Project 4: Sample Runs

Example 1: Displaying a Graph

Once you have the Graph class working, you can run the main program to test and debug. The main program is **`SimGUI`**, which can be compiled and run with the following commands if you are on Windows:

```
javac -cp .;../310libs.jar *.java
java -cp .;../310libs.jar SimGUI
```

or the following commands if you are on Linux/MacOS:

```
javac -cp .:../310libs.jar *.java
java -cp .:../310libs.jar SimGUI
```

Why is there extra stuff? The **`-cp`** is short for **`-classpath`** (meaning "where the class files can be found). The **`.;../310libs.jar`** or **`.:../310libs.jar`** has the following components: **`.`** the current directory, **`;`** or **`:`** the separator for Windows or Linux/MacOS respectively, **`310libs.jar`** the provided jar file which contains the library code for JUNG. By default, the **`.jar`** file is placed right above the folder with all .java files of P4. If you have moved it, you will need to use a different classpath to run **`SimGUI`**.

If you run the simulator with the above command, you will get a six-node graph with some random edges. Each time you hit "reset" you get another graph, but the same sequence of graphs is always generated (for your testing). However, the simulator can also be run with some additional optional parameters to get some more interesting results: The number of nodes, the likelihood that two nodes have an edge between them, the random seed for the graph generator. The next few tables give examples of what you can do.

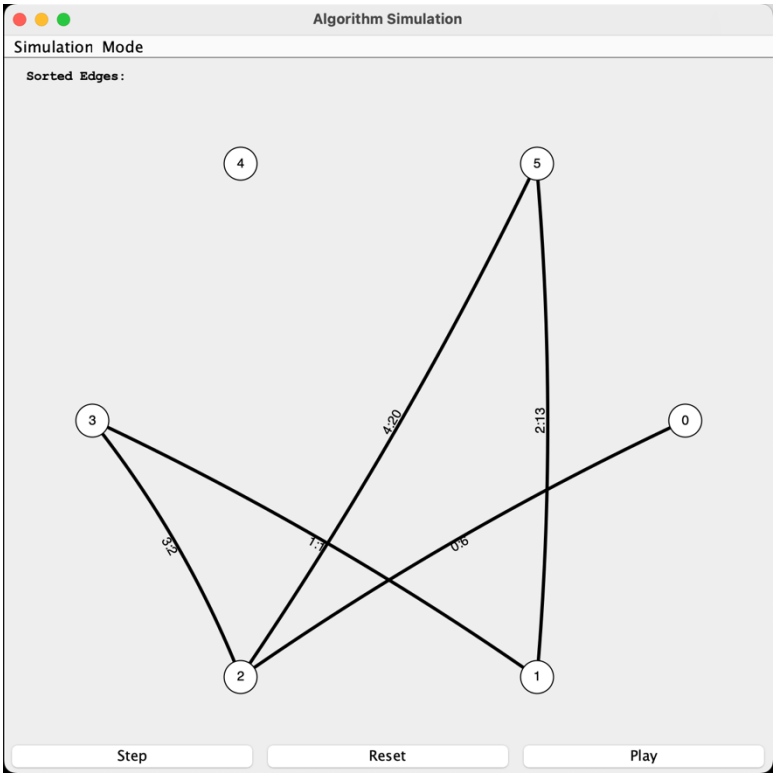
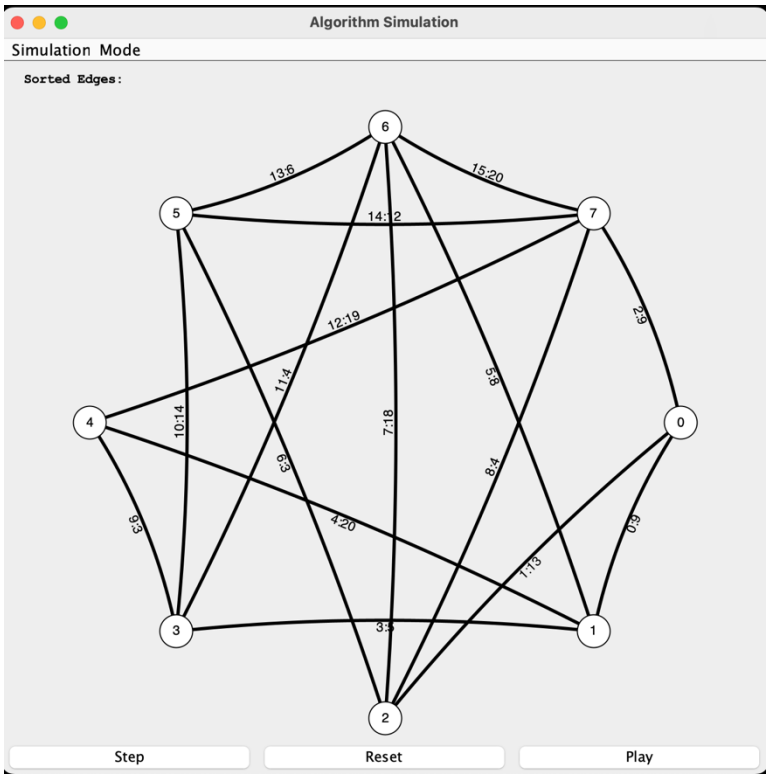
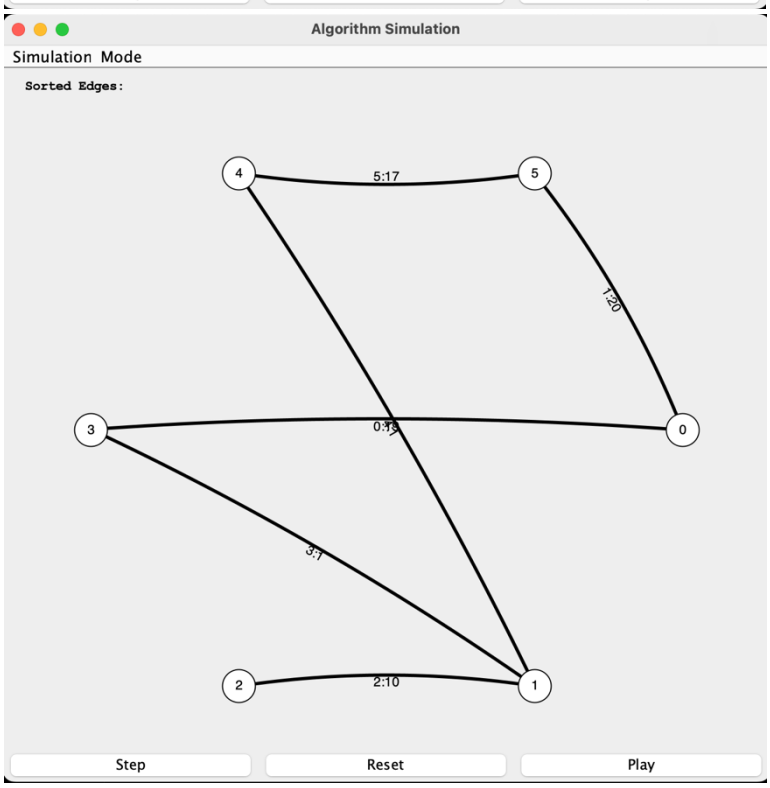
Image	Command +Explanation
	**`java -cp .;../310libs.jar SimGUI`** Generate a six-node graph, with connection probability of 0.5, and seed 0. NOTE: edge is labelled as ID: weight.

