

Report OOP Lab (IT3102E) – 20212

Team 1 – Topic 7

Teacher: PhD. Bui Thi Mai Anh

Team member: + Le Dinh Huy – 20194776

+ Pham Minh Dang – 20194737

+ Dang Minh Khoi – 20194781

I. Topic

Given a graph G , we aim to find the shortest paths weights from a particular single source vertex to all other vertices in a directed weighted graph (if such paths exist)

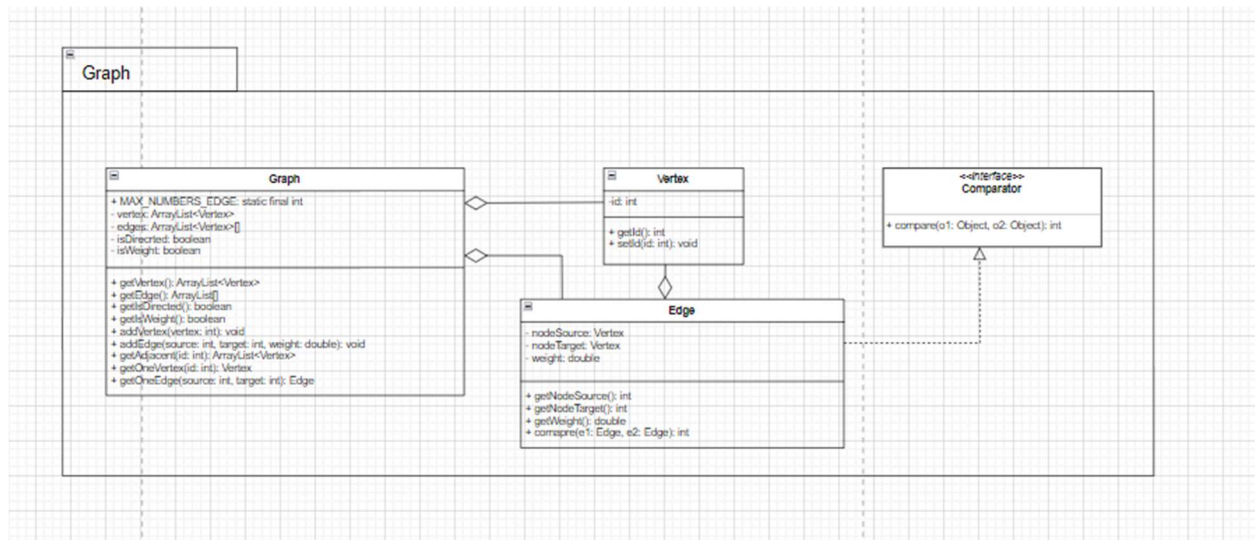
Algorithms:

- Bellman Ford
- BFS
- Dijkstra

II. Design and class diagram

Here is our full class diagram:

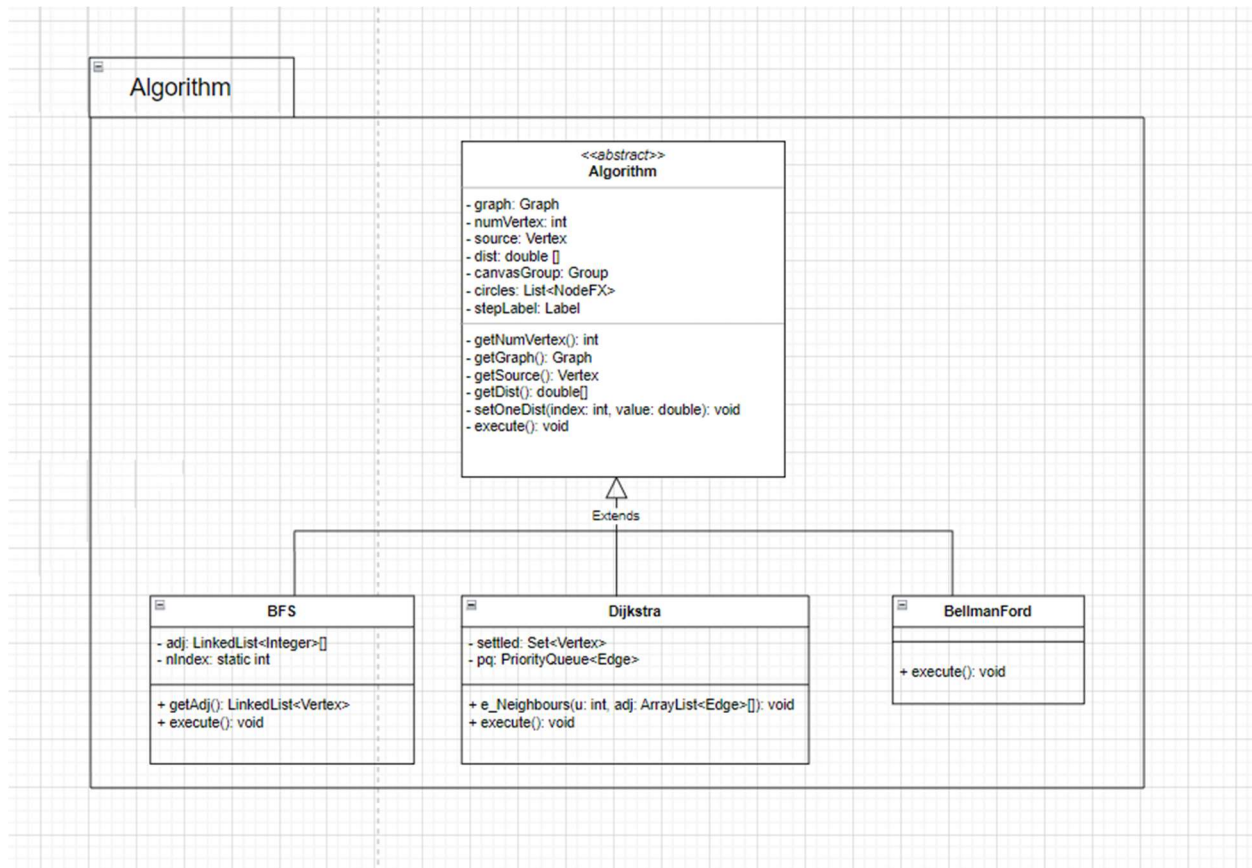
Graph package: In graph package, it includes class Vertex, Edge as follow:



Graph class is aggregation with Vertex class and Edge class. Edge class is implement from Comparator class and aggregation with Vertex class

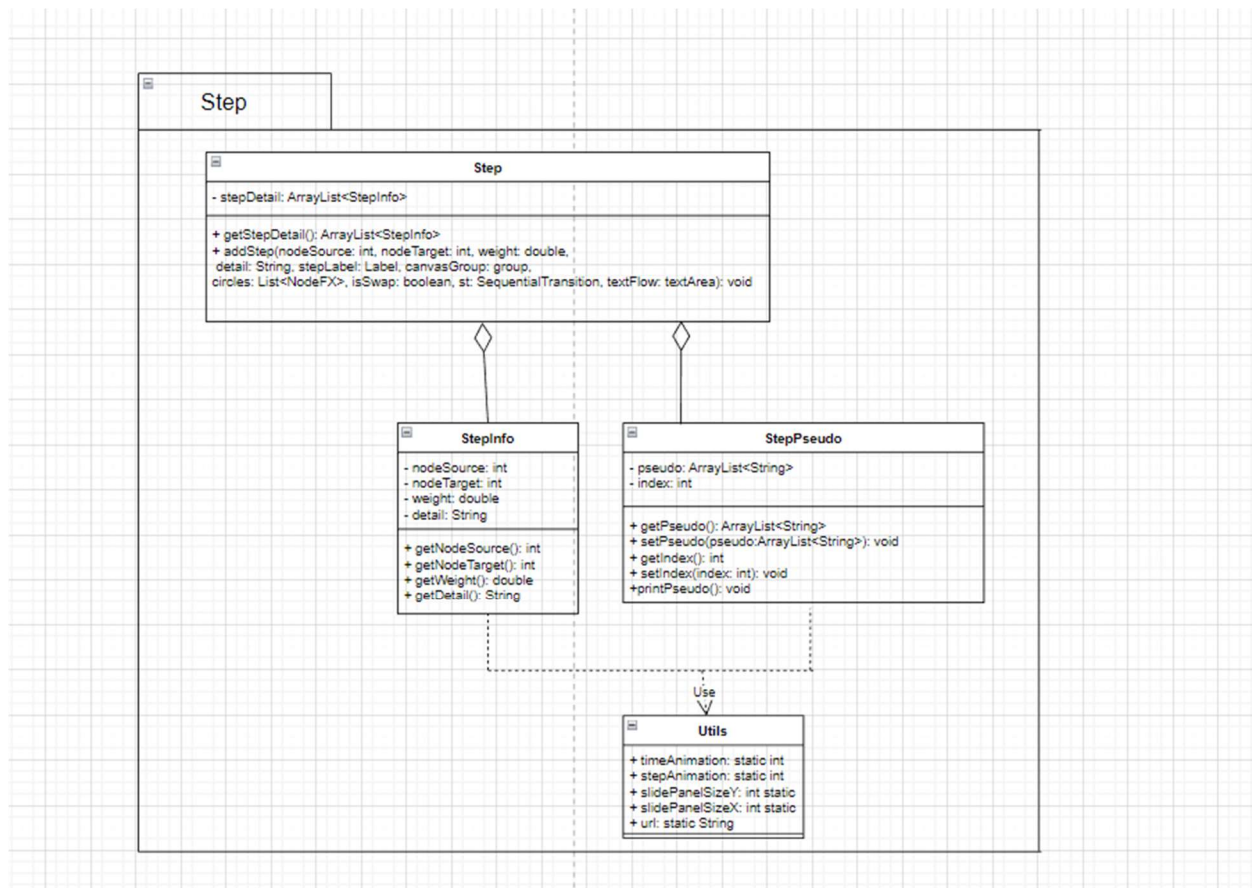
2. Algorithms package

Algorithms is about: Algorithms class is an abstract class and three class BFS, Dijkstra, BellmanFord is inherit from it



