User Manual

Sponsor: Dr. Matthias Stallmann
CSC 492 Team 43

Christina Albores
Vitesh Kambara
Julian Madrigal
Neha Ramesh
Minghong Zou

North Carolina State University
Department of Computer Science

04/22/2024

Table of Contents:

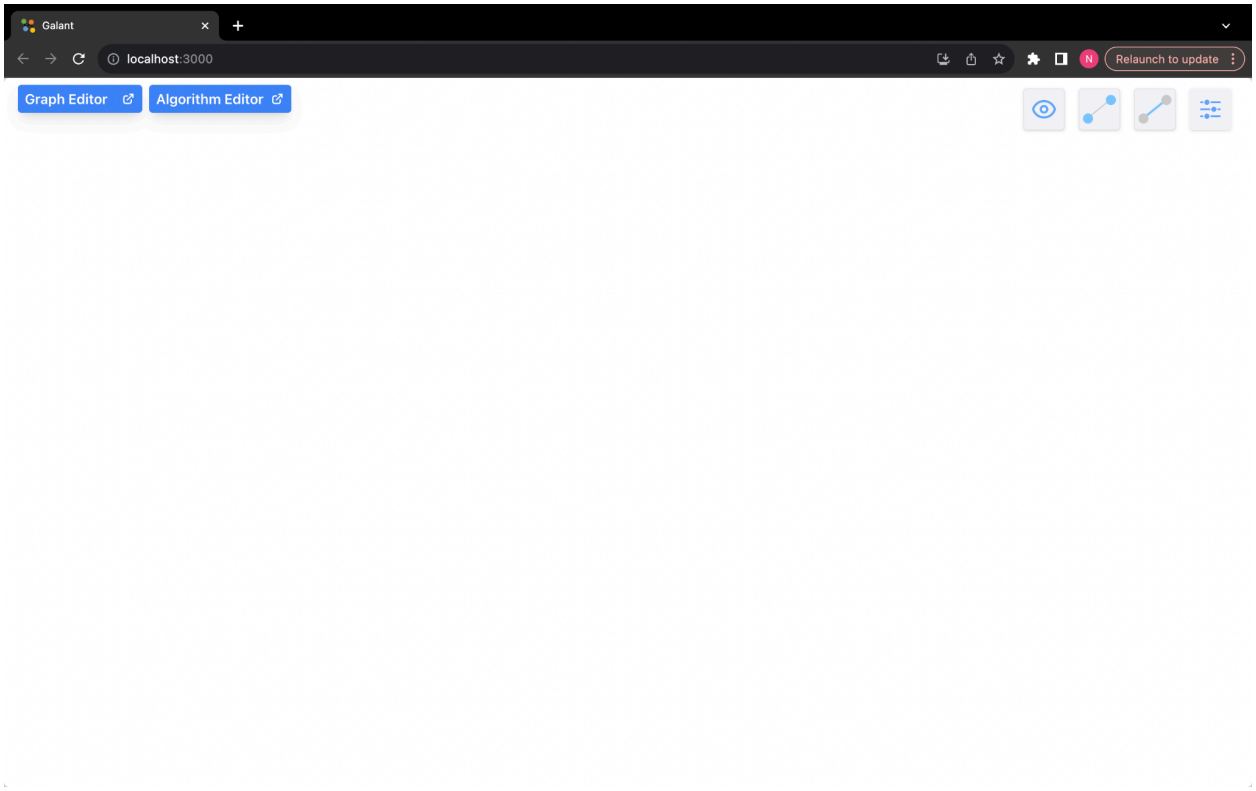
Introduction

Galant-JS is a web application that facilitates visualizing graph algorithms for research and classroom instruction. This is a manual on how to use the Galant-JS application.

Application usage

Start the Web Application

1. Navigate to the website https://galant.csc.ncsu.edu/ in a compatible browser (Chrome/Firefox)

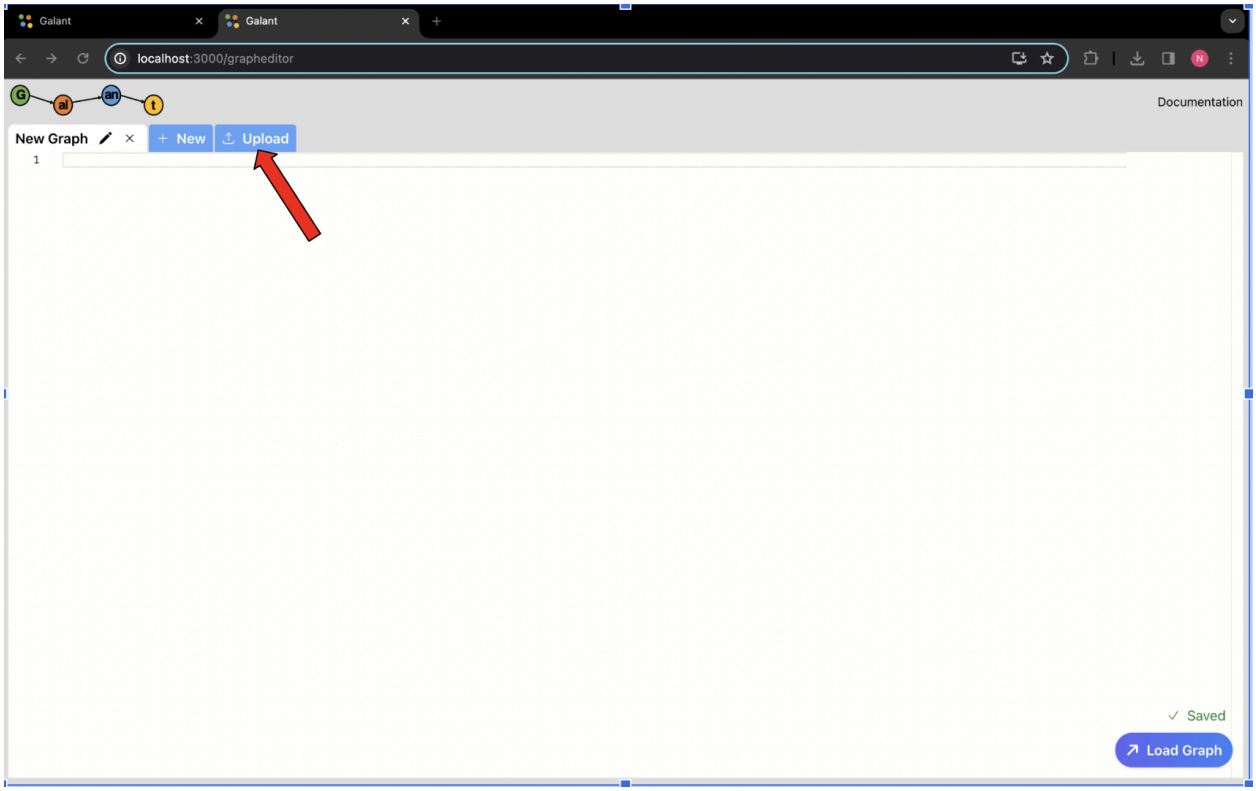


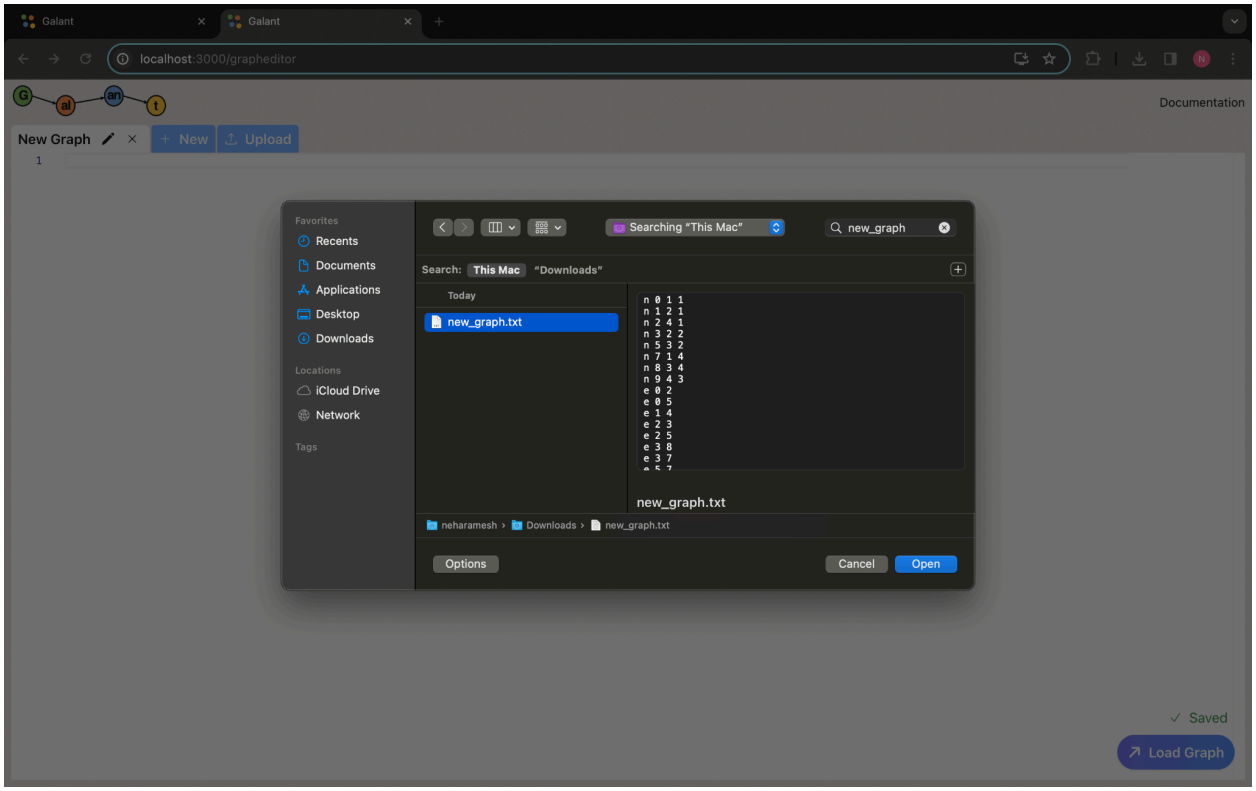
2. Remove any pop-up blockers that might be installed on the web browser.

