CS6905 (AGA) Fall 2023 – Assignment 7 (Major) Due Wednesday November 15, 2023, by 10pm.

This assignment is for you to finish the implementation of the Ford-Fulkerson algorithm and use it in an application.

Note that the data and constructor for `GraphFlow` (item 1) are the same as for `FlowGraph` in assignment 6; the renaming is to keep the pieces of the two assignments clear, since `GraphFlow` needs an additional method (and you may add more). Class `DGraphReach` is as you implemented for Assignment 6, though you can add other methods as appropriate. Methods added need to respect the separation of the classes and their specifications.

You need to:

1. write a `GraphFlow` class that uses both `DGraphWtAL` and `DGraphReach`, to store a flow graph and its residual graph. It should have data as follows:

   - `network` field – of class `DGraphWtAL`. Used to store the flow graph, where the capacity will be stored in an edge's `weight` field and its current flow in the edge's `mark` field.
   - `residual` field – of class `DGraphReach`. Used to store the residual graph. Residual capacities of the edges will be stored in the `weight` field of each edge.
   - `source` – an integer storing the index of the source node of the flow graph.
   - `target` – an integer storing the index of the target node of the flow graph.

   You need to write the constructor for this class, which will take a properly constructed flow graph (of class `DGraphWtAL`) and the indices of the source and target nodes as parameters, store these parameters in the appropriate fields, and build the residual graph.

2. write a `FordFulk()` method for `GraphFlow`, that implements the Ford-Fulkerson algorithm, using the pieces you wrote for Assignment 6. This method needs to compute the maximum flow, storing the flow values on the edges of the flow graph as described above. It will be tested using the provided `DrAGA7.java` code.

3. write a `Projects` class that will solve the Project Selection problem (as discussed in class, at the end of the network flow slides), using your Ford-Fulkerson implementation. You can also add methods to your `DGraphReach` and `GraphFlow` classes, as appropriate to assist in your solution, as long as they are not specific to the `Projects` application. `Projects` needs to support the following methods (as used by `DrAGA7p.java`):

   - `Projects(int` n`)` : constructor that sets up the objects needed to represent the project information for n projects
   - `addPrereq(int` i`, int` j`)` : adds the information that the project at index i is a prerequisite for the project at index j

- `addProfit(int` i`, int` p`)` : adds the information that the project at index i has profit p
- `projSelect()` : returns an integer array of length n (where there are n projects), with a 1 in the positions whose project should be completed and 0 otherwise

Ensure that your code works with the provided `DrAGA7.java` and `DrAGA7p.java` code, which will be used to test your submitted solution. Organize and comment your code appropriately.

Submit on D2L: your `DGraphReach.java`, `GraphFlow.java`, and `Projects.java` files, and the I/O from one test run of your solution for each of the two drivers. Please submit each file as a separate attachment.