Image	Command +Explanation
	**`java -cp .;../310libs.jar SimGUI 8 0.8`** Generate a seven-node graph where nodes have an 80% chance of being connected, seed=0.
	**`java -cp .;../310libs.jar SimGUI 6 0.5 20`** Generate a different sequence of graphs with six nodes, 50% chance of connection but using seed 20.

Example 2: Adding nodes and edges to a graph

You'll want to test out adding multiple nodes and edges to make sure you've gotten out all the bugs.

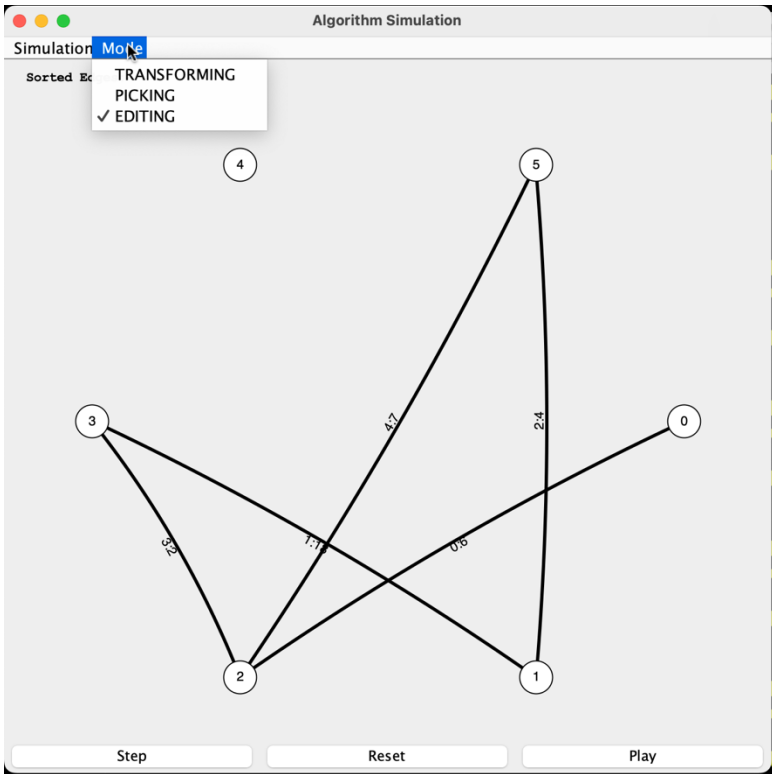
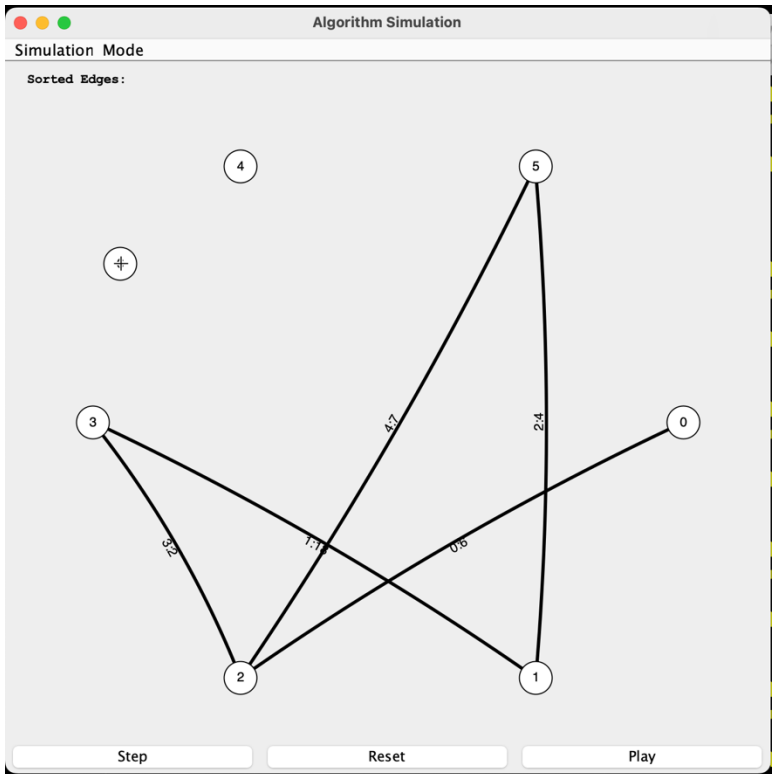
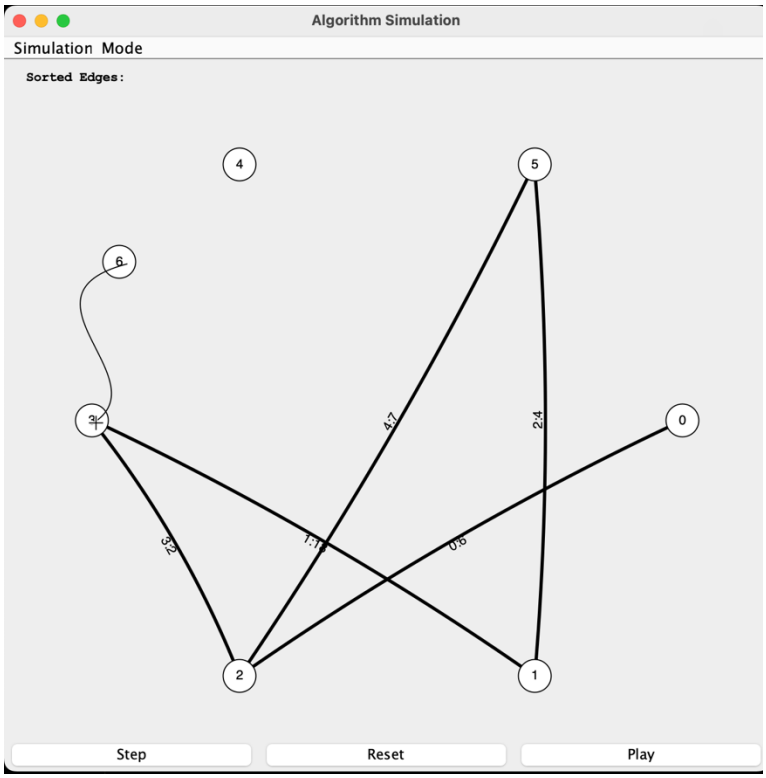
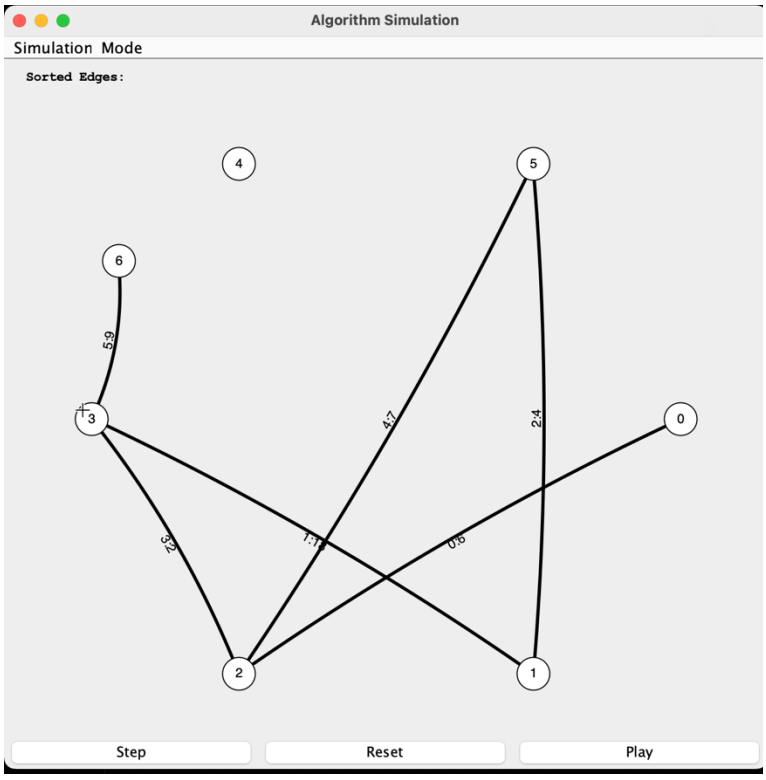
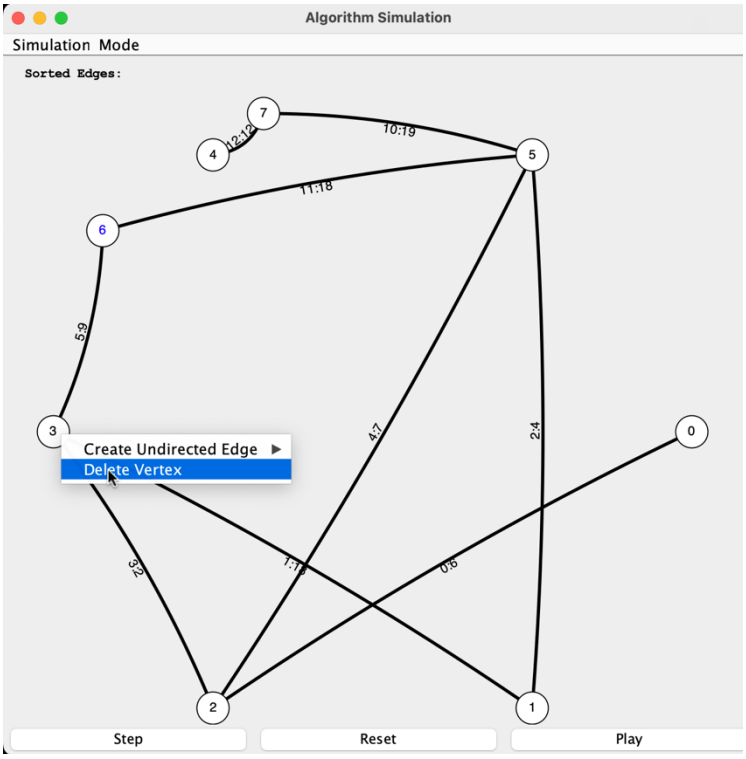
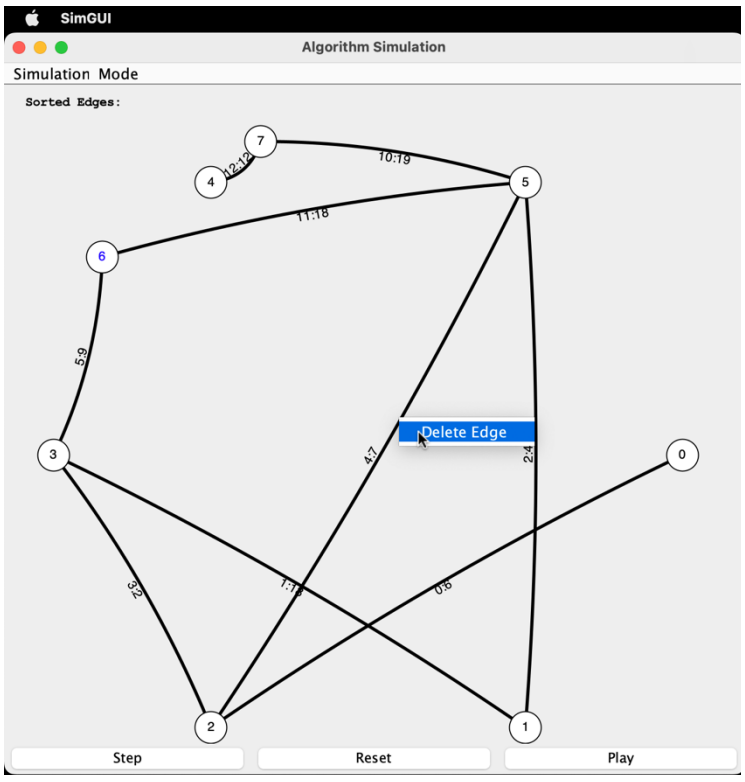
Image	Command +Explanation
	Select "Mode", then "Editing".
	Click anywhere on the graph surface to add a node.

