CS2040 Lab 8
Graph DS

Lab 8 – Graph DS

- First lab not introducing new Java API!
- From this point on, almost all relevant data structures and algorithms will have to be manually coded
- Graph data structures will be used extensively throughout the remainder of this semester
  - As such, please do voice out if you have any difficulties understanding the material for this lab

Lab 8 – Graph DS – General

- For the following slides, we define an edge to contain three different pieces of information, as follows:
  - w – the weight of each edge. An edge is typically considered to have an edge weight of 1 if it is in an unweighted graph
  - u – the vertex which the edge is pointing from (or where the ‘tail’ of an edge is when drawn)
  - v – the vertex which the edge is pointing to (or where the ‘head’ of an edge is when drawn)
- We also use the variables V and E to refer to the total number of vertices and the total number of edges in the graph, respectively

Lab 8 – Graph DS – Adjacency Matrix

- For both unweighted and weighted graphs, we typically declare an adjacency matrix as follows:
  - `int[][] adjMatrix = new int[V][V];`
- An edge is typically stored as follows:
  - `adjMatrix[u][v] = w; // w is 1 in the case of unweighted graphs`
- An entry in the array with value 0 is usually assumed to indicate that the edge (u, v) is not present in the graph
  - You may need to find a workaround for this if the graph contains edges with weight 0

Lab 8 – Graph DS – Adjacency List

- For unweighted graphs, we declare an adjacency list as follows:
  - `ArrayList<ArrayList<Integer>> adjList = new ArrayList<ArrayList<Integer>>();`
- An edge is typically stored as follows:
  - `adjList.get(u).add(v);`
- For weighted graphs, we declare an adjacency list as follows:
  - `ArrayList<ArrayList<IntegerPair>> adjList = new ArrayList<ArrayList<IntegerPair>>();`
- An edge is typically stored as follows:
  - `adjList.get(u).add(new IntegerPair(v, w));`

Lab 8 – Graph DS – Adjacency List

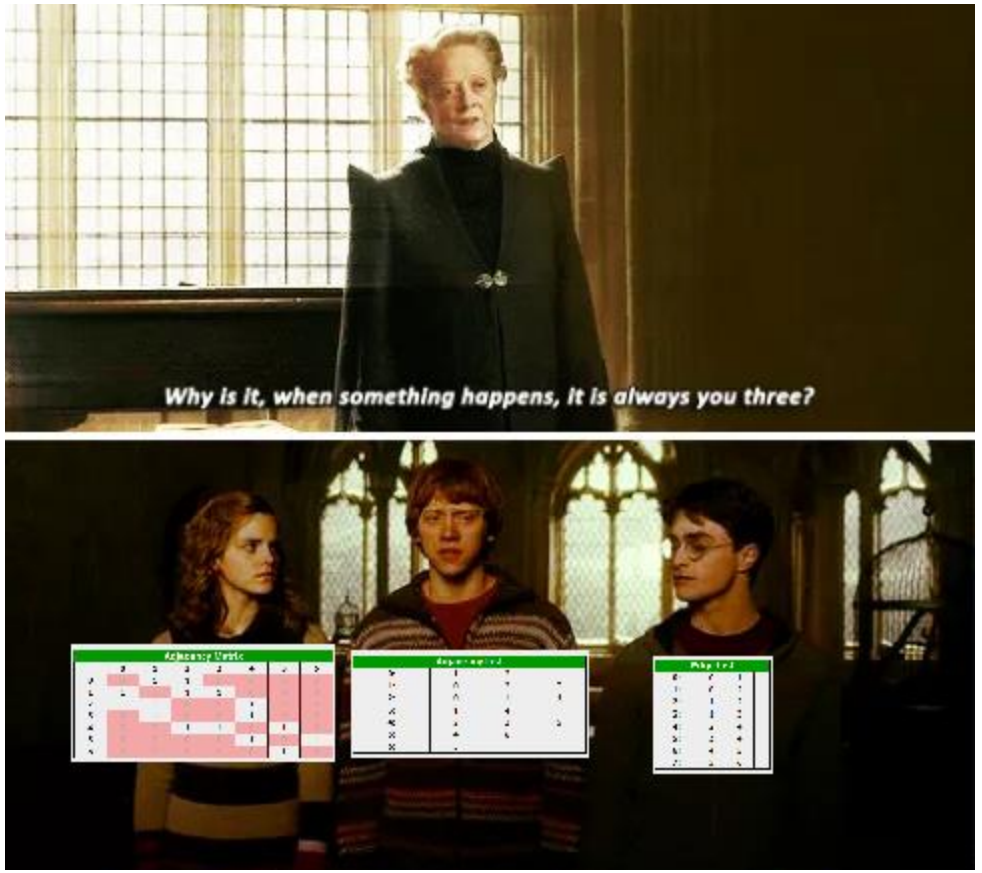
- Note that unlike 2D arrays, 2D ArrayLists will require you to manually add the second layer of ArrayLists eg:
  - `ArrayList<ArrayList<Integer>> adjList = new ArrayList<ArrayList<Integer>>();`
  - `for (int i = 0; i < V; i++) {`
  - `    adjList.add(new ArrayList<Integer>());`
  - `}`
- Note: ensure that you are adding a new ArrayList per call (like the code above), and not adding the same ArrayList multiple times

* It is also possible to create Adjacency Lists using HashMaps, but usually only useful in some situations

Lab 8 – Graph DS – Edge List

- For unweighted graphs, we declare an edge list as follows:
  - `ArrayList<IntegerPair> edgeList = new ArrayList<IntegerPair>();`
- An edge is typically stored as follows:
  - `edgeList.add(new IntegerPair(u, v));`
- For weighted graphs, we declare an edge list as follows:
  - `ArrayList<IntegerTriple>> edgeList = new ArrayList<IntegerTriple>();`
- An edge is typically stored as follows:
  - `edgeList.add(new IntegerTriple(u, v, w));`

Lab 8 – Graph DS



Graph for thought

- Which representations are better if I have space / memory constraint?
- If I have a multi-graph (can have multiple edges between two vertices), which representations will work? Any modifications that must be made?

(KIV) Which representation is better for our algorithms?

One-Day Assignment 7 – Weak Vertices

- Given a graph in adjacency matrix form, determine which vertices are not part of a triangle
- Three distinct vertices a, b, and c form a triangle if
  - Edge (a, b) is present in the graph; and
  - Edge (a, c) is present in the graph; and
  - Edge (b, c) is present in the graph