Upload a Graph

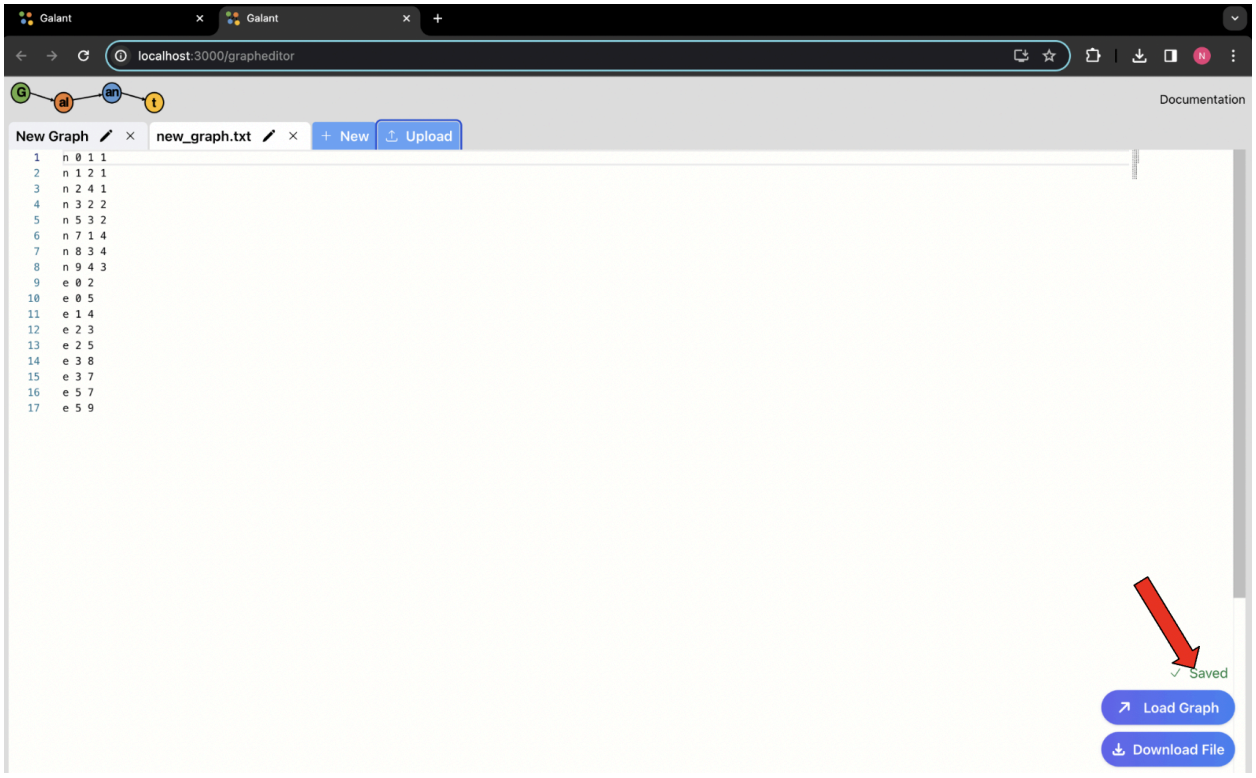
1. Load the application on a webpage
2. Click on the "Graph Editor" button on the menu bar

3. Click on the “Upload” button in the window that pops up. Choose the graph that you want to upload from the file explorer window.




Galant
Galant
localhost:3000/grapheditor
Documentation
New Graph
New
Upload
1
Favorites
Recents
Documents
Applications
Desktop
Downloads
Locations
iCloud Drive
Network
Tags
Searching "This Mac"
new_graph
Search: This Mac "Downloads"
Today
new_graph.txt
n 0 1 1
n 1 2 1
n 2 4 1
n 3 2 2
n 5 3 2
n 7 1 4
n 8 3 4
n 9 4 3
e 0 2
e 0 5
e 1 4
e 2 3
e 2 5
e 3 8
e 3 7
new_graph.txt
neharamesh › Downloads › new_graph.txt
Options
Cancel
Open
Saved
Load Graph

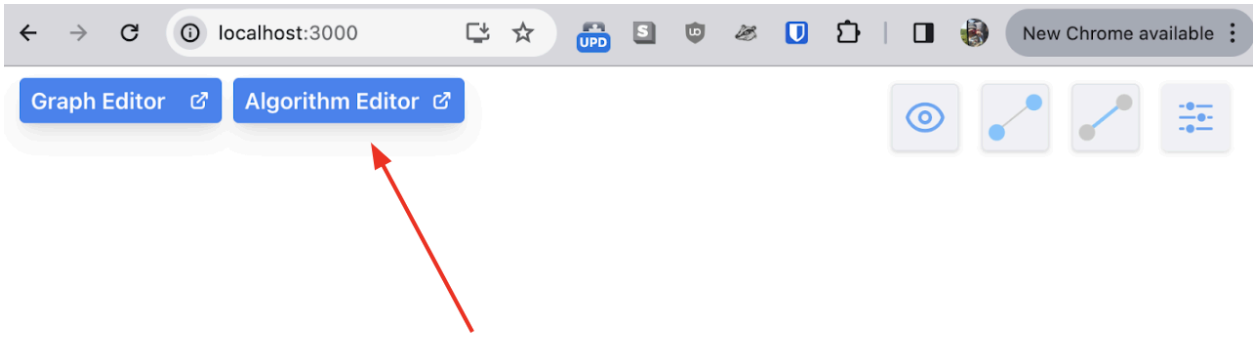
4. The uploaded graph should be loaded in a new tab with the graph name.