3. Step package

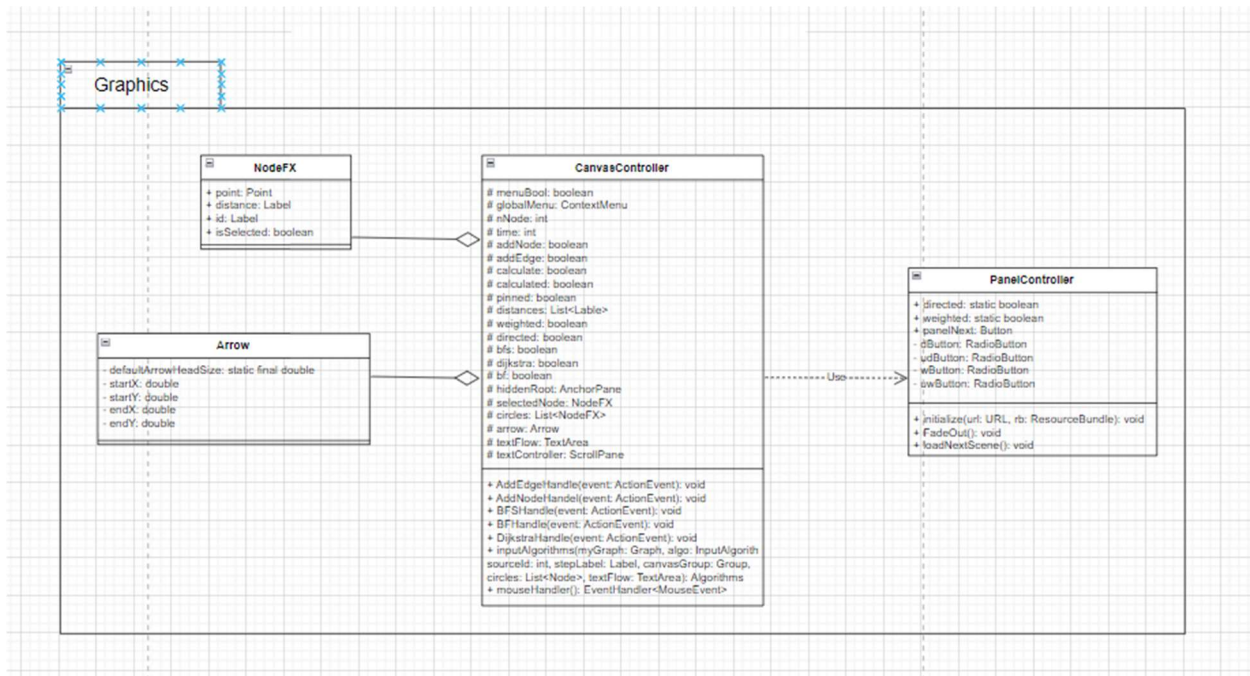
Step package is about: Step class, StepInfo class and StepPseudo class



Step class is aggregation with StepInfo class and StepPseudo class

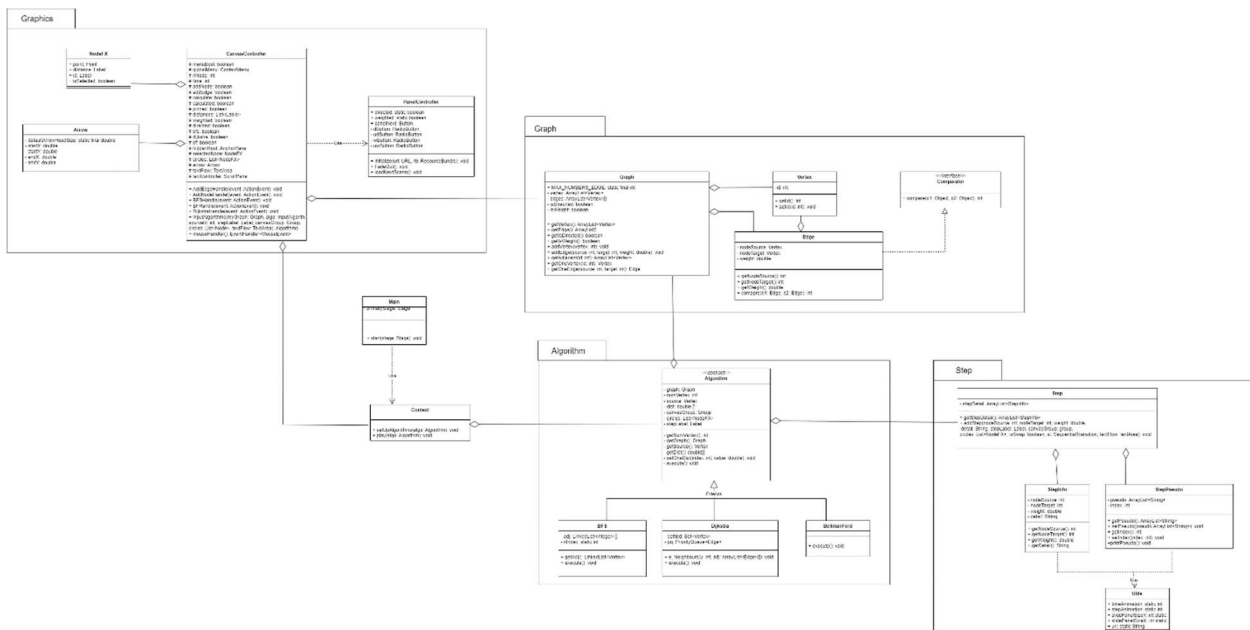
4. Graphics package

Graphics package package is about: CanvasController class, Arrow class, PanelController class, NodeFX class



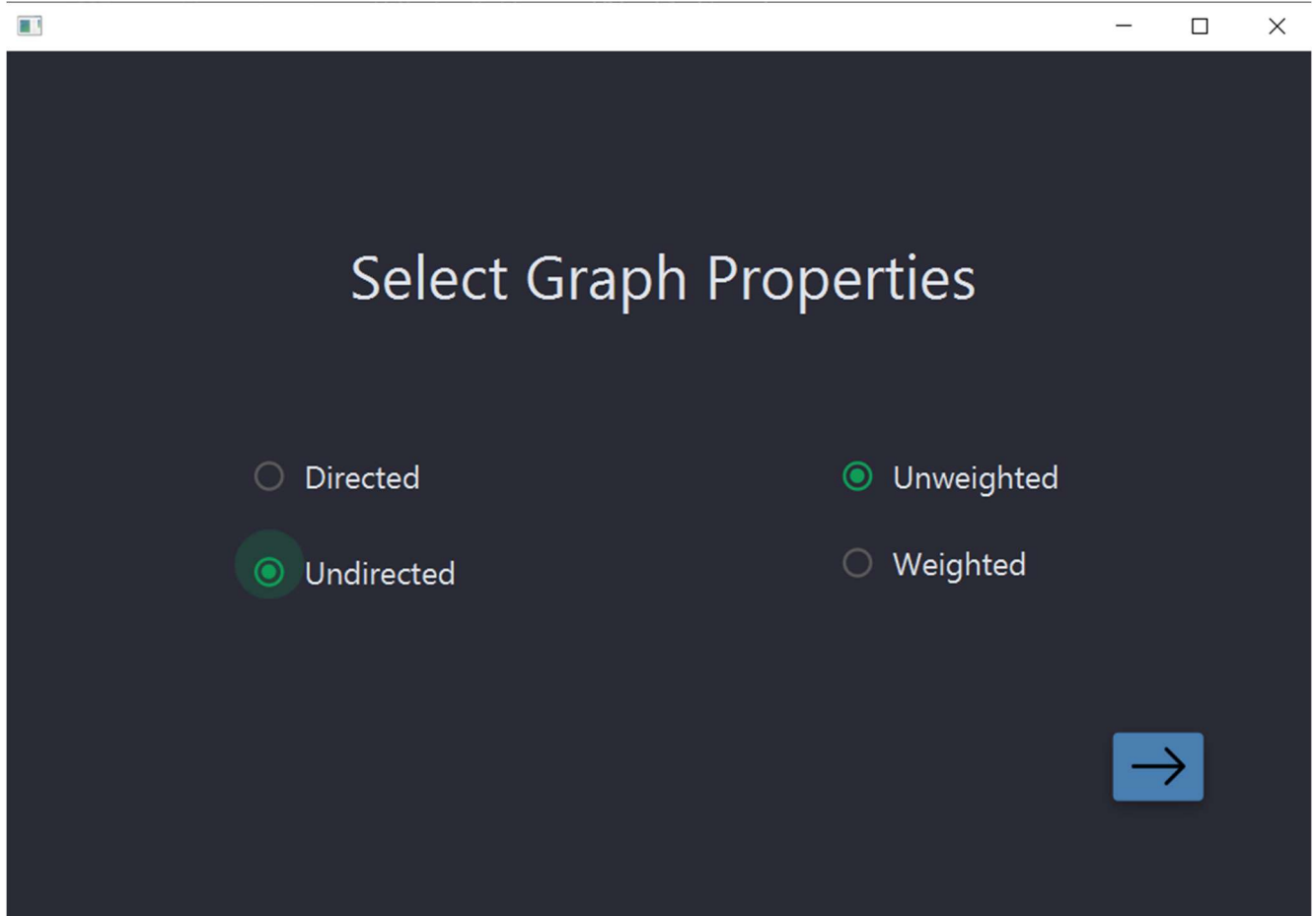
In this package, CanvasController class is aggregation with NodeFX class and Arrow class, and use PanelController class