Image	Command +Explanation
	Drag from one node to another node to add an edge. (No self-loops nor parallel edges allowed.)
	Release the mouse to see the edge added (with a random weight). (No self-loops nor parallel edges allowed.)

Example 3: Removing nodes and edges from a graph

Image	Command +Explanation
	Right click a node and select "Delete Vertex".
	Right click an edge and select "Delete Edge".

Example 4: Kruskal's Algorithm in Simulator

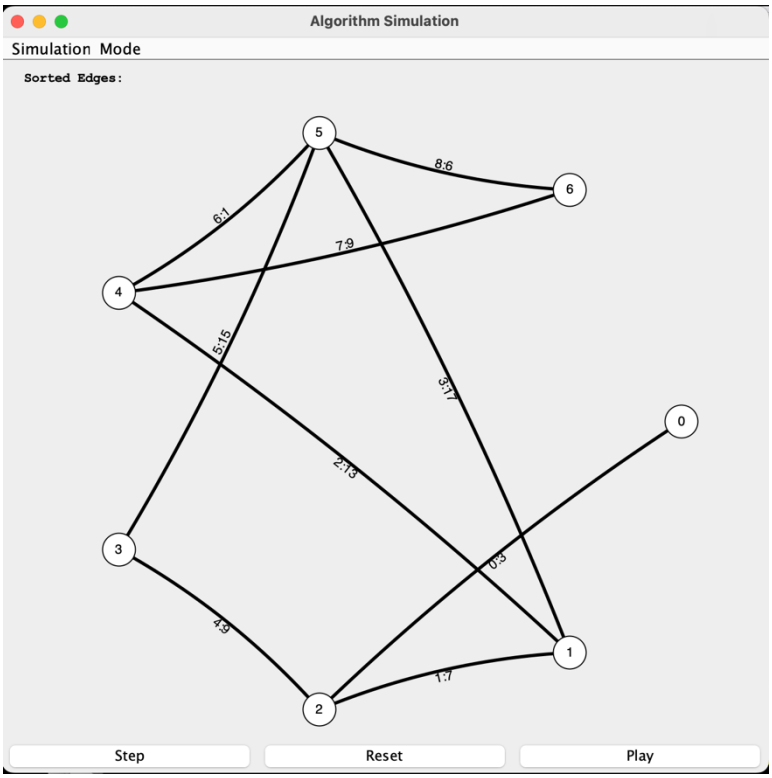
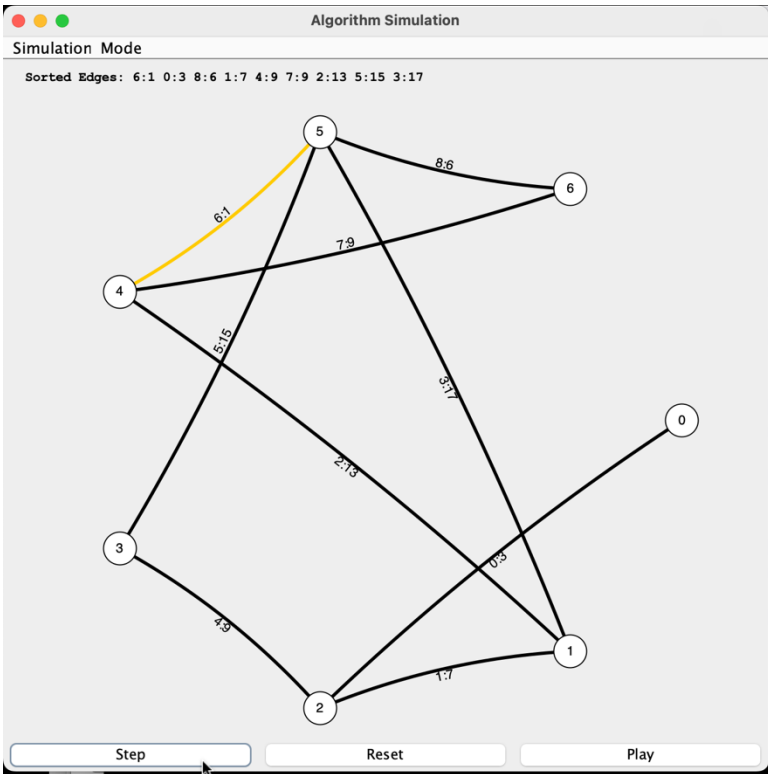
Image	Command +Explanation
	Set up your graph (empty sorted edges). Graph generated by command: `java -cp .:../310libs.jar SimGUI 7`
	Hit "Step": • all edges added to BST and displayed at the top of panel in the ascending order of edge weights (break the tie by lowest edge ID first). • edge of lowest weight highlighted (COLOR_HIGHLIGHT).

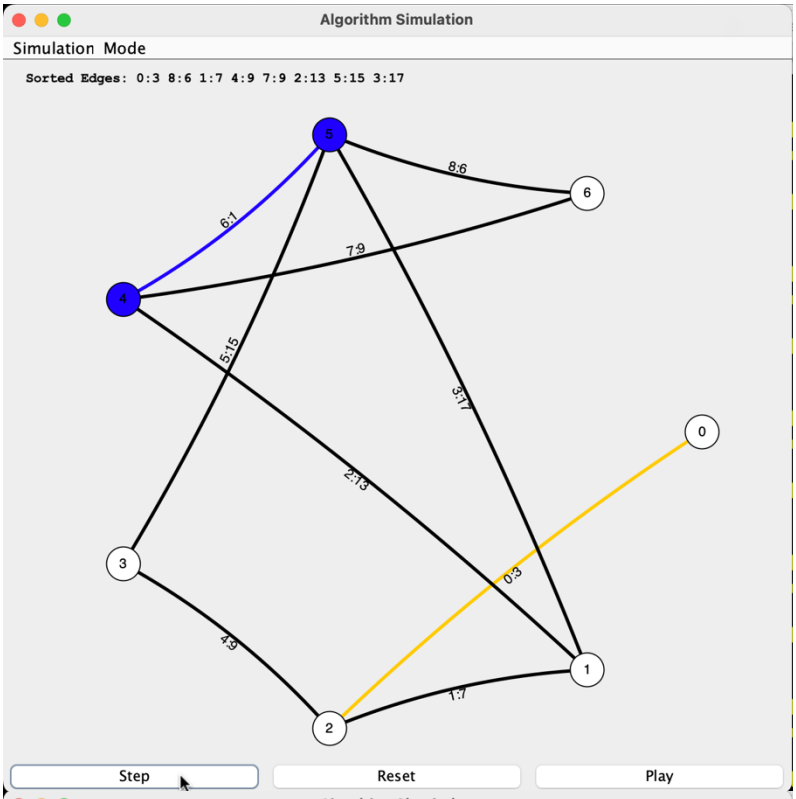
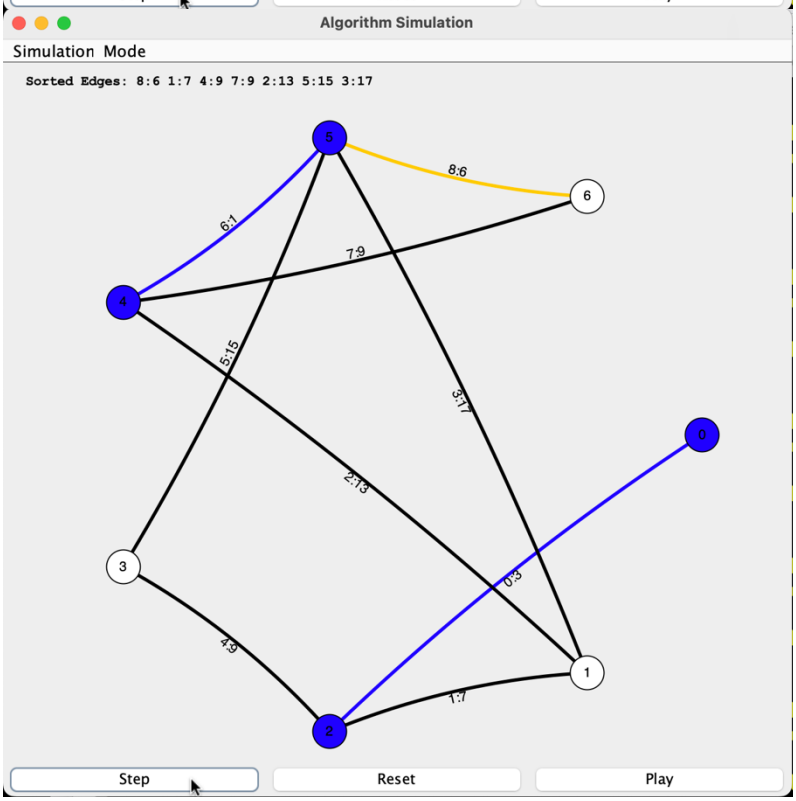
Image	Command +Explanation
	Hit "Step": • add min edge to MST if it does not create a cycle. • min edge and its endpoints change to COLOR_SELECTED indicating they are added into the MST. • min edge removed from "sorted edges". • next min edge highlighted.
	Hit "Step": same as above, repeat …

