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

Lab 3 –Space Analysis of Graphs
Assigned – March 2 (Tuesday), 2016
Lab – March 14 (Monday by 11.59pm), 2016

Objective
Analysis of the space requirements between adjacency-lists representation and adjacency-matrix representation. See if you can come up with a more compact adjacency "matrix" representation – your representation may not be a matrix.

Preparation
Lab 3 zip file contains two graph implementations:

- AdjMatrixGraph.java : implements, adjacency-matrix representation.
- AdjListGraph.java : implements, adjacency-lists representation.

Please go through the codes before starting to this lab.

Your Task
- Basic Requirement (70%): Perform an empirical measurement of how efficient each method in storing the same graph.

- Additional Task (+30%): Create a new AdjMatrixMoreEfficientGraph.java that is more efficient than the existing AdjMatrixGraph by storing only ½ of the full matrix.

Get started:

Download the file lab3.zip from IVLE into your working directory with the two Java files above. Uncompress the zip file and you should find the following document and data files: `CS2010-Lab3.pdf`, `AdjMatrixGraph.java`, `AdjListGraph.java`, `Bag.java`, `In.java`, `L3Main.java`, `Size.java`, `Stack.java`, `StdIn.java`, `StdOut.java`, `StdRandom.java`, `g1.txt`, `g2.txt`, `sample.txt`.

Task:

(Basic) Your task is to compare graph two different representations using two graphs implementations for different input graphs. In particular, we compute the two graph representations to determine the amount of memory usage needed to store the graph and its overall space efficiency.

Recall that an adjacency list uses a linked list ("bag" in our textbook implementation). For every item inserted in the linked list, there is a reference variable to the next node. Note that lists store redundant copies of the edge, for example, edge (3-5) has a link in vertex 3 for vertex 5, and a link in vertex 5 for vertex 3. The adjacency matrix representation uses a Boolean 2D matrix to store the edges. The 2D matrix also has redundancies, element V[2][4] = true and V[4][2]=true for edge (3-5).

Your task is to compute how much memory is being used the adjacency-list and the adjacency-matrix to store the edge information for different input graphs. For various reasons, Java doesn't reveal the size of the underlying primitive data types, but we will make an educated guess to use in our assignment.

Integer variable requires 32 bits or 4 bytes.
References variable requires 64 bits or 8 bytes.
Boolean variable requires 8 bits or 1 byte.

The lab's task is to compute the necessary memory to store the edges of the graph and how effective this is. Your output should be as follows. Create a graph object using both the adjacency list and adjacency matrix implements. For each, output the following:

1. number of vertices
2. number of edges
3. Print the graph (i.e. for each vertex – a list of all vertices that share an edge with this vertex) [this is just to show that API works]
4. Output the memory usage info – print all below

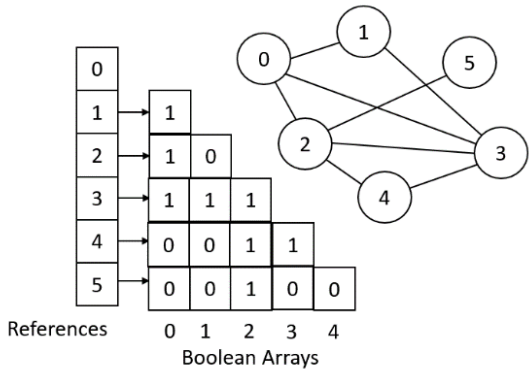
(a) Number of edge items stored times memory required to represent the edges[1]
(b) Total amount of memory used (including memory for total # of integers, reference variables, and/or booleans) to store the edges.
(c) Efficiency (edge memory)/(total memory)

An example output for test case `sample.txt`.

```
1. Number of vertices = 6
2. Number of edges = 8
3. Output of the graph using adjacency list:
6 vertices, 8 edges
0: 2 1 5
1: 0 2
2: 0 1 3 4
3: 5 4 2
4: 3 2
5: 3 0
4. Adjacency list
 (a) Memory needed to record edges = 32
 (b) Total amount of memory used  = 192
 (c) Efficiency  = 16.666666
5. Output of the graph using matrix:
6 vertices, 8 edges
0: 2 1 5
1: 0 2
2: 0 1 3 4
3: 5 4 2
4: 3 2
5: 3 0
6. Adjacency matrix
 (a) Memory needed to record edges = 8
 (b) Total amount of memory used  = 36
 (c) Efficiency  = 22.222221
```

[1] To simplify the computation, you only need to compute the memory needed to store all the nodes in the Linked List. You do not need to consider the additional overhead of the Bag class (private variables) or the array of references to each bag class.

Additional Task (30%)
Can you make the matrix implementation more efficient, or at least consume less overall memory? Recall that the matrix is symmetric, since an edge (v1,v2) = (v2, v1), so you can instead only store ½ of the matrix (see diagram below).



For your new class implementation, also output the efficiency and memory usages.

Your code will need to be able to use this implementation to support the full Graph API as done in the other implementations (i.e. you will need to also allow it to return an Iterable Object for each vertex). If you implement this, also print out the information including efficiency as done with the adjacency-list and adjacency-matrix above.

Submission Instruction (IVLE submission) – Due March 14 (Monday by 11.59pm), 2016

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

1. Please put your Java codes in a folder with the source files and submit the entire folder zipped. Use the following convention to name your zipped folder: MatriculationNumber_yourName_Lab3. 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_Lab3.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.