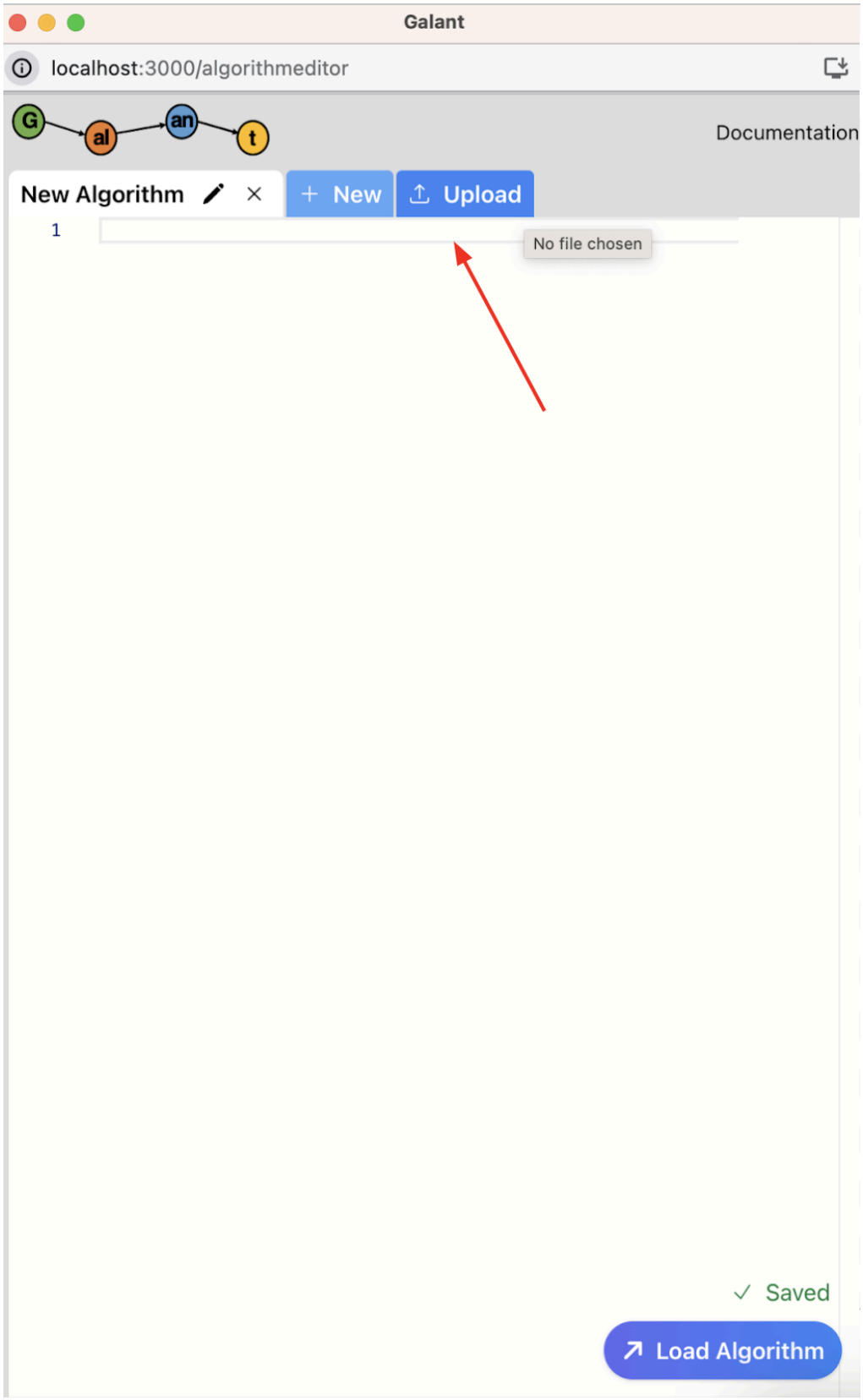
Upload an Algorithm

1. Load the application on a webpage

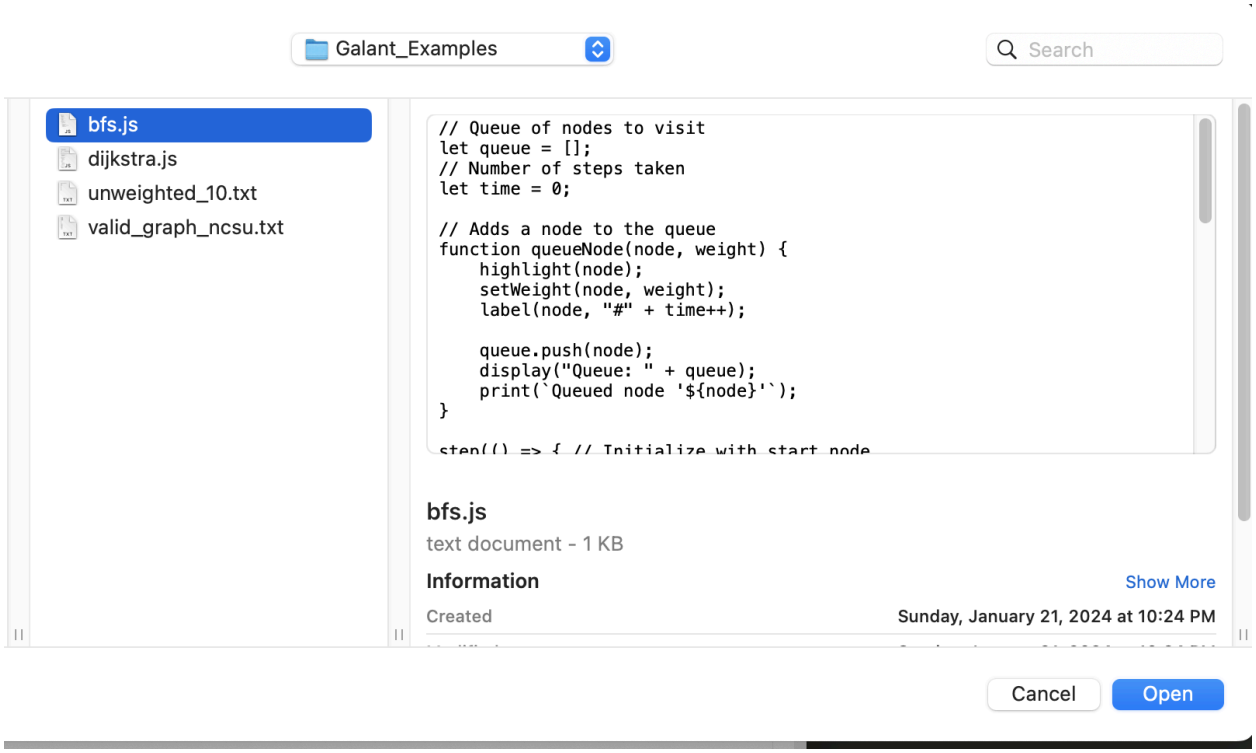
2. Click on the "Algorithm Editor" button on the main view page.



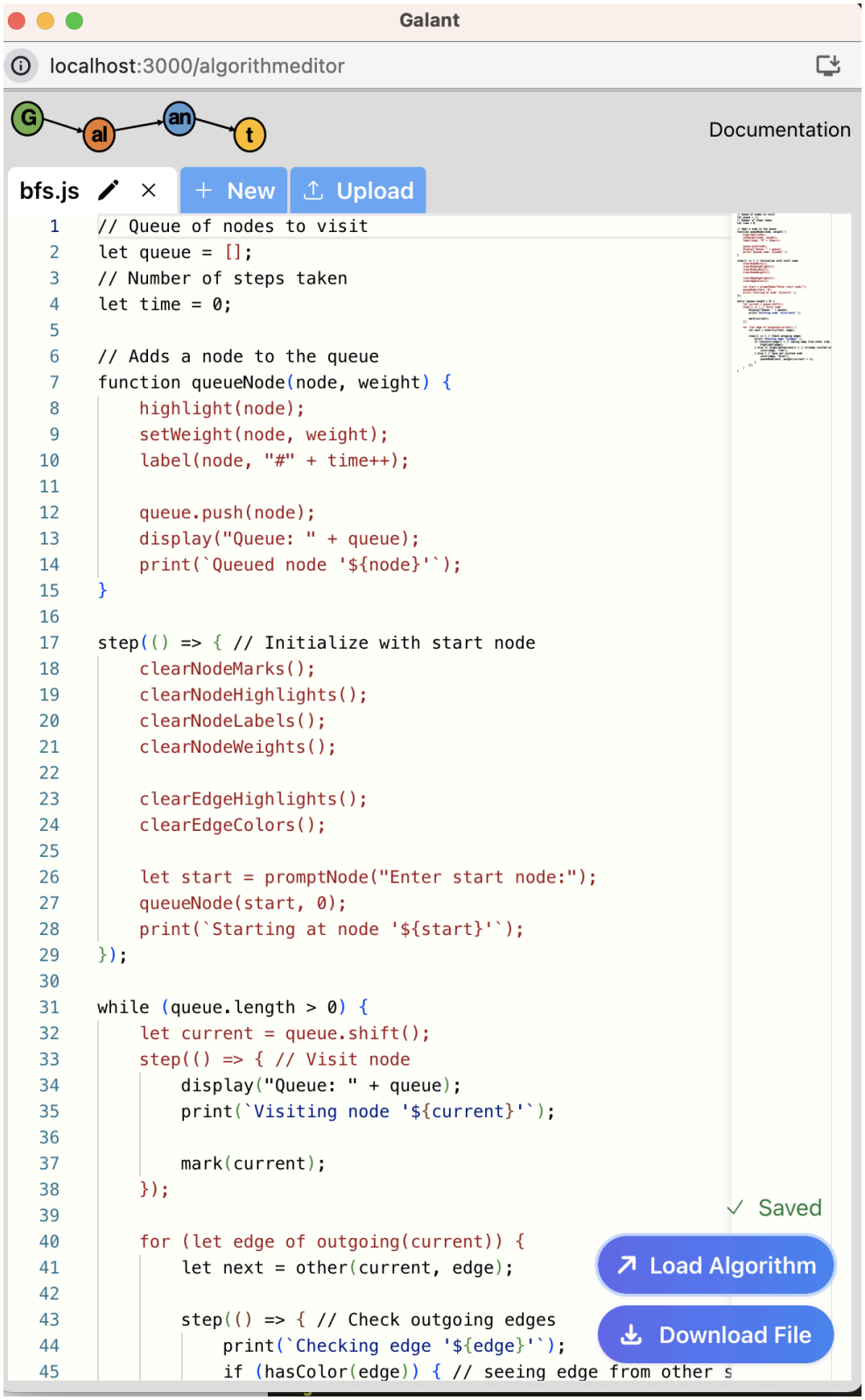
3. In the Algorithm Editor pop-up window, click on the “Upload Algorithm” button.