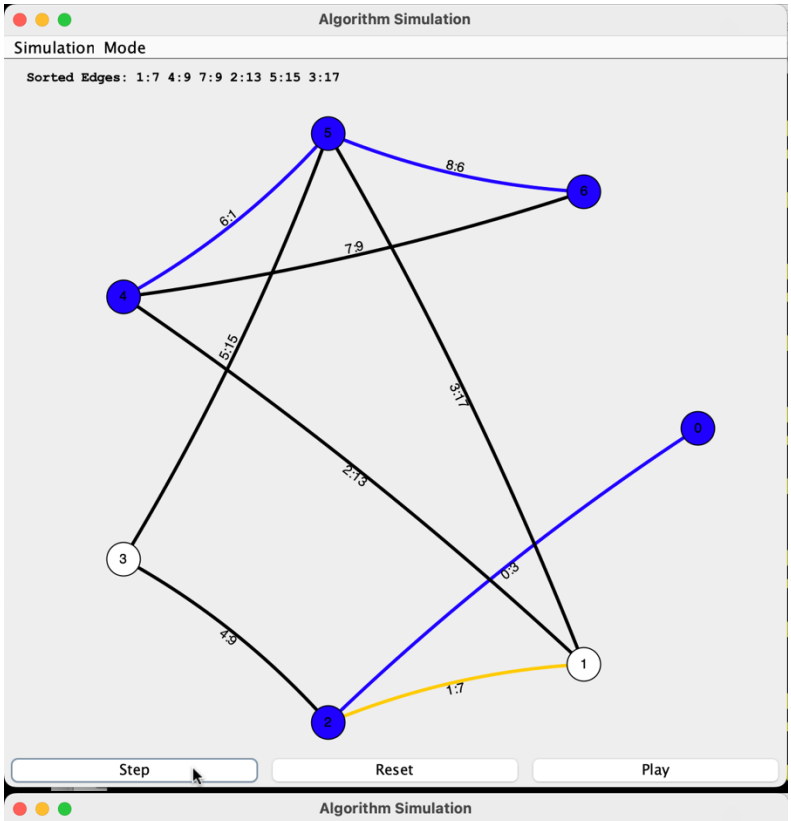
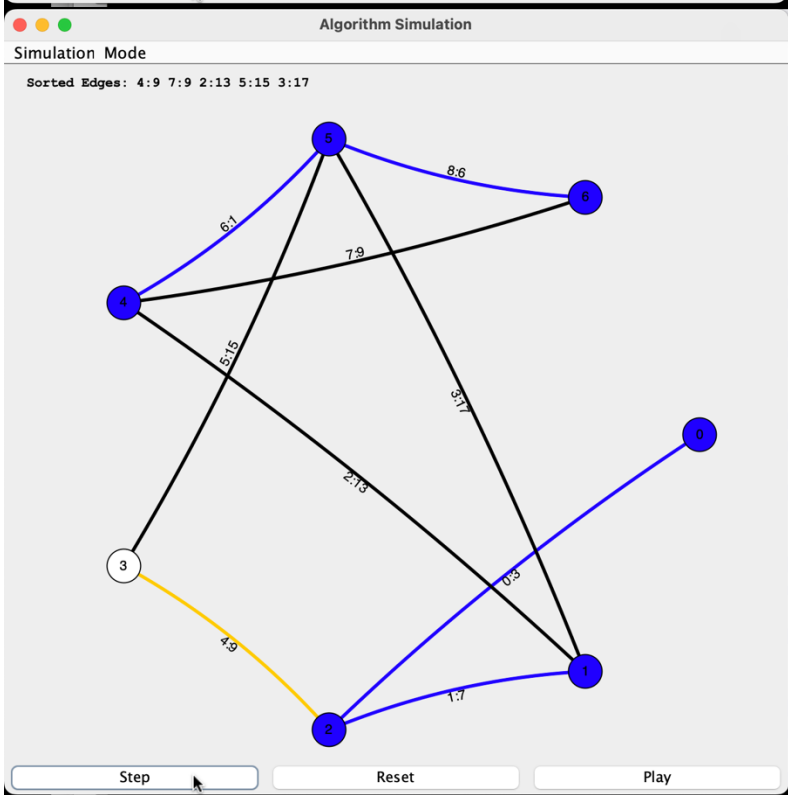
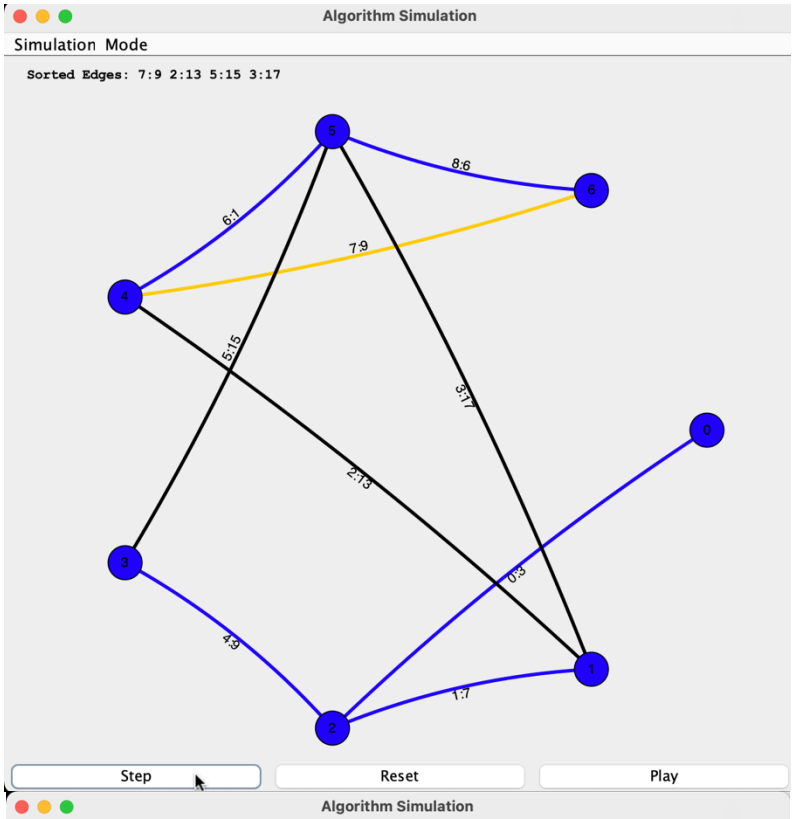
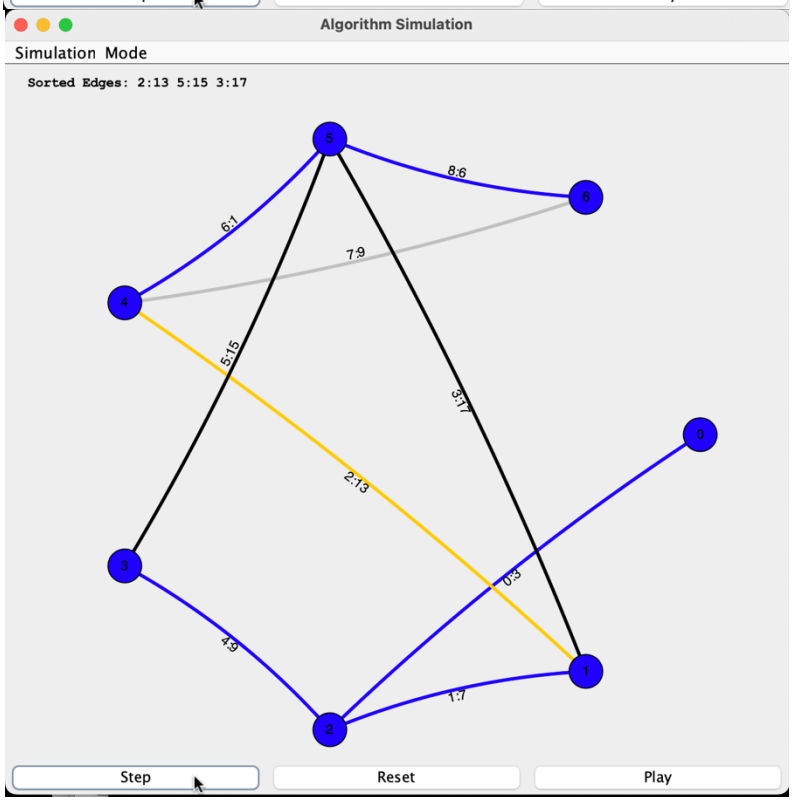