⇒ And finally, the class diagram become:



III) Running program

- In initial screen, the screen will give you to choose the attributes of the Graph directly, we also validate all cases for users when they decide attributes of Graph:



Select Graph Properties

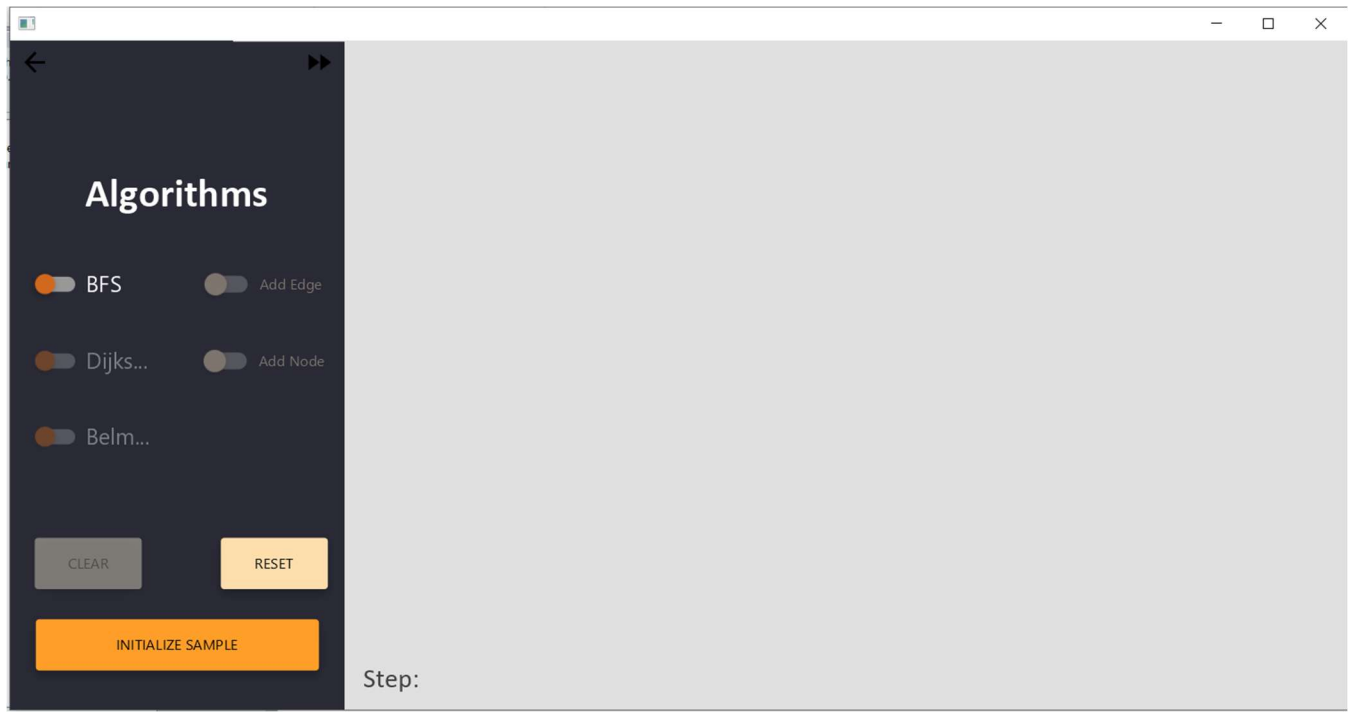
☐ Directed ☒ Undirected

☒ Unweighted ☐ Weighted

→

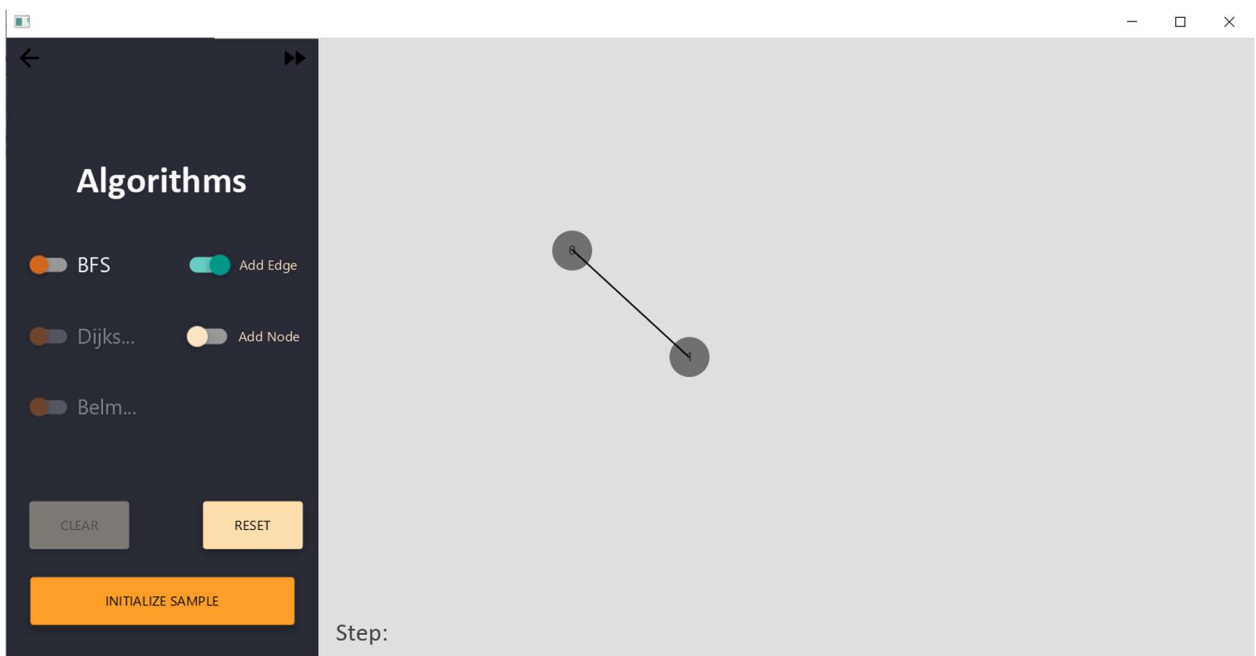
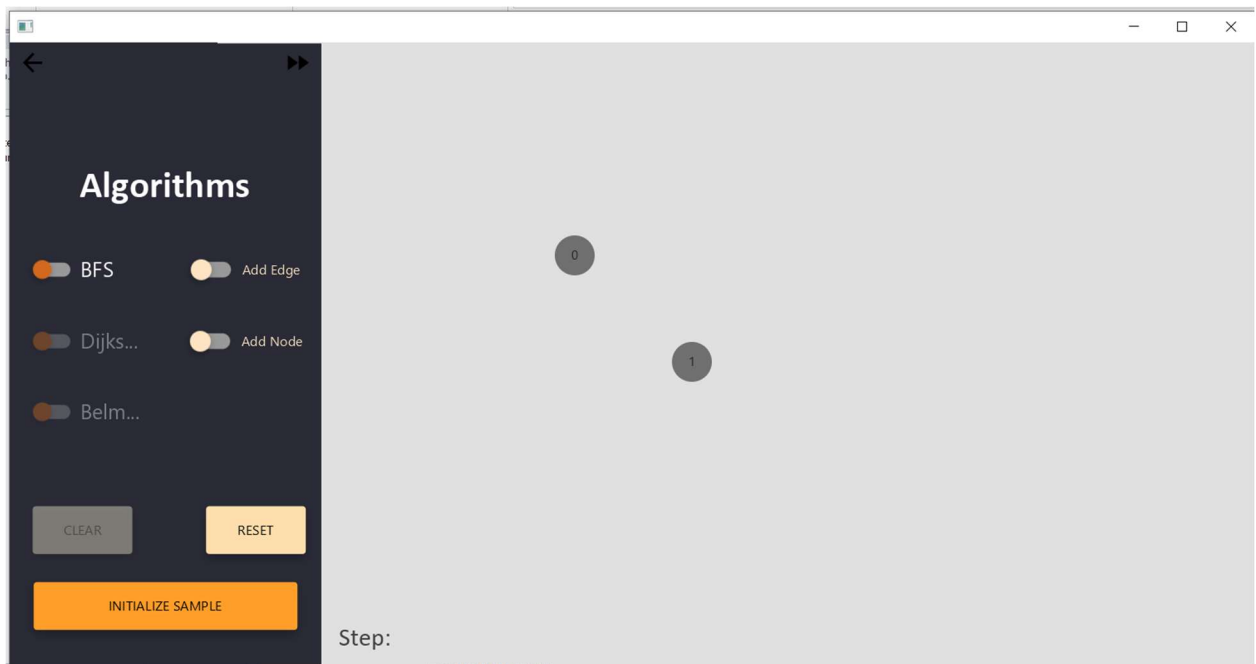
- If you choose unweighted with all case of direction cases, you only can run BFS Algorithm
- If you choose undirected and weighted, you only can run Dijkstra Algorithm
- If you choose directed and weighted, you can run Belmond Ford Algorithm and Dijkstra Algorithm

