing union-find operations, while Prim has overhead in scanning the edge list of edges to add as the tree grows.

Your task is the following:

- For **both methods**, count the number of times edges are compared in the priority queue.
- For **Kruskal**, count the number of iterations in the find operation which is performed in the Union Find data structure (see code `UF.java`).
- For **Prim**, count the number of iterations in the scan operation to find all incident edges to a vertex recently added to the Tree (see code `scan` method in `LazyPrimMST.java`).
- **Additionally**, in the given LazyPrimMST.java, the termination condition of computing MST can be improve with a simple code modification. See if you can determine what to modify.

Edge Compares: To compare two edges (E1, E2), the code will first compare E1 < E2; if not, then compare E1 > E2 and finally get the result. In this assignment, the this whole comparison procedure should be counted a 1 edge compare.

Counting UF and scan iterations. Consider the following code to help remind you how to count loops:

```
void foo1(…) {
      ……
      int flag = 5;
      While (flag > 0) {
          flag--;
          ……
          yourCounter++;
      }
      ……
}
void foo2(…) {
      ……
      ArrayList<Integer> array = new ArrayList<>();
      array.add(1);
      array.add(2);
      array.add(3);
      array.add(4);
      for (int i: array) {
          ……
          yourCounter++;
          if (i == 3)    break;
          ……
      }
      ……
}
```

In foo1(…), the number of iterations is 5. In foo2(…), the number of iterations is 3. If the foo1(…) is called 10 times, the number of iterations in foo1(…) is 50.
You can use yourCounter as shown in this example to count the number of iterations.

【input】

The data of a graph is in a text file.

First line is the number of vertices N. Second line is the number of edges M. The following M lines are edges and weights.

【Output】

The number of comparisons and iterations for Kruskal's Algorithm, Lazy Prim's Algorithm and your improved Lazy Prim's Algorithm.

Please follow the instruction in `main` method of `Lab4Main.java`.

Example of input file and output and their format

【input file(sample.txt)】

```
8              ← number of vertices
16             ← number of edges
4 5 0.35       ← v1 v2 w: v1 and v2 are integers, two vertices; w is double, the weight of edge (v1,v2)
4 7 0.37
5 7 0.28
0 7 0.16
```

```
1 5 0.32
0 4 0.38
2 3 0.17
1 7 0.19
0 2 0.26
1 2 0.36
1 3 0.29
2 7 0.34
6 2 0.40
3 6 0.52
6 0 0.58
6 4 0.93
【output】
Number of Vertices = 8
Number of Edges = 16
Compute MST by Kruskal Algorithm
  Heap edge comp + UF find = 81 + 19 = 100
  Time of Computing MST by Kruskal Algorithm = 0 ms
Compute MST by Lazy Prim Algorithm
  Heap edge comp + Scan = 72 + 32 = 104
  Time of Computing MST by Lazy Prim Algorithm = 0 ms
Compute MST by improved Lazy Prim Algorithm
  Heap edge comp + Scan = 71 + 32 = 103
  Time of Computing MST by improved Lazy Prim Algorithm = 0 ms
```

Submission Instruction (IVLE submission) – Due April 4th (Monday by 11.59pm), 2016

Please submit your code to IVLE. You only need to zip up your source code.

1. Please put your source files and a readme file (if you have) in a folder and submit the entire folder zipped. Do not submit data files and whole project files. Use the following convention to name your zipped folder: MatriculationNumber_yourName_Lab4. For example, if your matriculation number is A1234567B, and your name is Chow Yuen Fatt, for this assignment, your file name should be A1234567B_ChowYuenFatt_Lab4.zip. Points will be deducted if you don't follow the naming scheme.
2. It is up to you to test to make sure that your Java code in the zipped file is complete. It is recommended that you unzip your Java code in a different location and test it to make sure it runs.
3. If you use a special way to run your code or receive the input from user, please write them clearly in a readme.txt file. If you do anything beyond the requirement or any other special things, write them in the readme.txt file.