4. Choose the algorithm that you want to upload from the file explorer window.



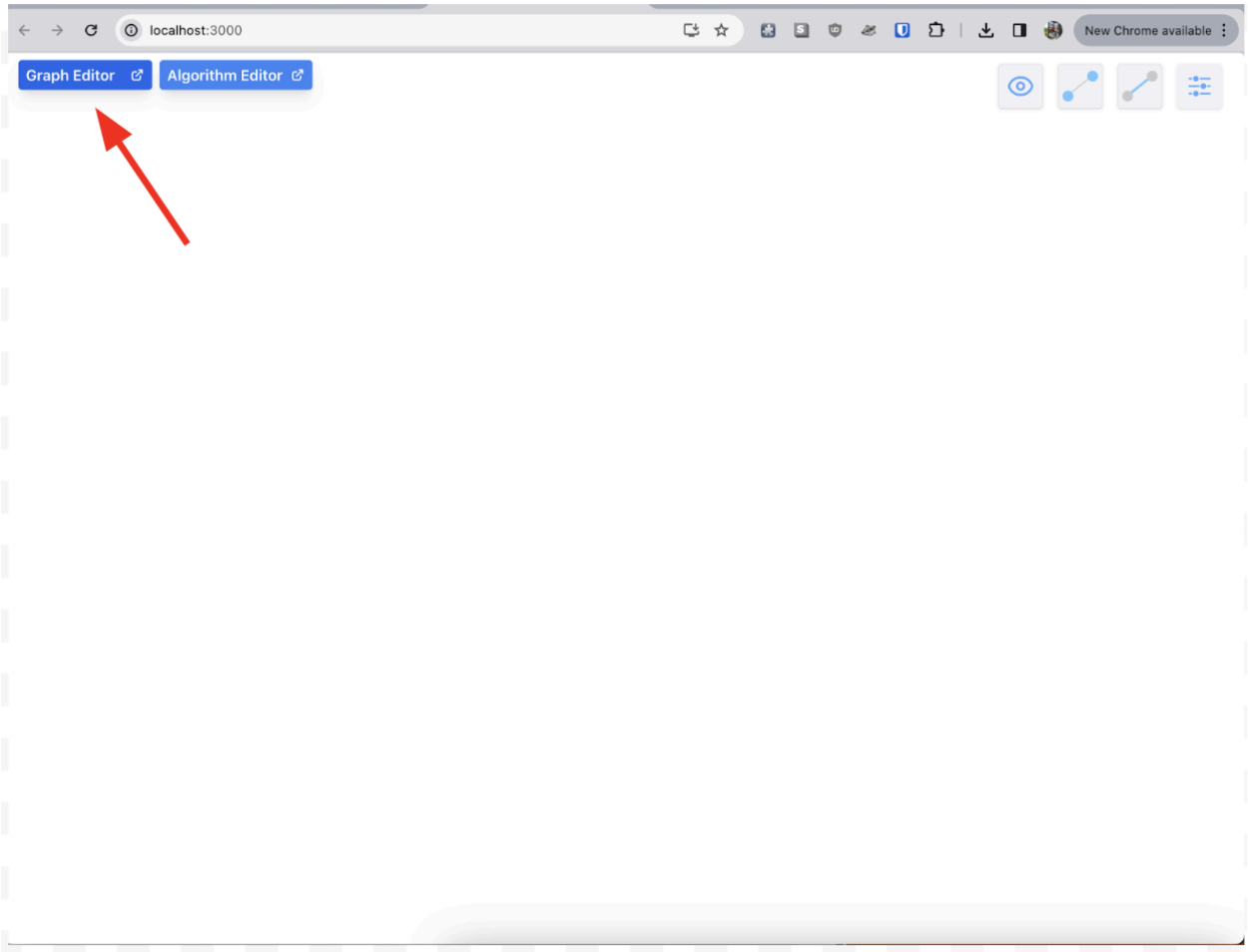
5. The algorithm editor should contain the text of your uploaded algorithm file.



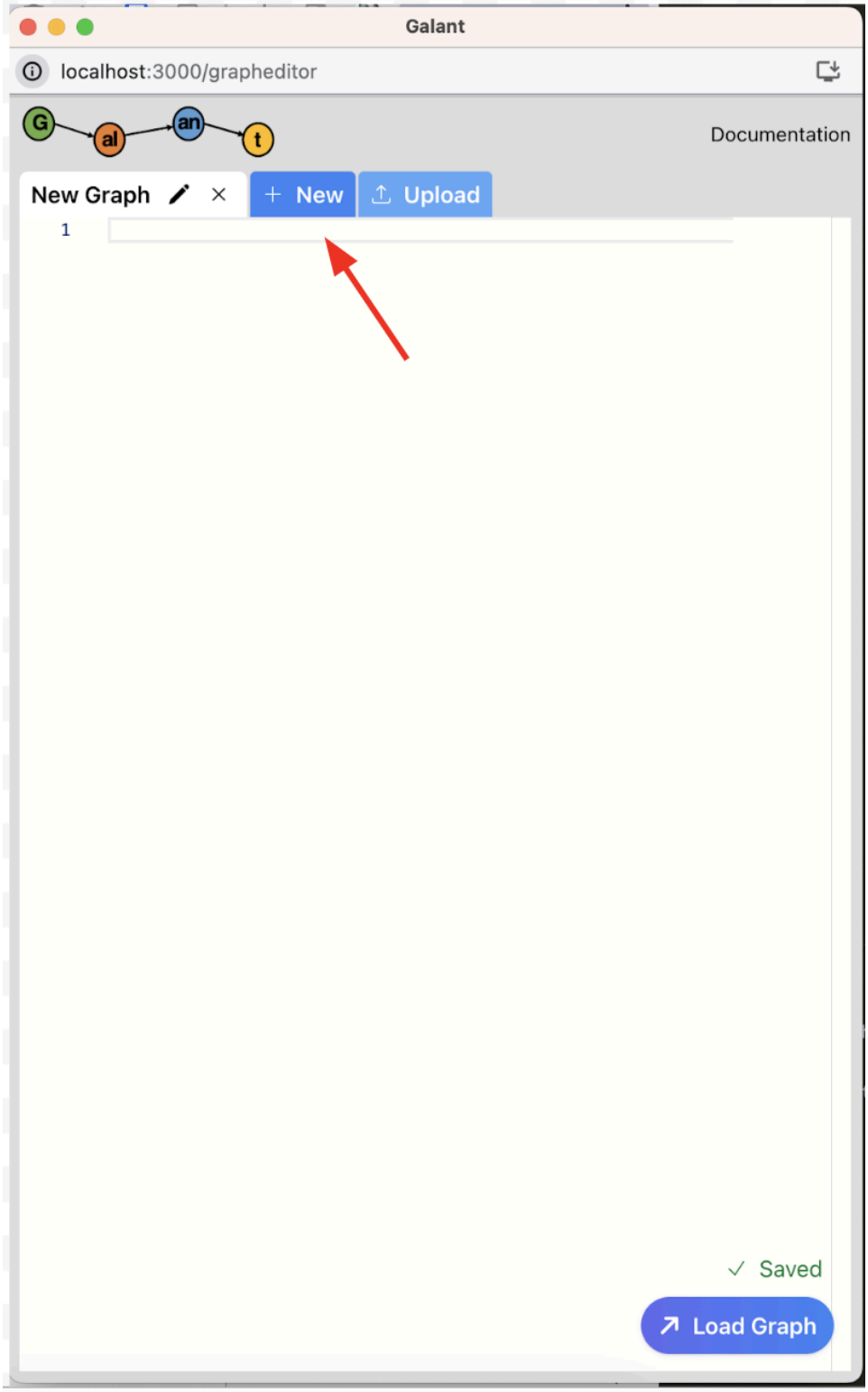
Load a Graph/Algorithm Example

Note: The graph steps apply the same as the algorithm steps

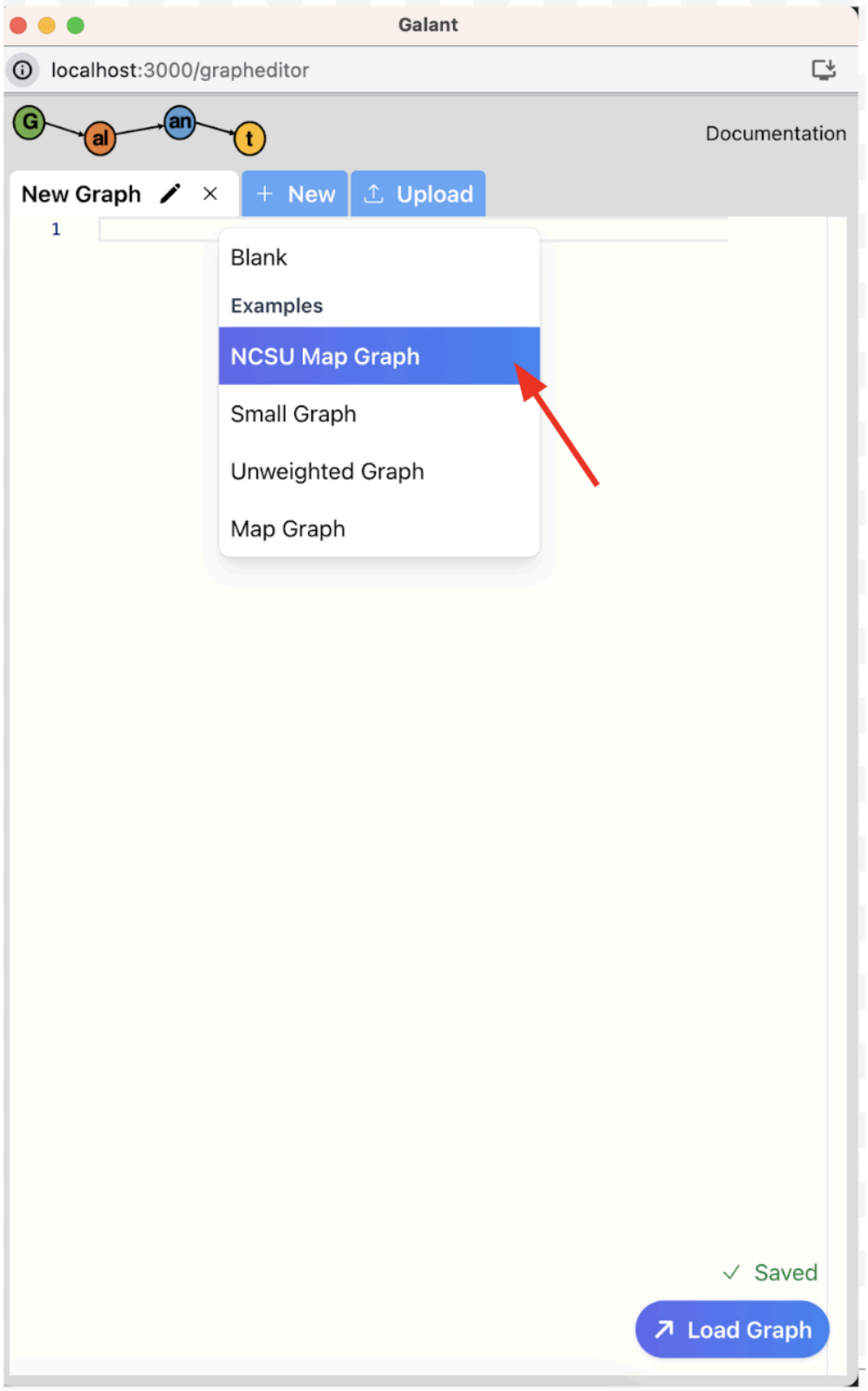
1. Load the application on a webpage.
2. Click on the "Graph/Algorithm Editor" button on the main view page.