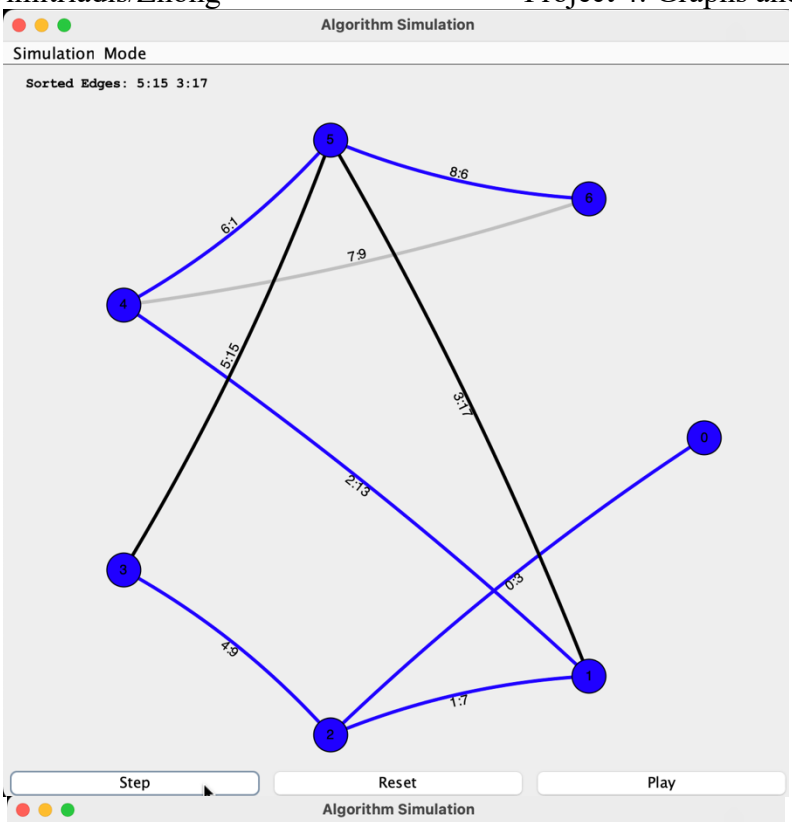
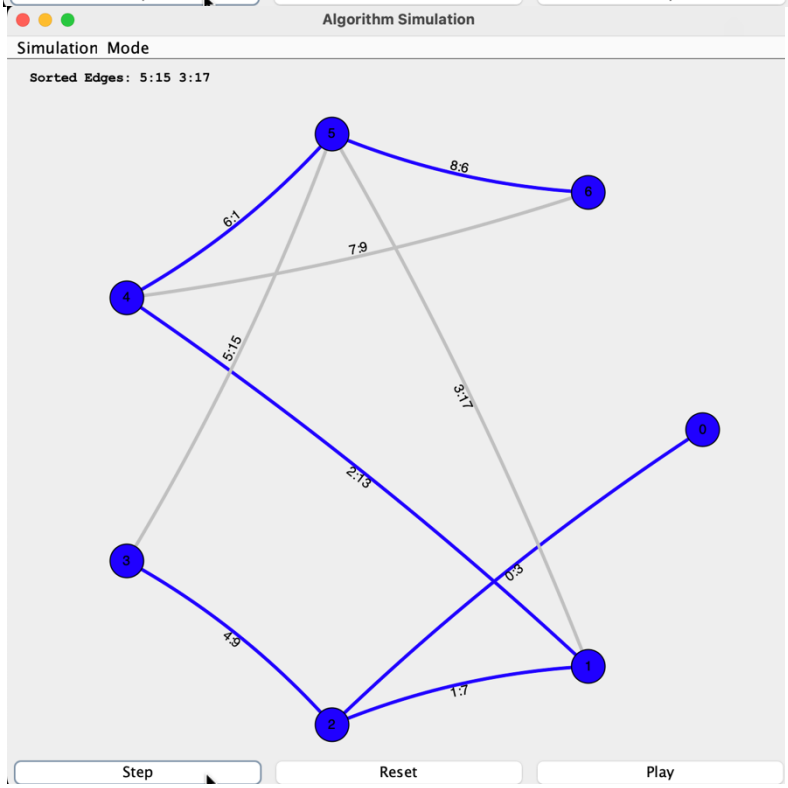
Image	Command +Explanation
	Hit "Step": same as above, repeat …
	Hit "Step": same as above, repeat …