Here is an example of validation in the attributes of Graph (when you choose unweighted):

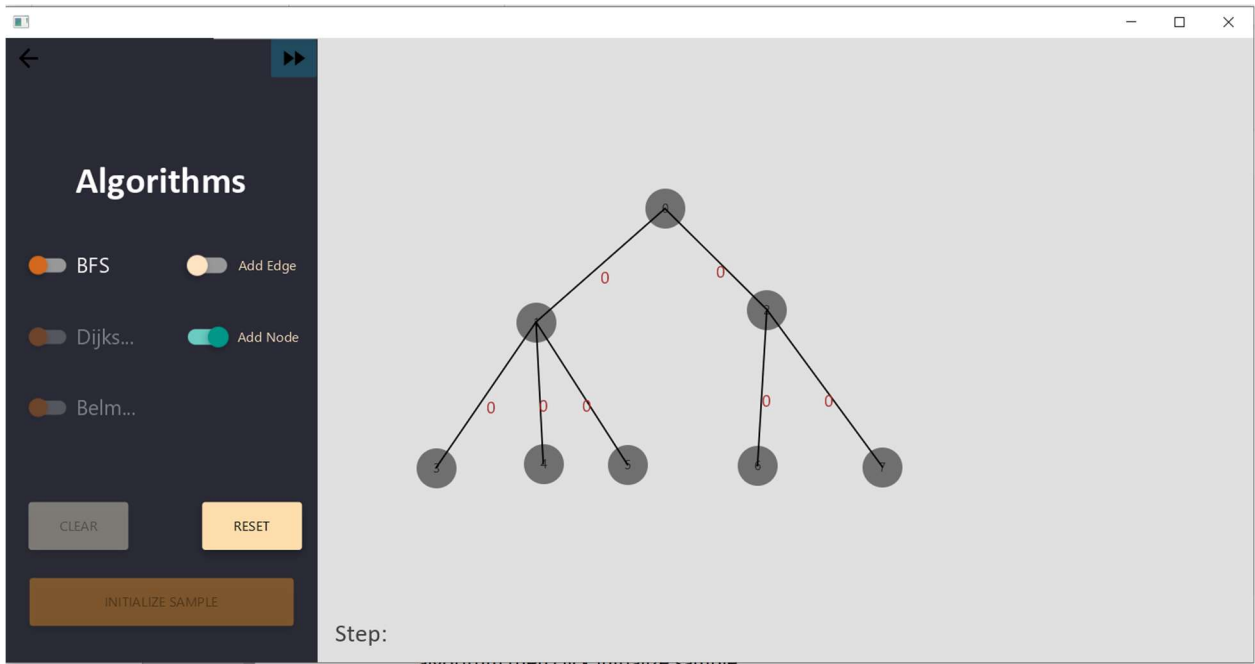


- In algorithm screen, you can add nodes by click to white screen on the right, if there are more than 2 nodes, you can click to two node to add edge between them. Besides, if you want to run it, you just choose BFS and click to one node to start the algorithm

Example of add node and edges:



Otherwise, you can initialize sample for not having to add nodes and edges, by choosing algorithm then click initialize sample. Here is BFS sample:

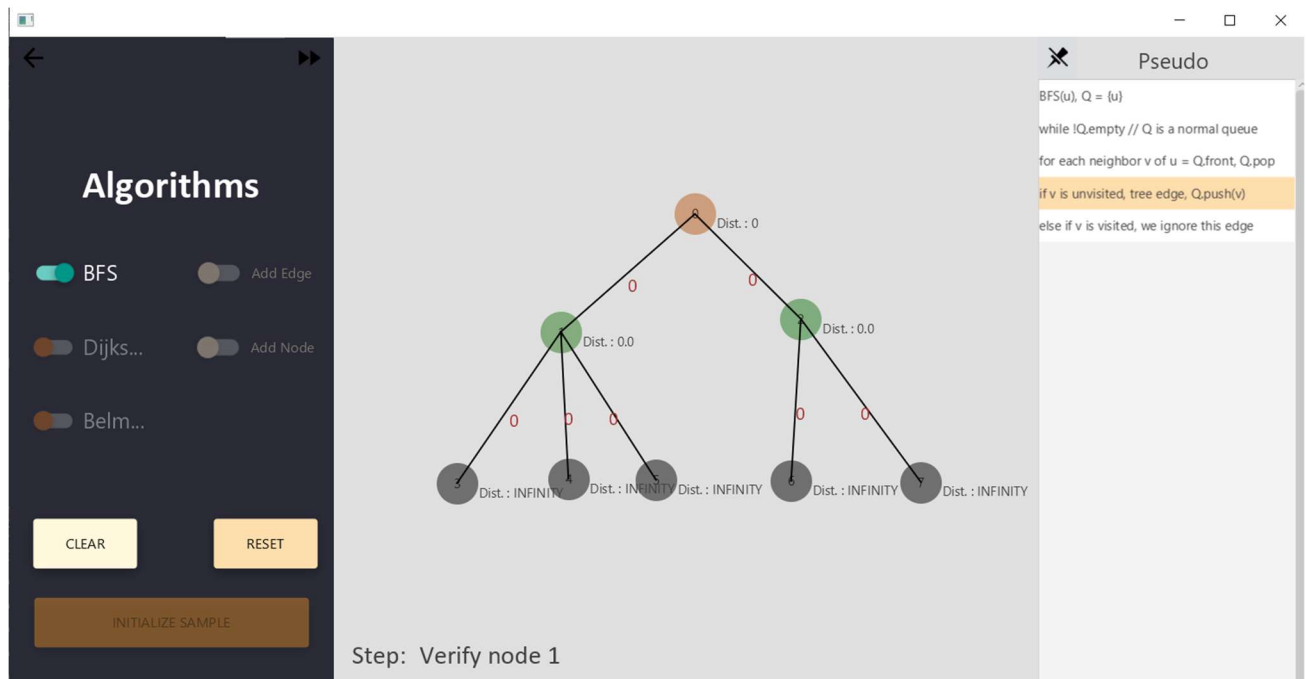


Let see how the program will execute the algorithm:

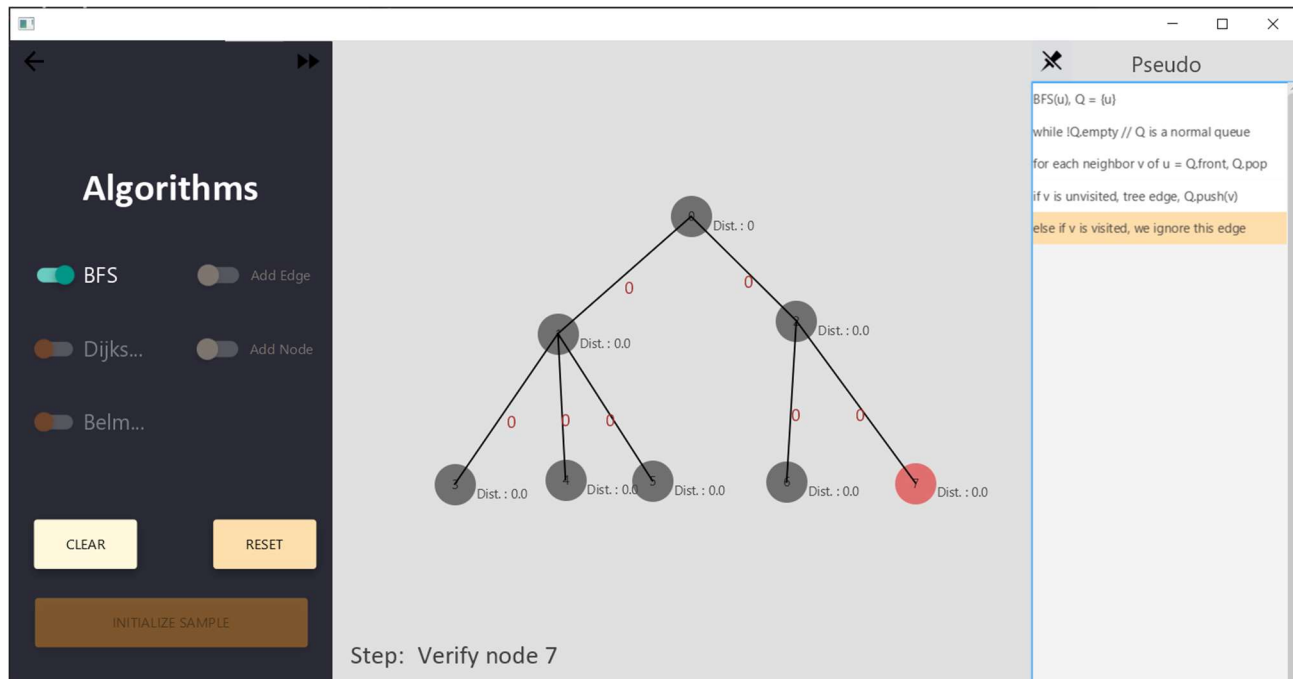
BFS:

+) You can hover to the right side to see panel (which show pseudo step and you can pin it to the screen)

+) When running:

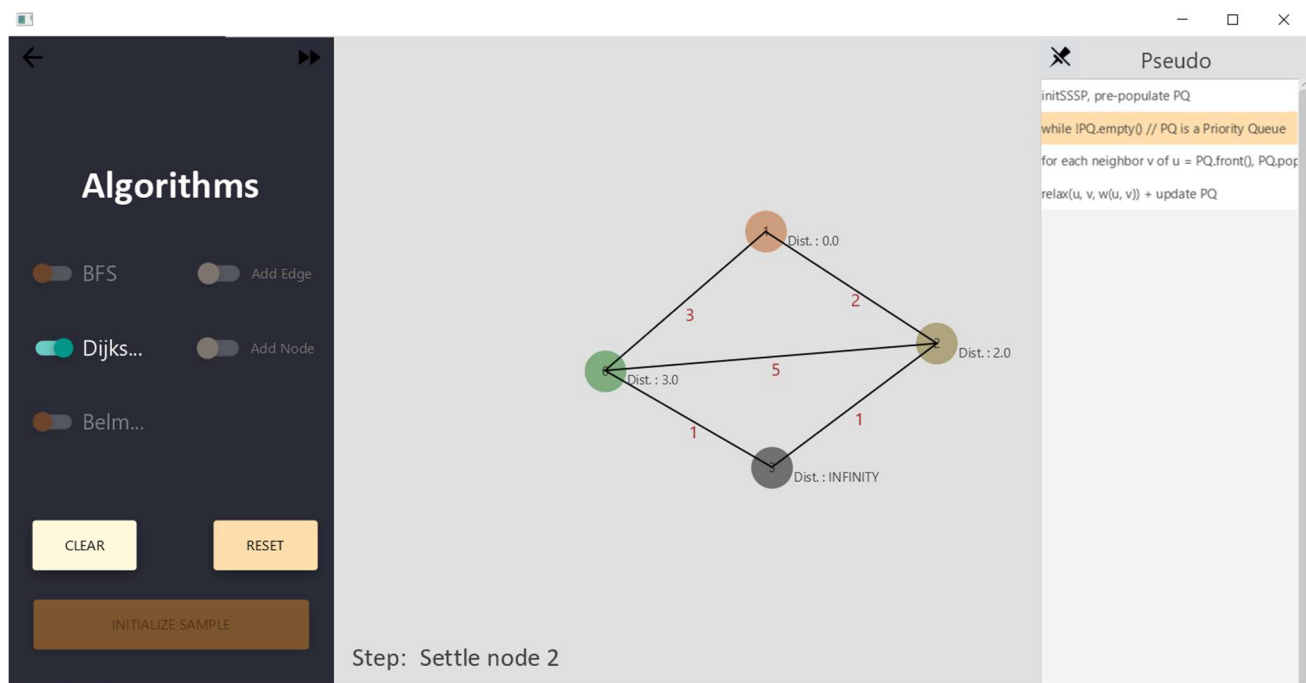


+) After finishing execution:

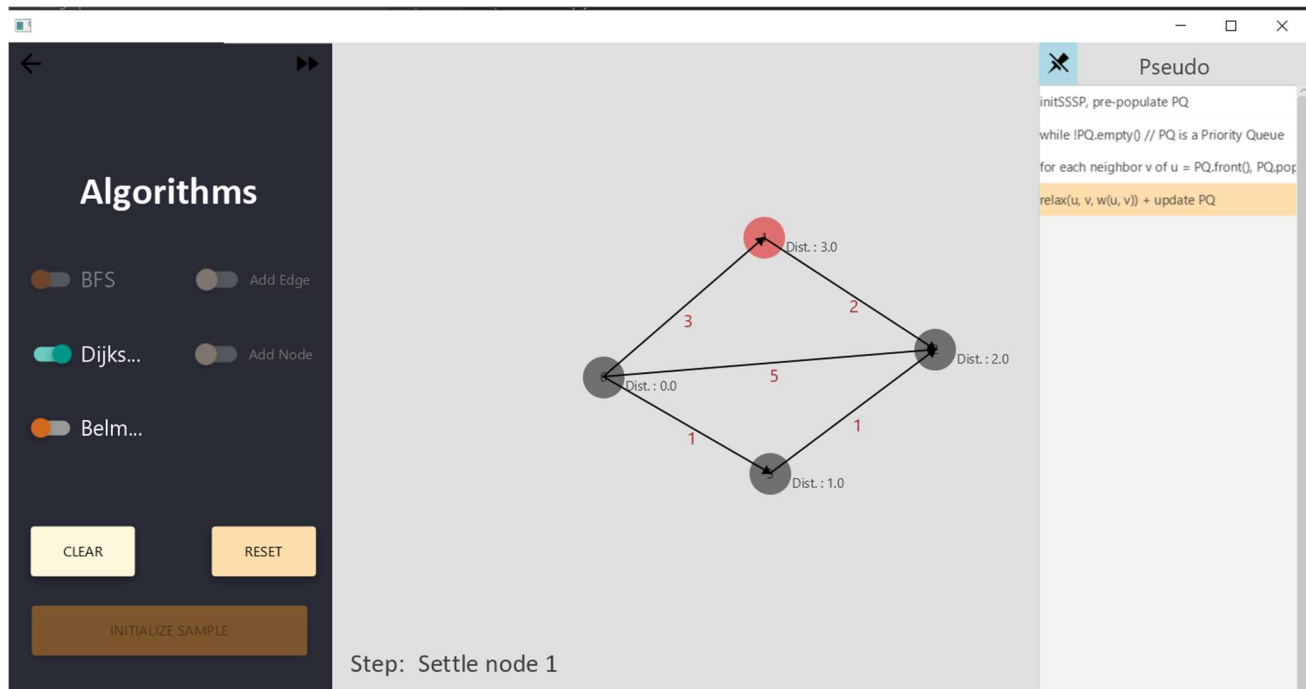


Dijkstra:

+) When running:

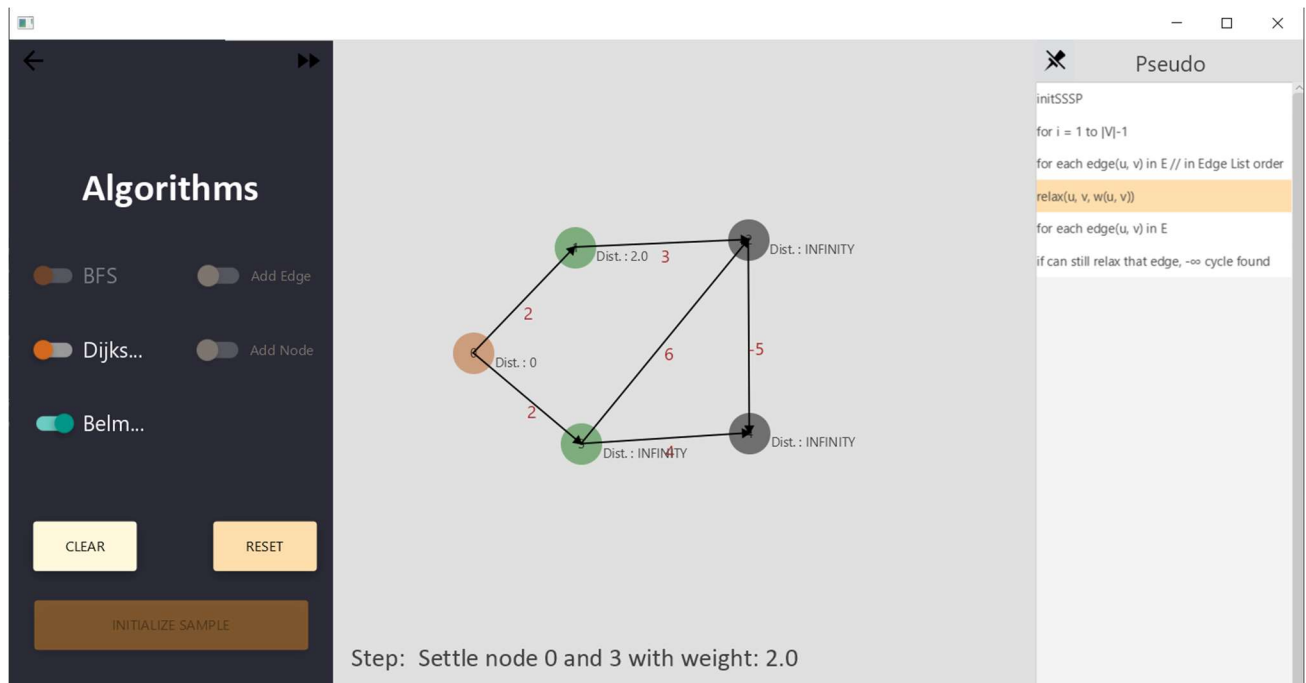


+ After finishing execution:

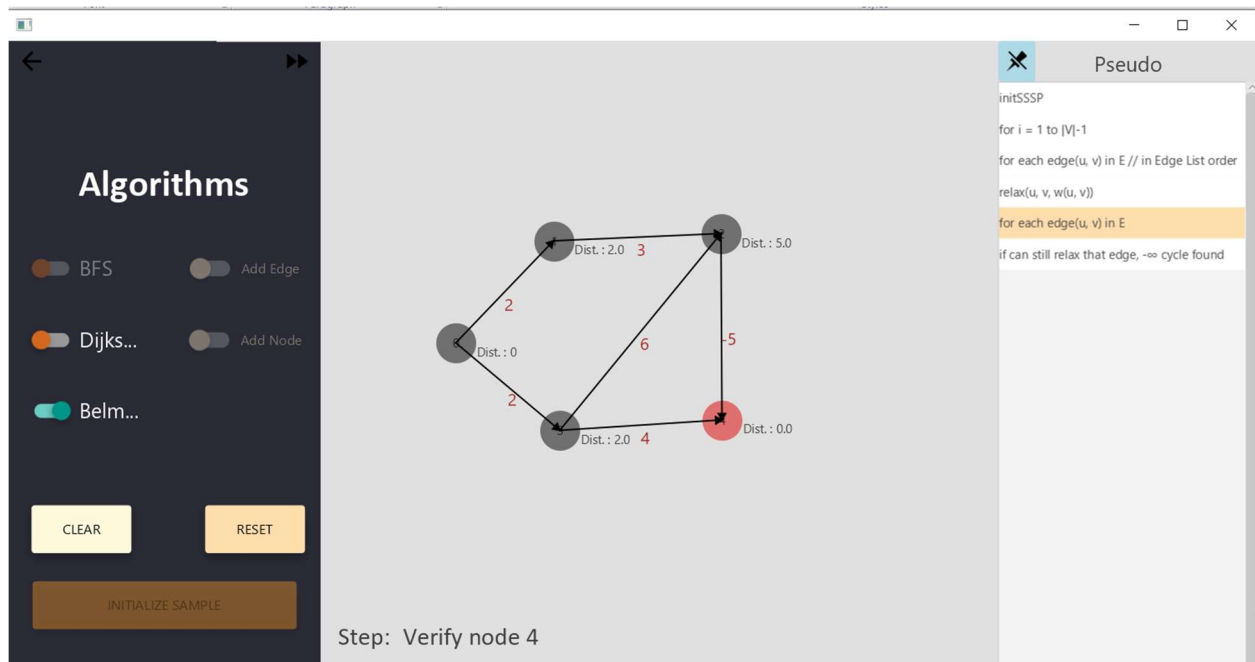


Belmond Ford:

+) When running:



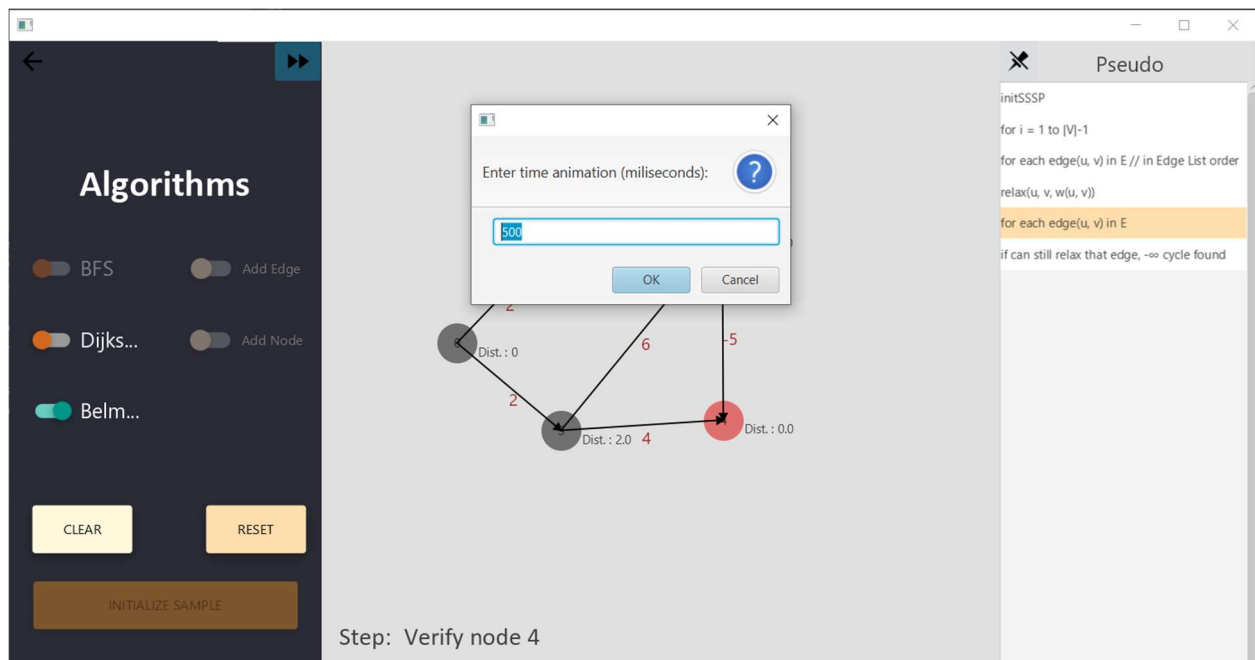
+ After finishing execution:



IV) Testing Program other functions for UX:

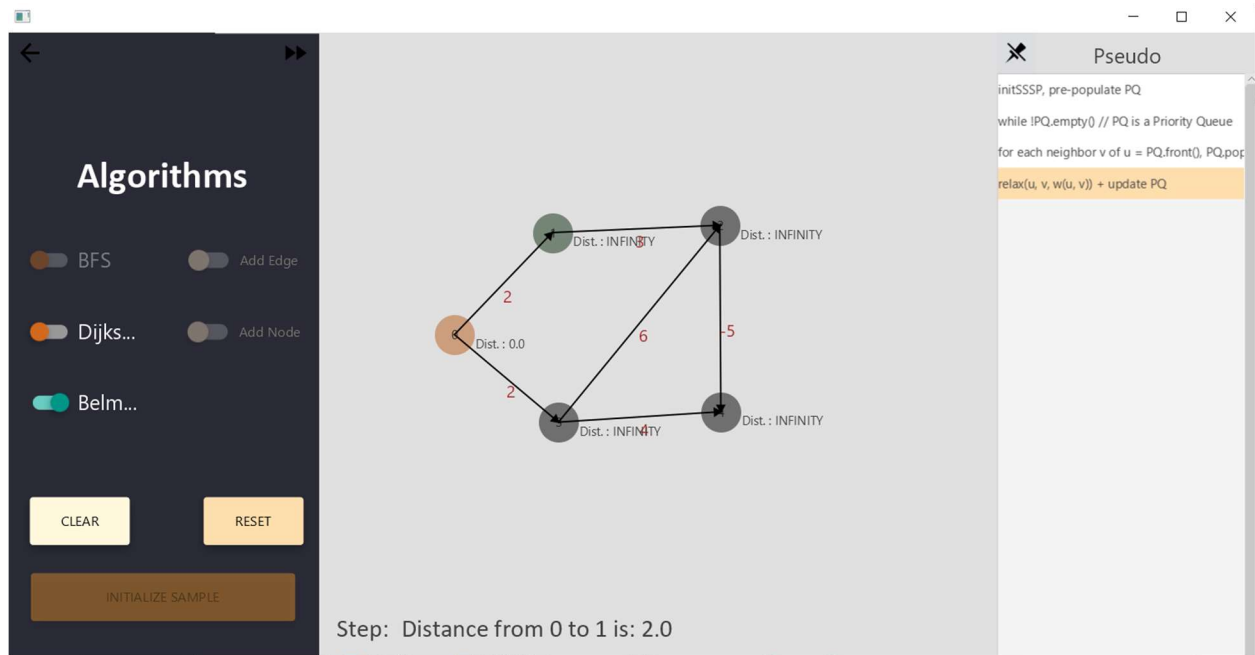
- Change time animation:

At default, time animation is 500 ms.