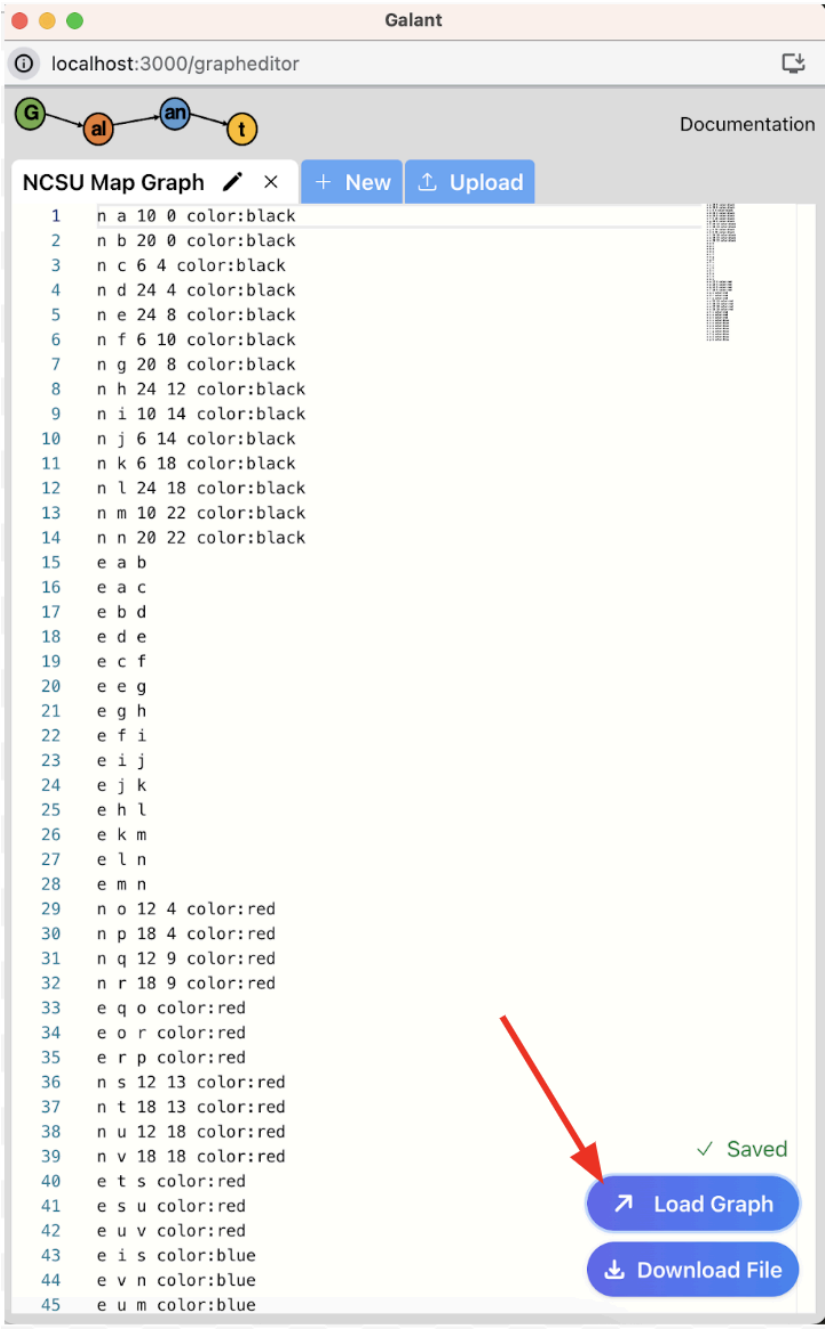
3. In the Graph/Algorithm Editor pop-up window, select the new button.