Image	Command +Explanation
	Hit "Step": • Now all nodes are included in the subgraph, but they are not connected yet. • This highlighted edge should not be added, otherwise there will be a cycle with vertices 4, 5, and 6.
	Hit "Step": • Edge that should not be added into MST is greyed out (COLOR_INACTIVE_EDGE). • Move on to the next candidate edge.



Hit "Step":

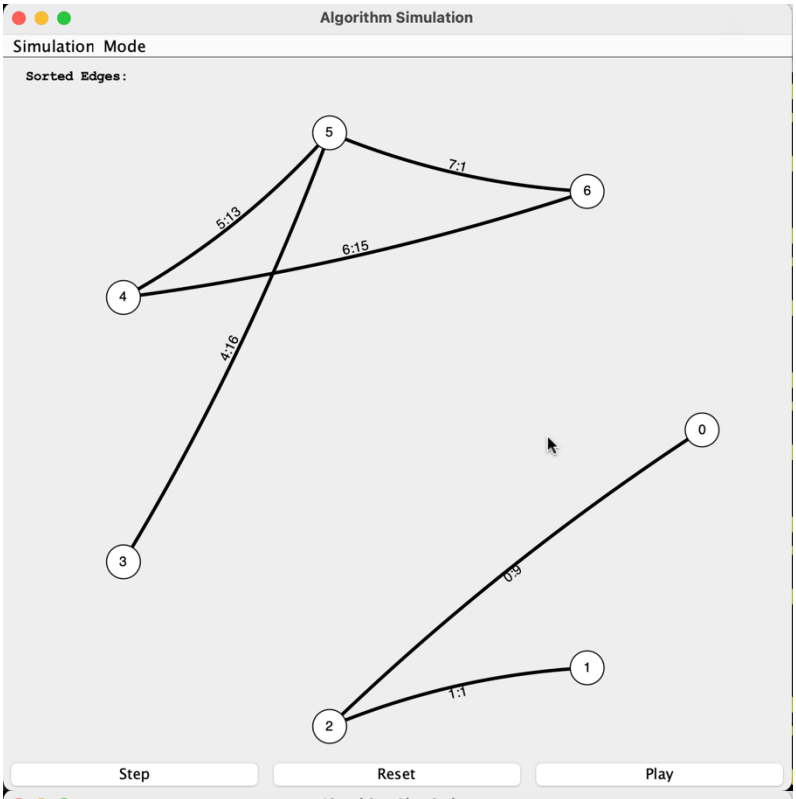
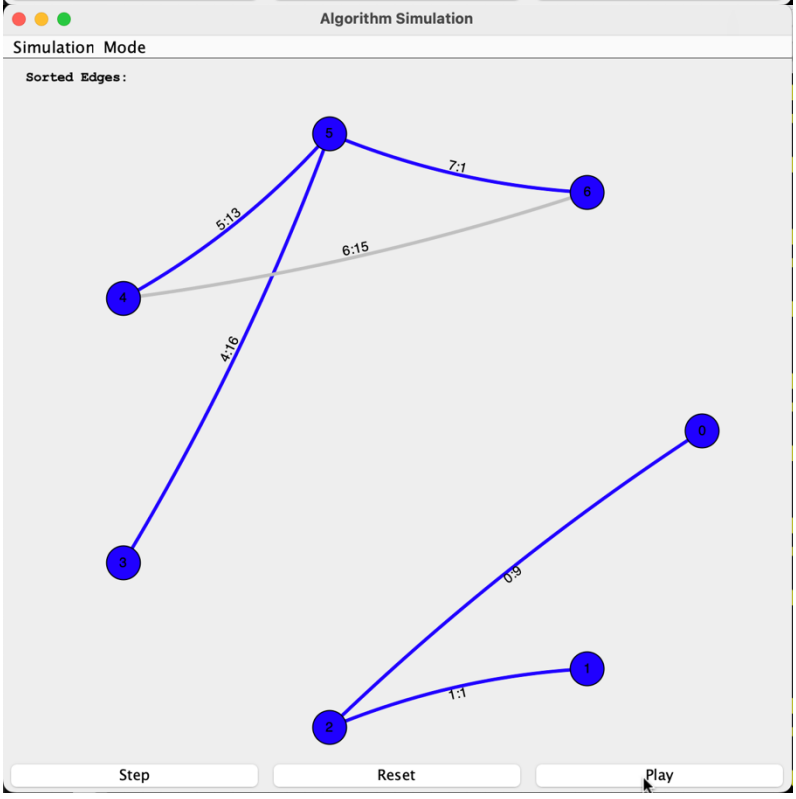
- Last edge added to complete the MST.
- No more edge needed: do not highlight another one.



Hit "Step":

- Edges not used will be greyed out at the end.

Example 5: Disconnected Graph Gets a Minimum Spanning Forest

Image	Command +Explanation
	Set up your graph (empty sorted edges). Graph generated by command: `java -cp .:../310libs.jar SimGUI 7 0.4` (then remove two edges)
	Hit "Step" repeated until done: result subgraph is a minimum spanning forest. NOTE: edge selection may vary if the graph you generated have different edge weights.