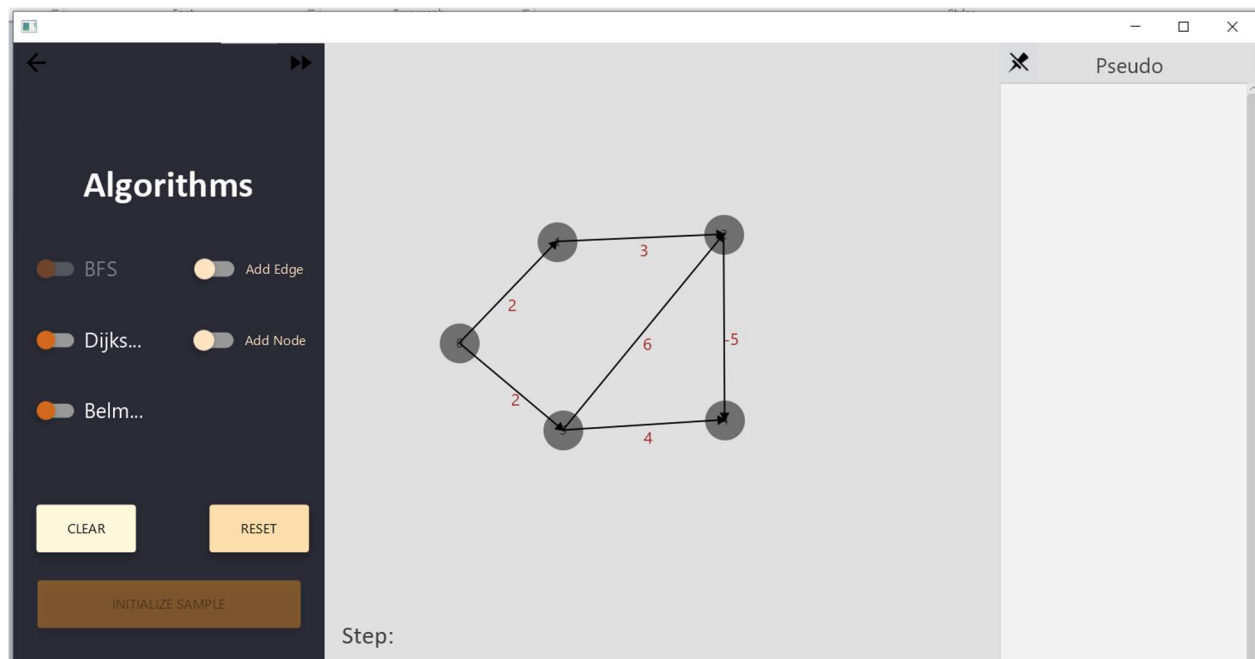


- Clear when you run algorithm (will stop all animations), then you can run it again:

+) Pressing button:

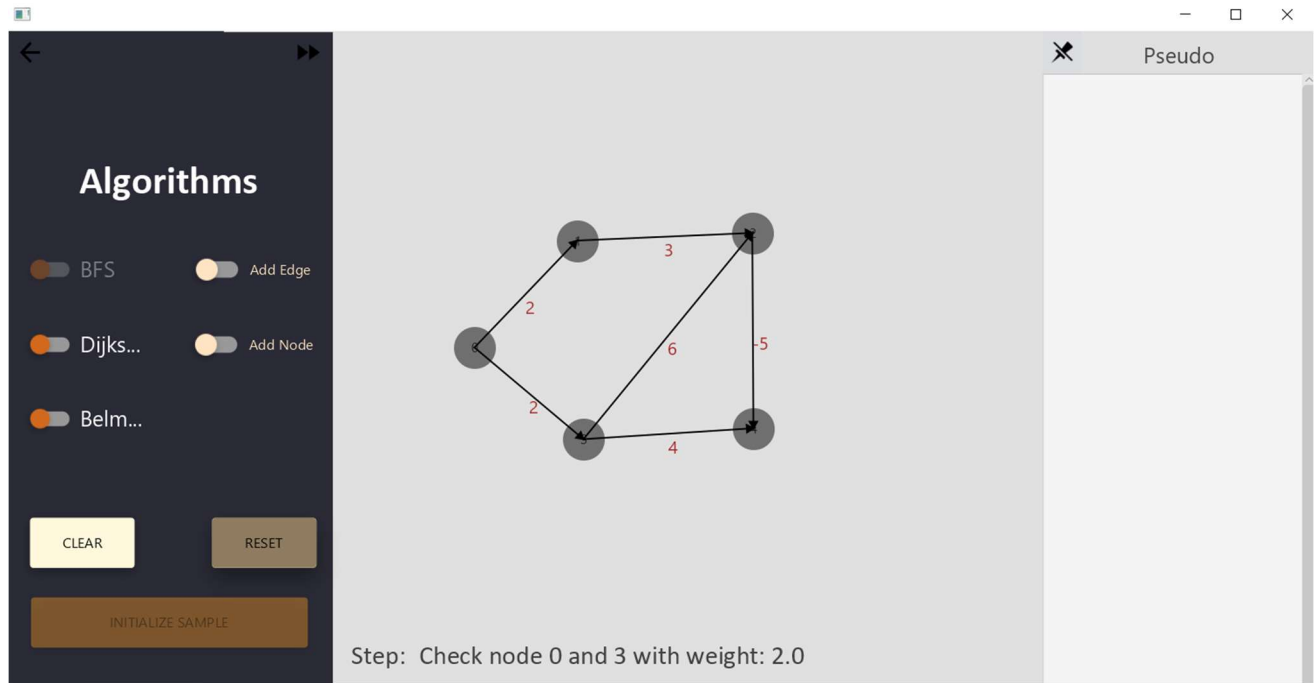


+) After pressed:

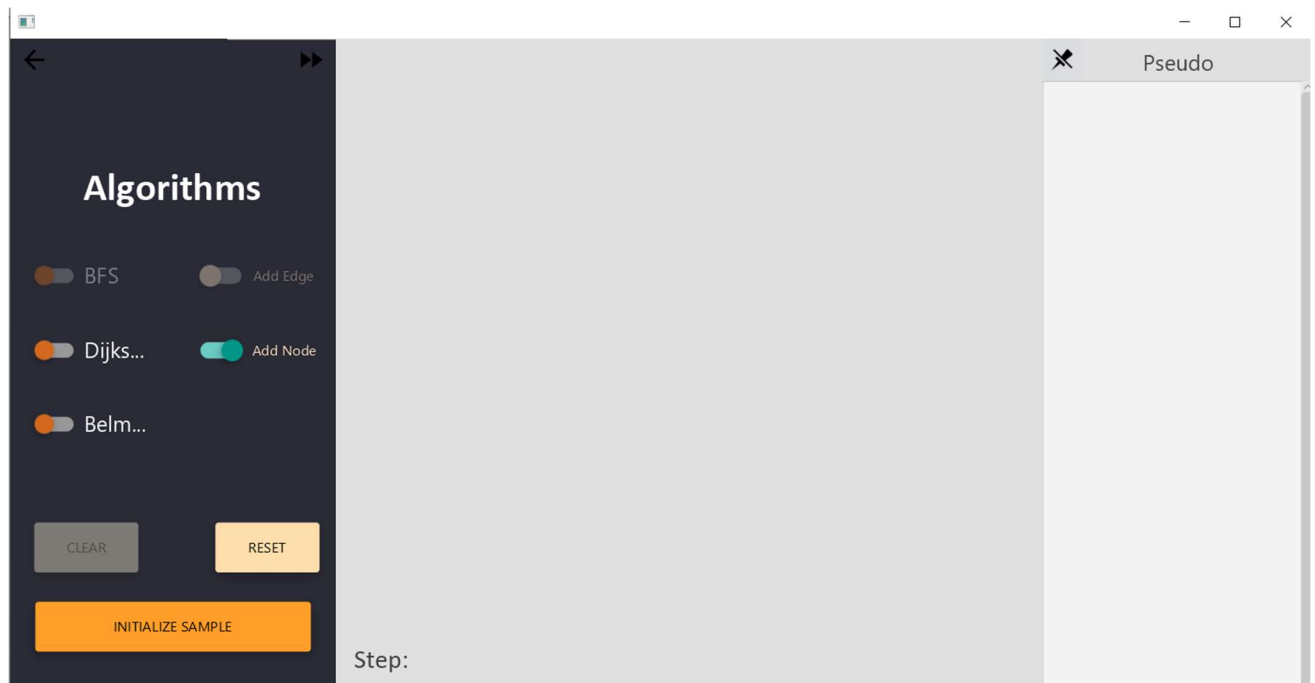


- Reset all nodes you have been add to create new Graph:

+) Pressing Reset button:



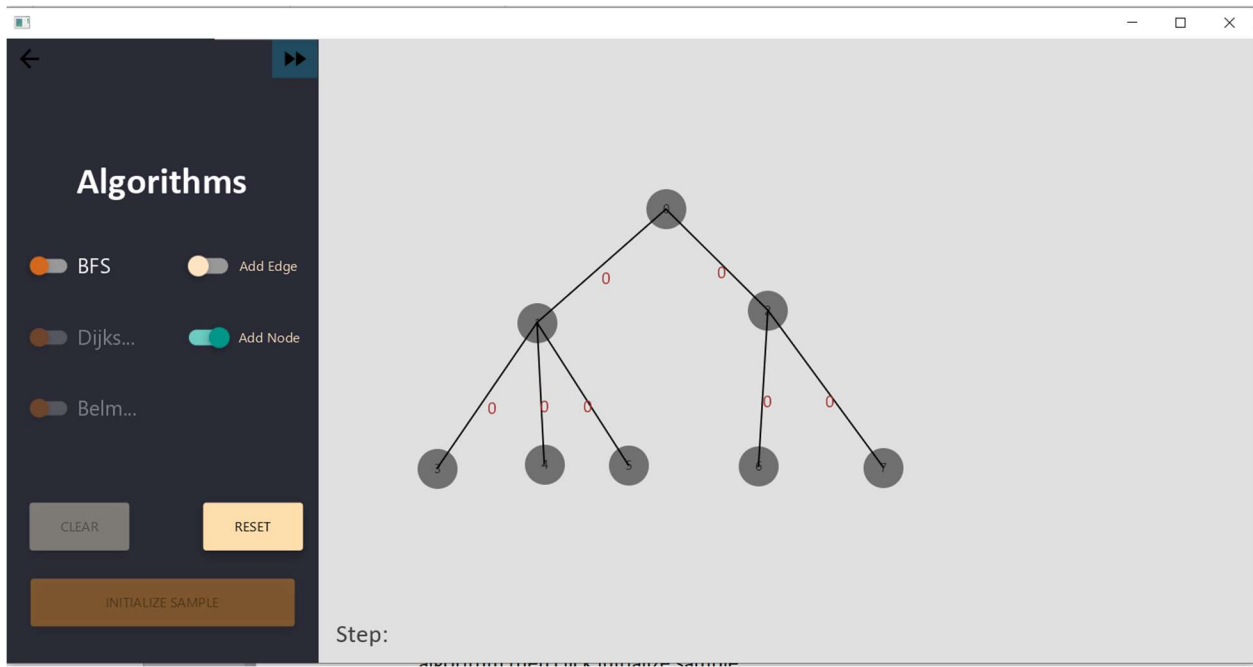
+) After pressed:



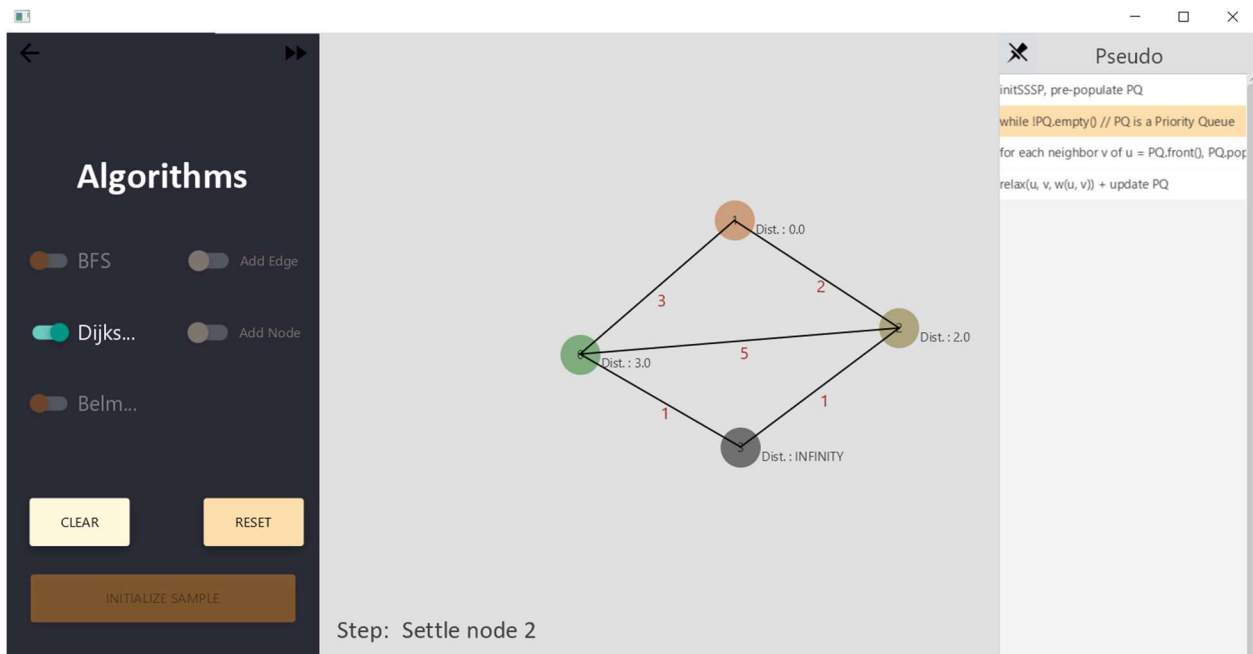
* After running algorithmmm, we must press clear button to run other algorithm

V) Test run time (in UI – not consider time running of animation)

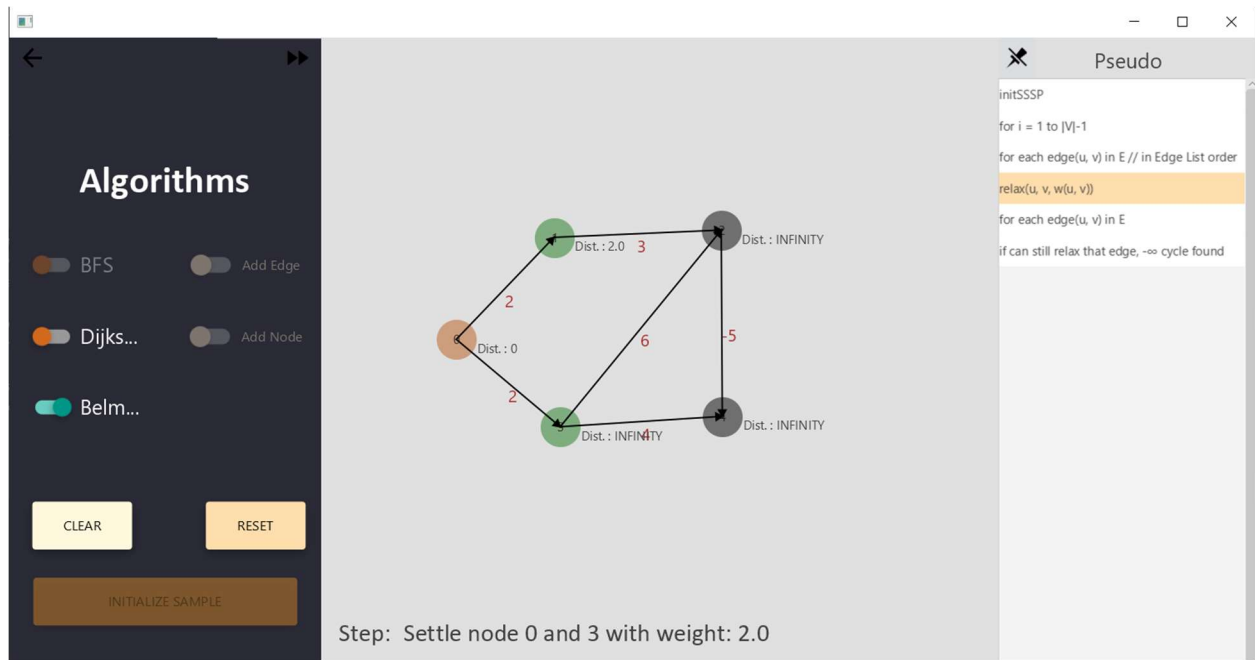
Graph 1:



Graph 2:



Graph 3:



	BFS	Dijkstra	Bellman-Ford
Graph 1 (undirected - unweighted)	0.4894 ms	0.5907 ms	1.0052 ms
Graph 2 (directed – weighted)		1.0891 ms	1.0947 ms
Graph 3 (directed – weighted)		1.0663 ms	1.1633 ms