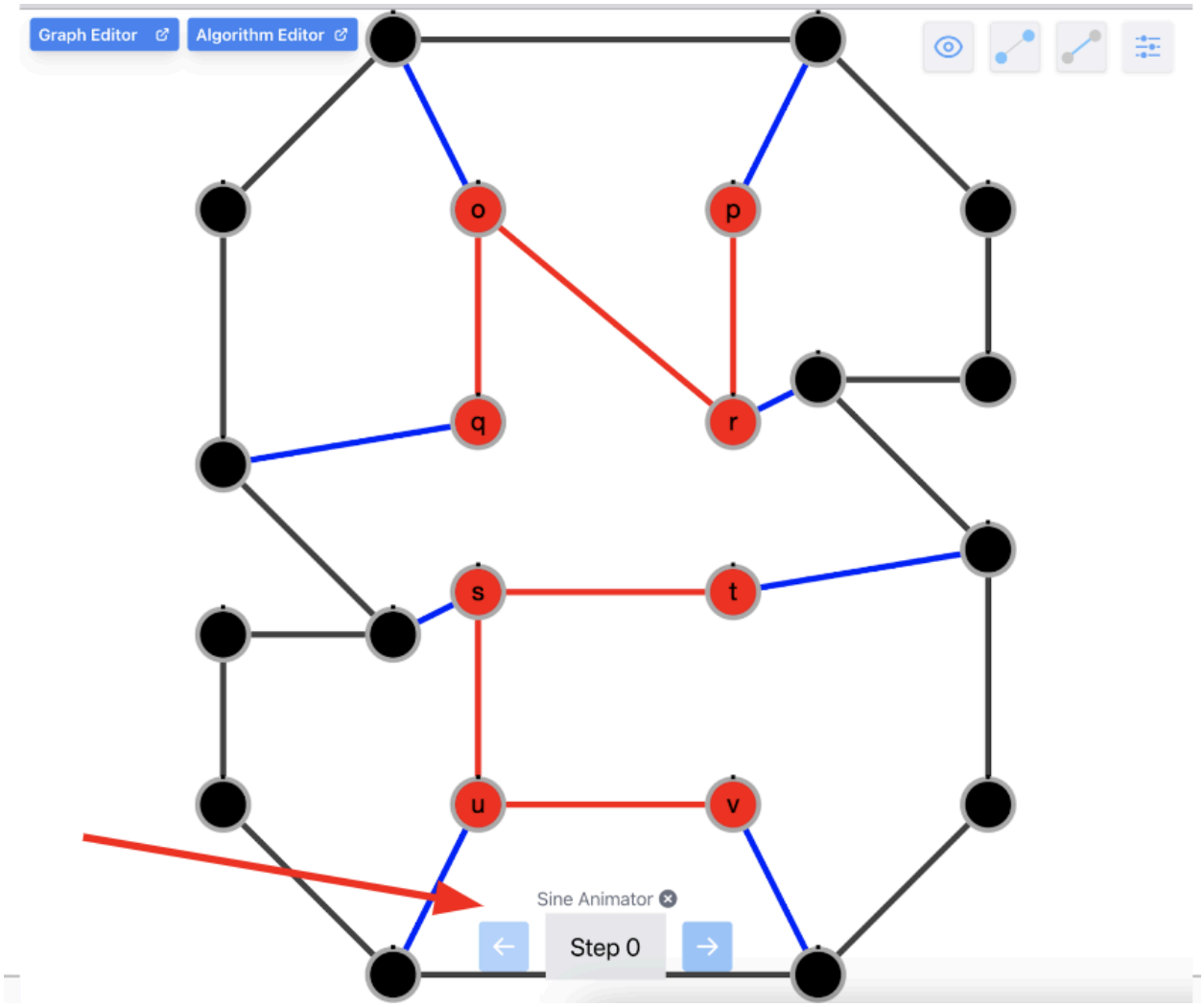


4. From the drop down select a graph/algorithm from the list of examples.



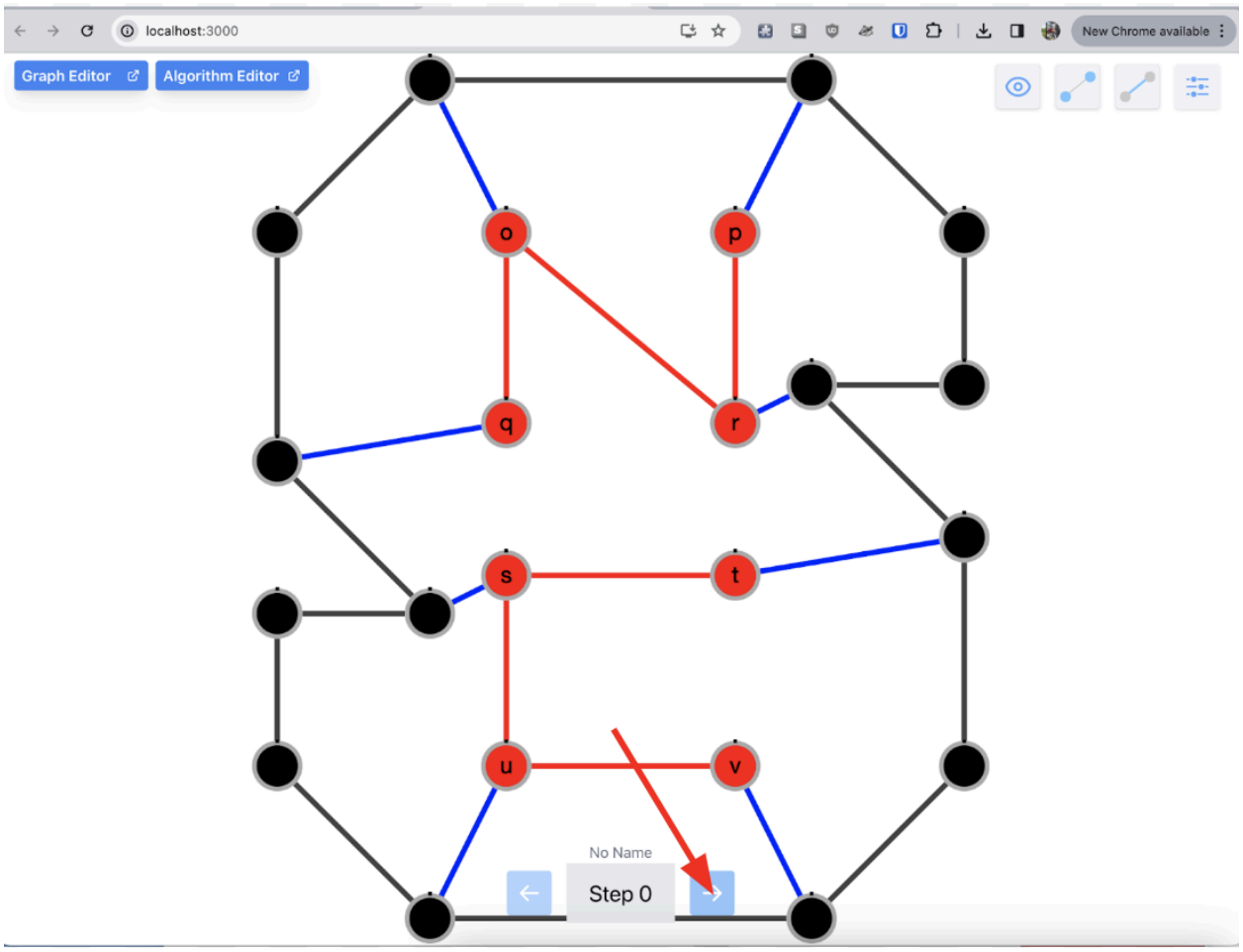
5. Click the "Load Graph/Algorithm" button, and if successful the graph view will be updated with the graph or the algorithm name at the bottom.