Example 6: Disconnected Graph with Orphan Nodes

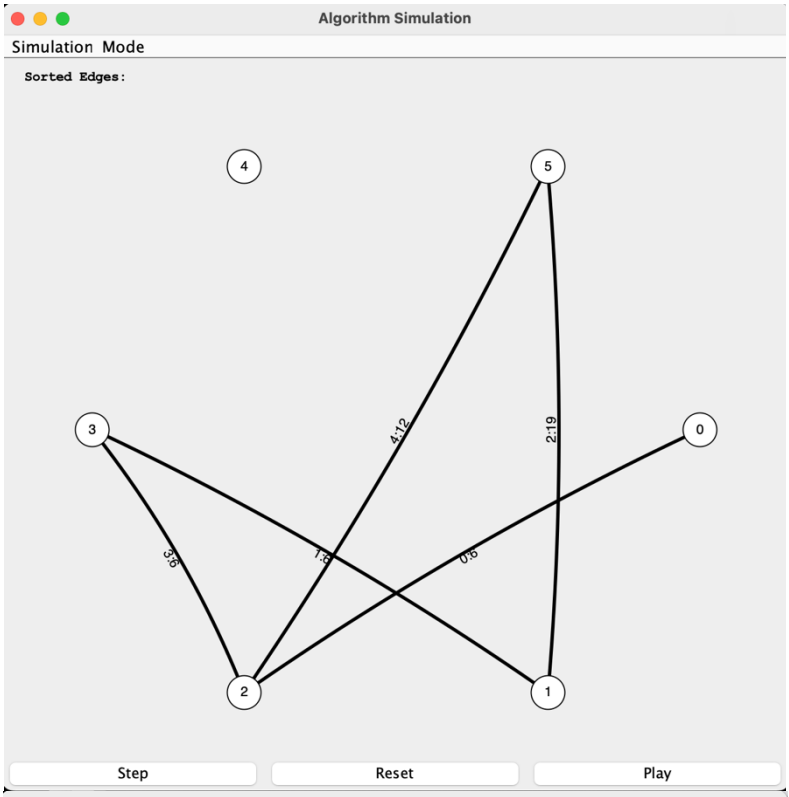
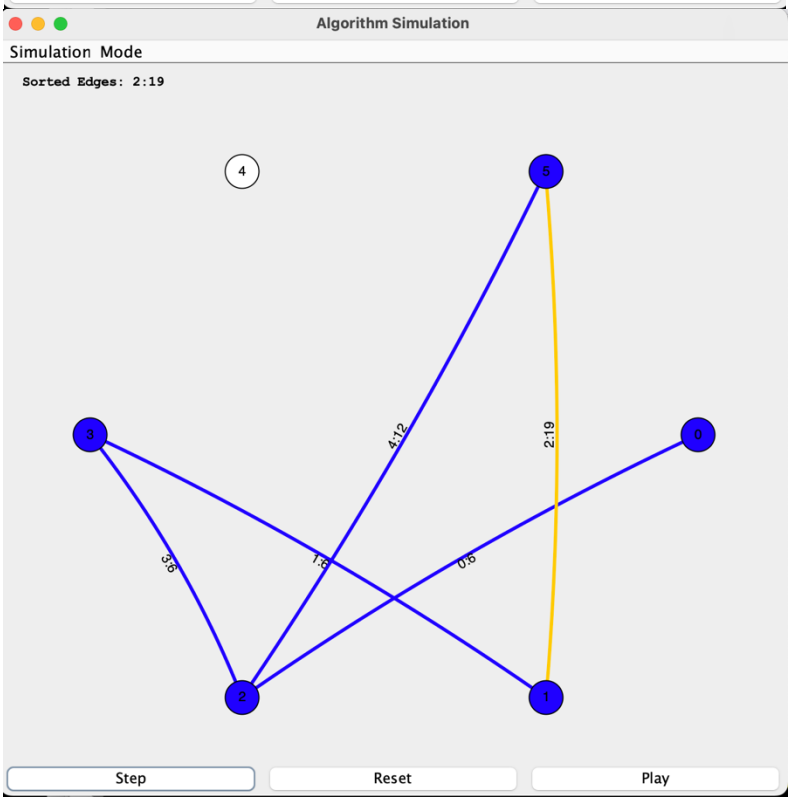
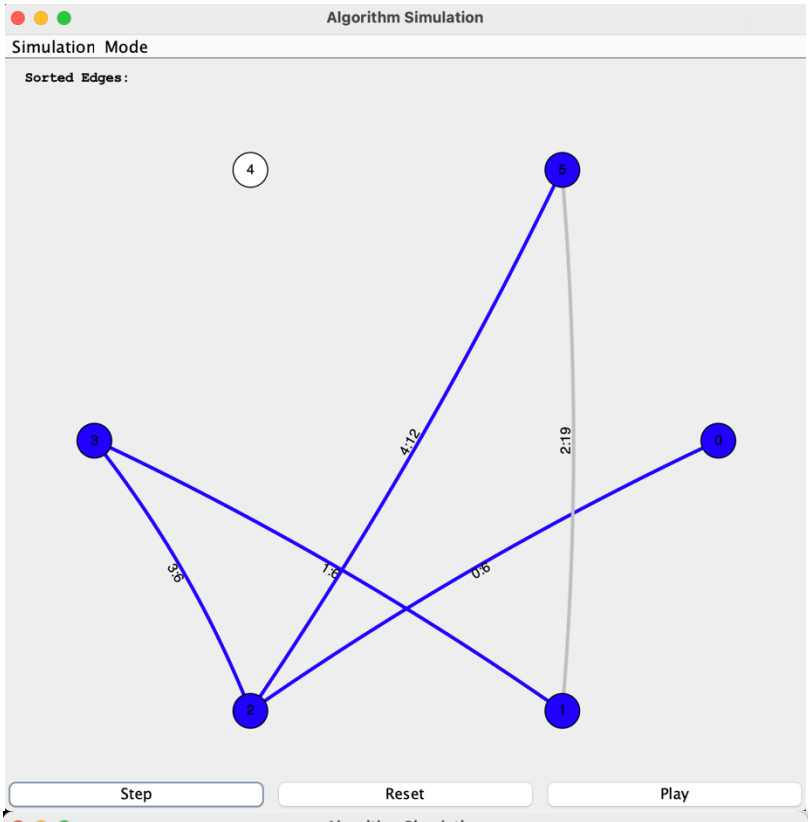
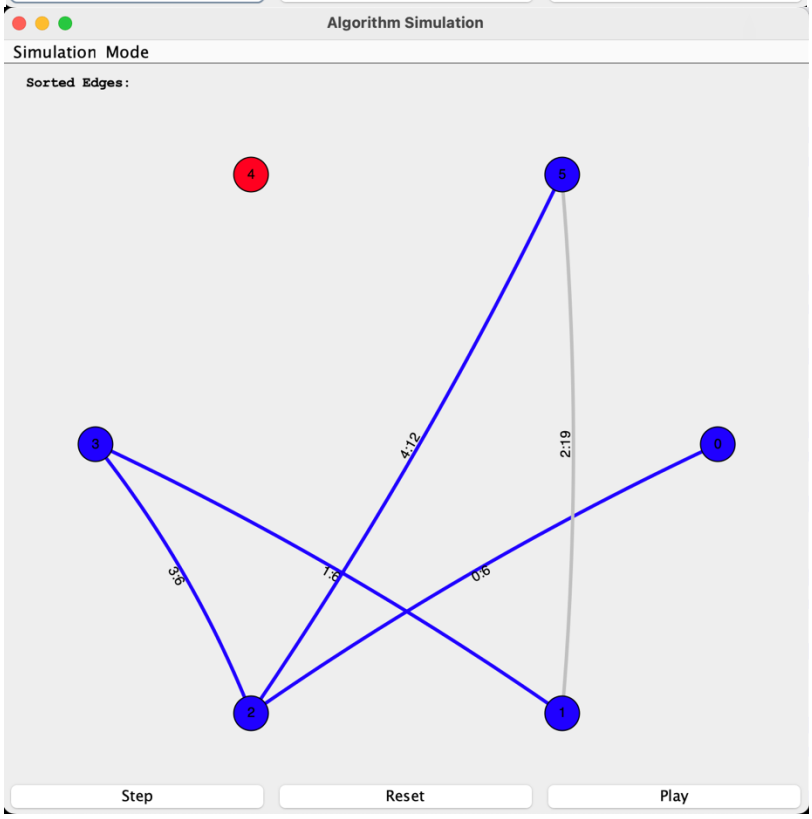
Image	Command +Explanation
	Set up your graph (empty sorted edges). Graph generated by command: `java -cp .:../310libs.jar SimGUI`
	Hit “Step” multiple times until the last edge is highlighted.

Image	Command +Explanation
	Hit "Step": • Edge should not be included since it will create a cycle. • No more edges left but node 4 is not included yet.
	Hit "Step": • In the final clean up, mark each unmarked vertex using COLOR_WARNING.