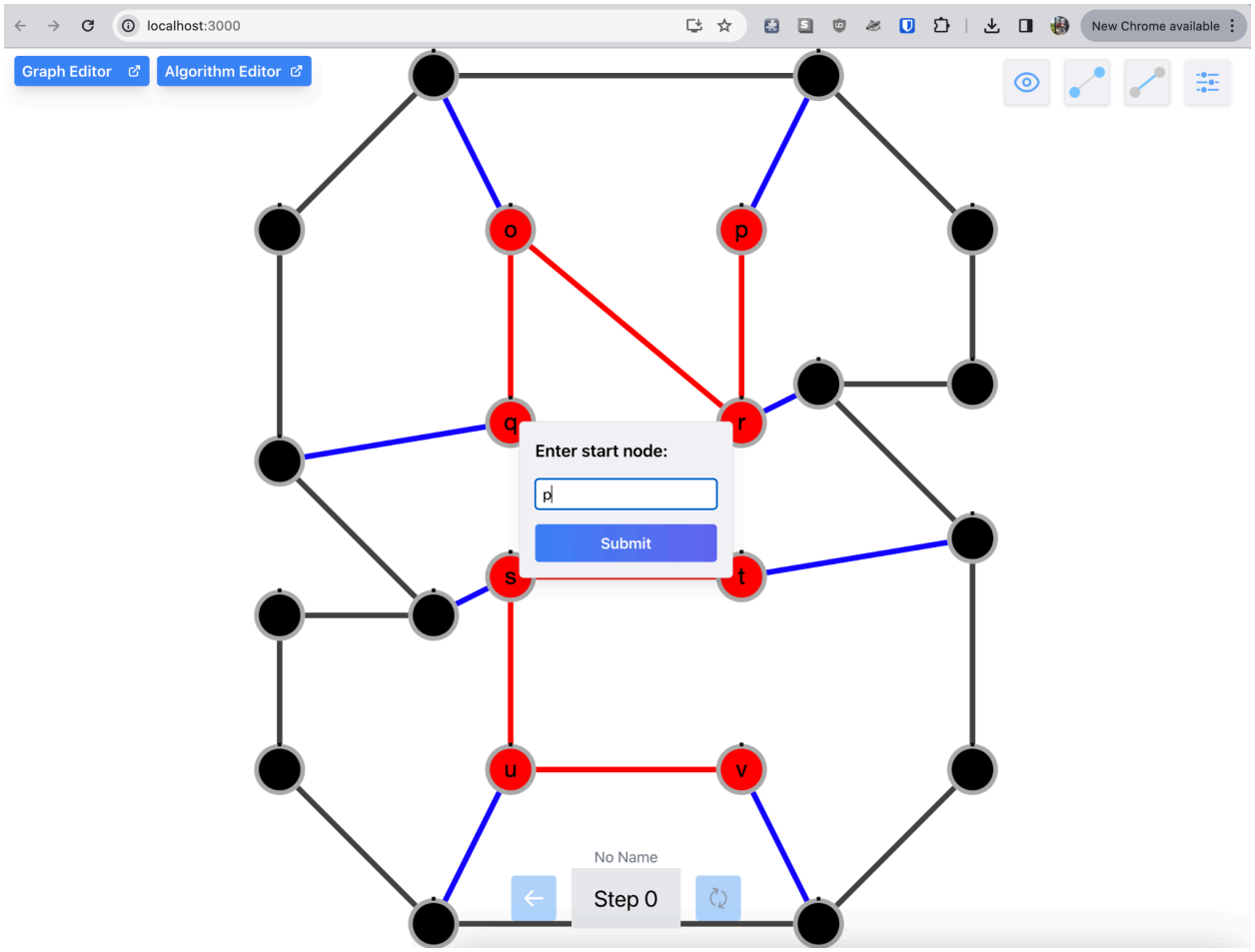

Graph Editor
Algorithm Editor
o
p
q
r
s
t
u
v
Sine Animator
Step 0

Running an Algorithm on a Loaded Graph

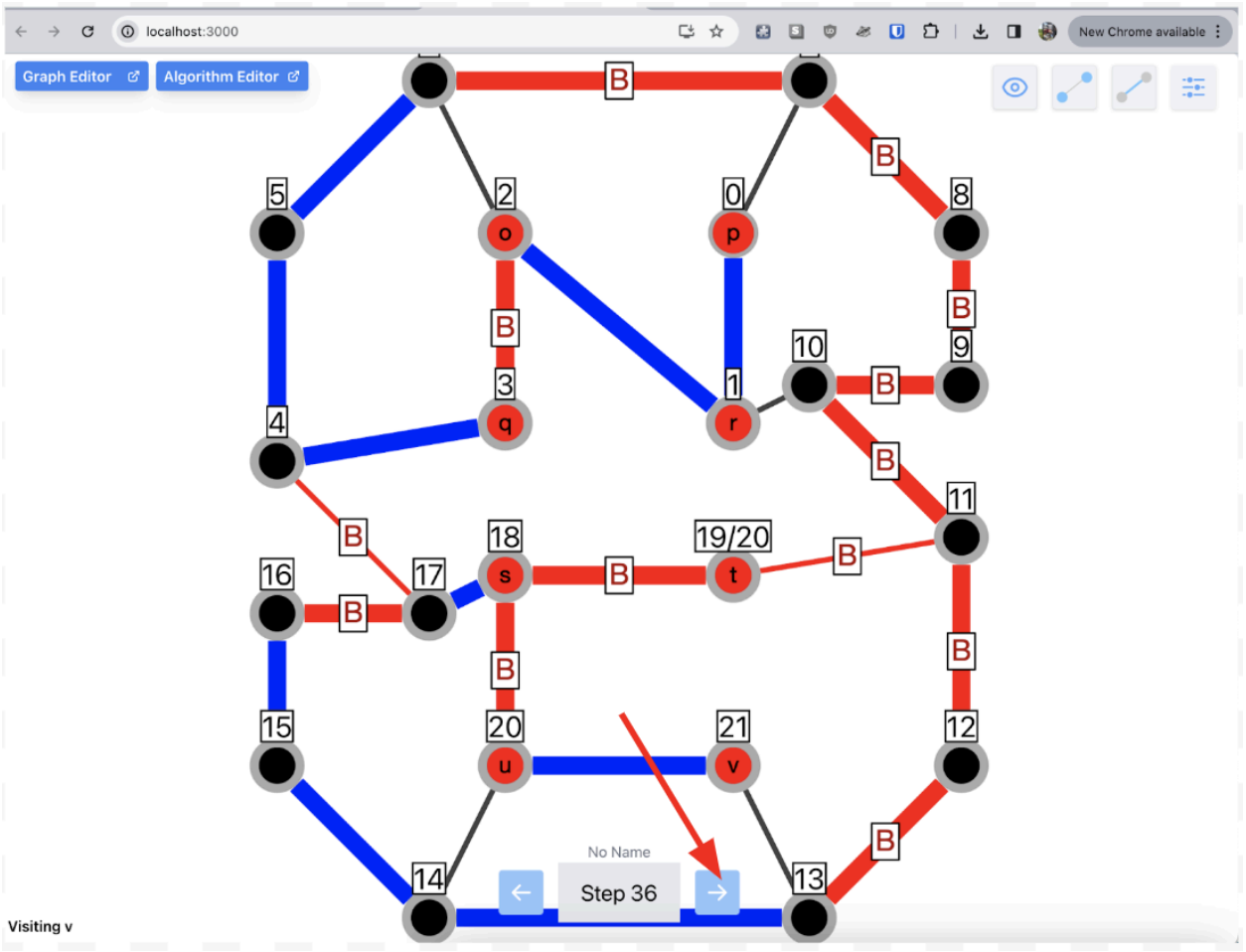
1. Once the graph and algorithm are loaded, click on the main window.
2. Click on the front step button at the bottom of the window to step through the algorithm on the graph. (You could also use the left and right keyboard arrows to step through)



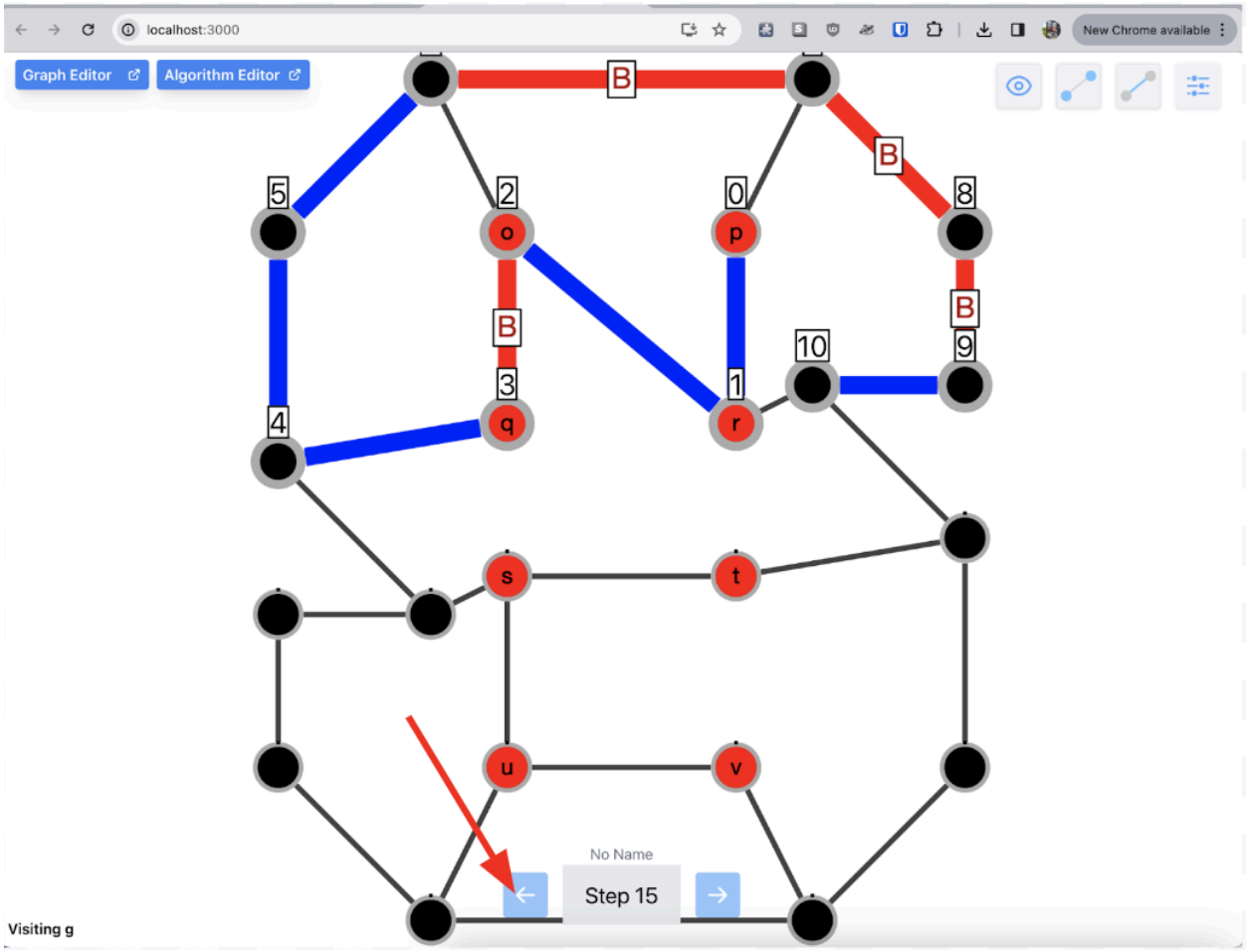
3. If prompted for a start node, enter the numeric value of the node that you would like to start the graph from.



4. Keep clicking on the front step button to traverse through the graph.



5. Click on the back arrow to trace back to a previous algorithm step on the graph.

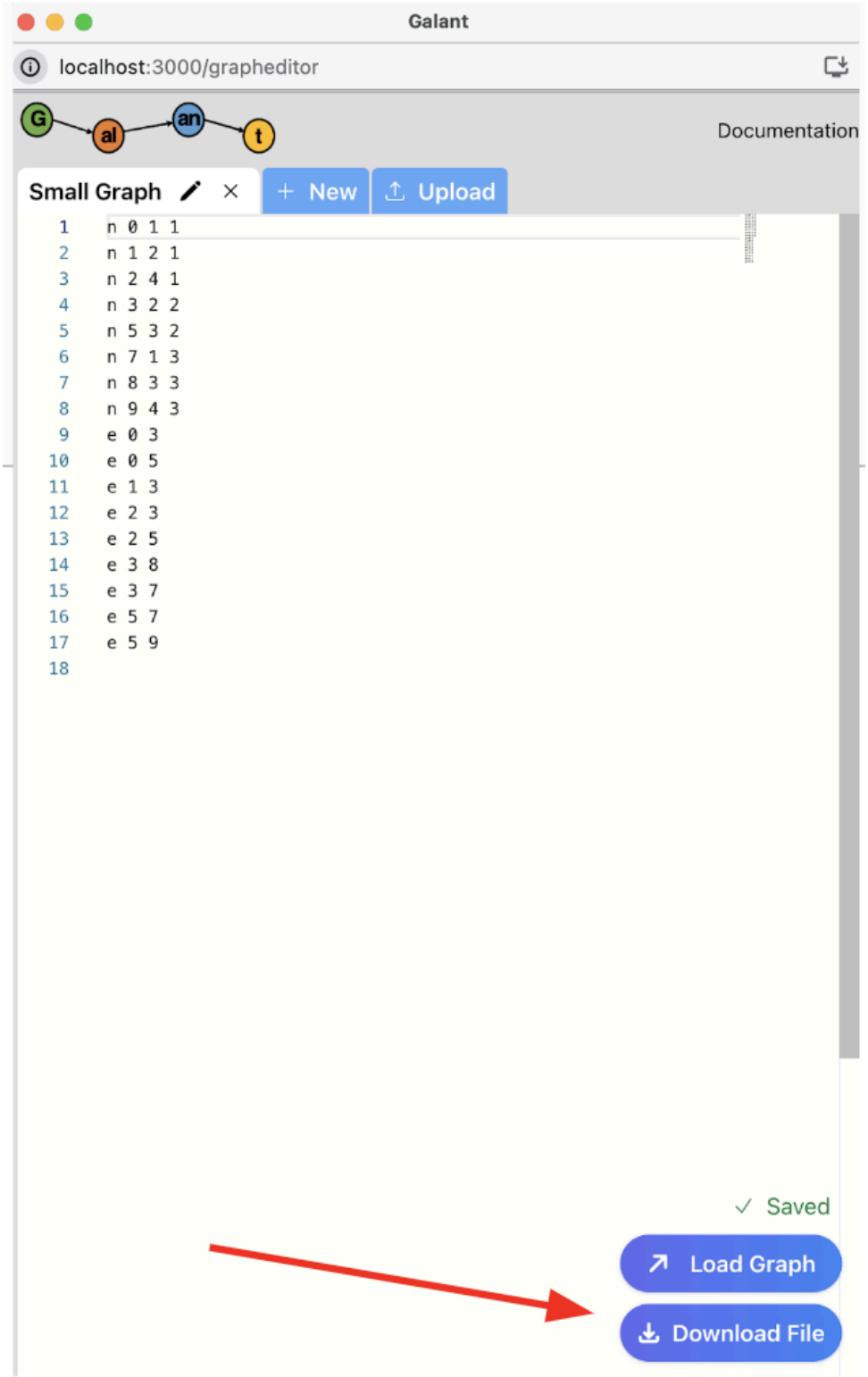


Exiting an Algorithm

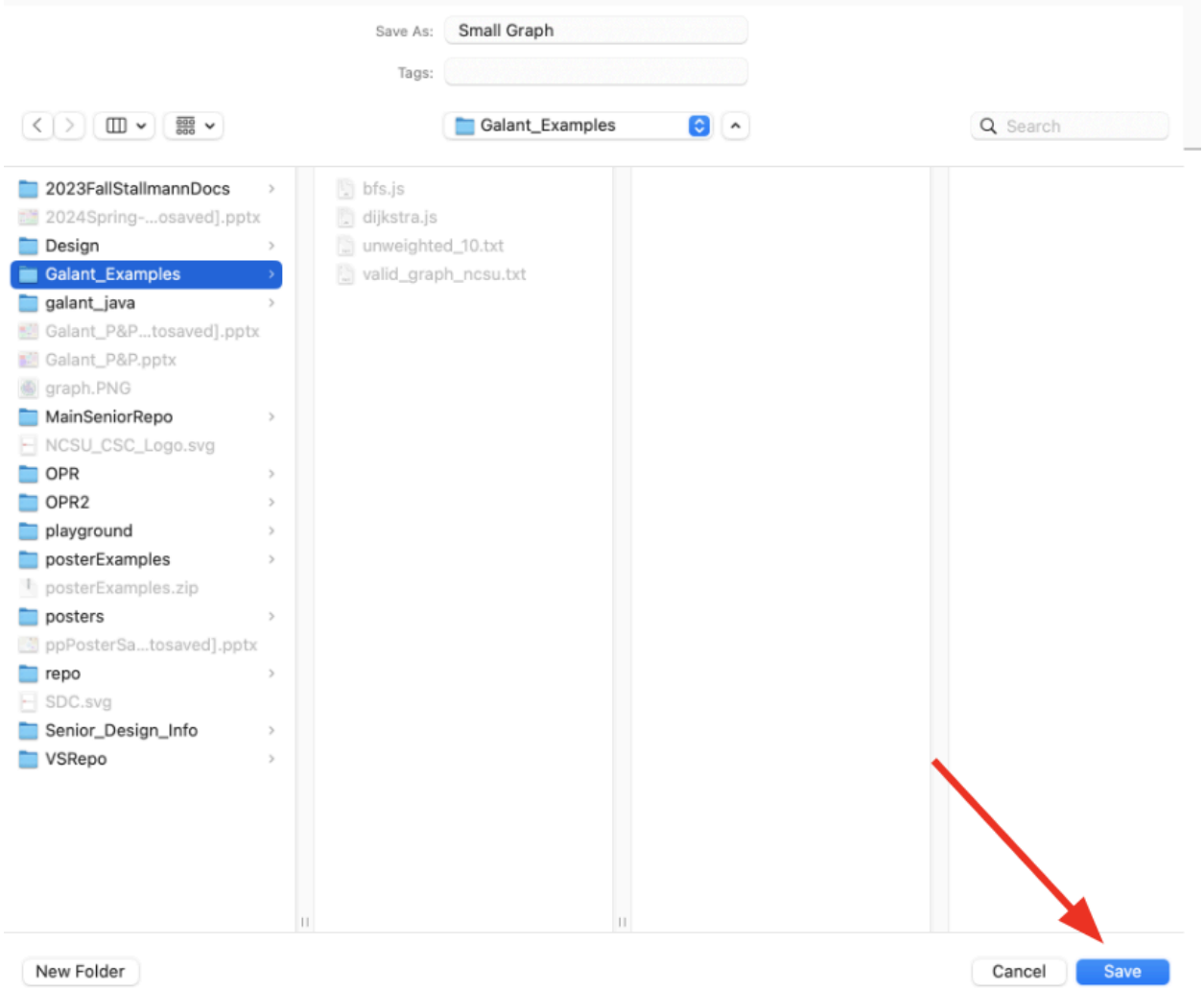
1. Ensure that the algorithm and the graph are loaded and running.
2. On the keyboard, hit the escape key to exit the algorithm, or click the small ‘x’ button next to the name of the algorithm above the step counter.

Download a Graph/Algorithm

1. Ensure that a graph has been successfully loaded in the graph editor window.
2. Click the “Download File” Button.

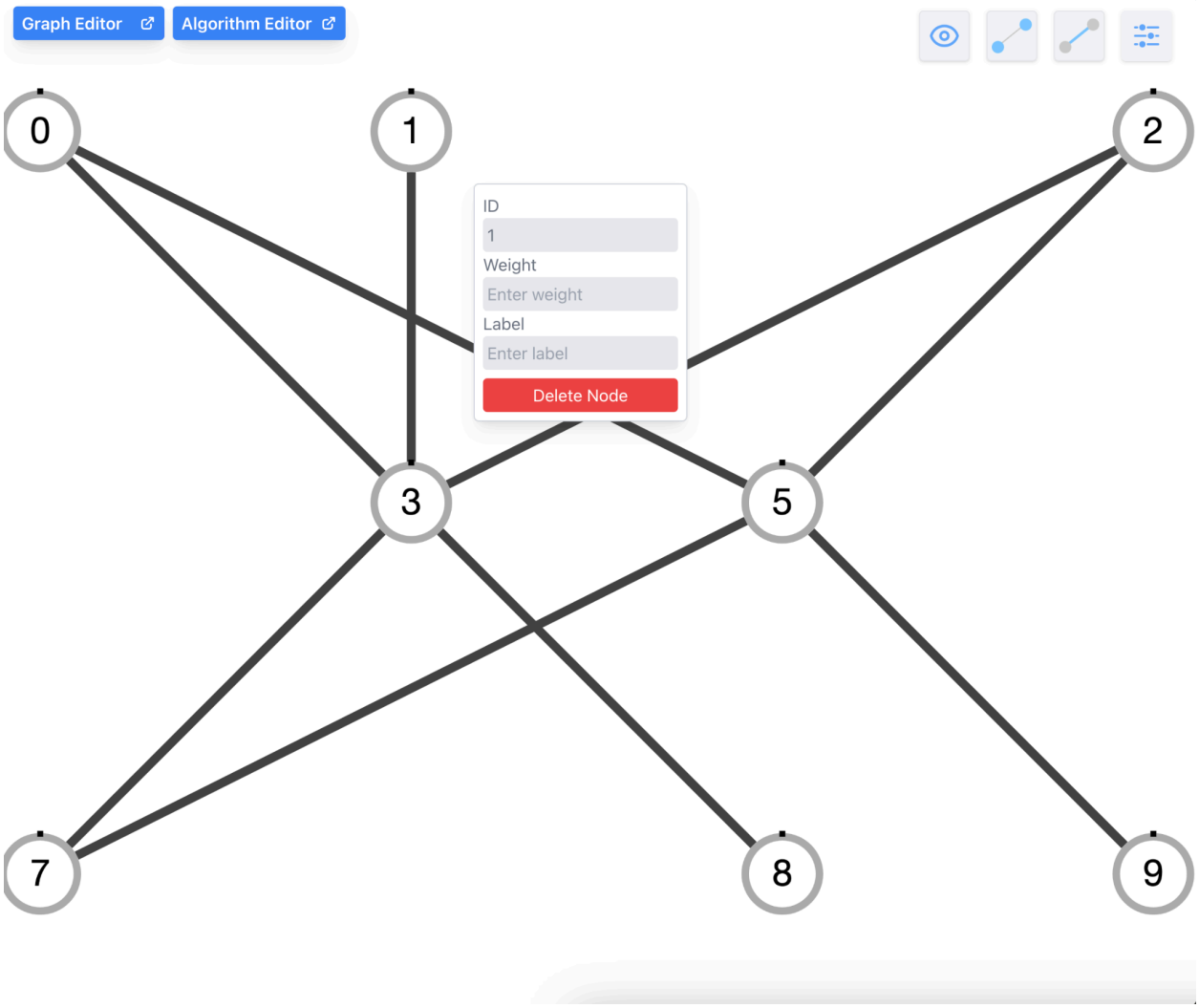

Galant
localhost:3000/grapheditor
G
al
an
t
Documentation
Small Graph
New
Upload
1 n 0 1 1
2 n 1 2 1
3 n 2 4 1
4 n 3 2 2
5 n 5 3 2
6 n 7 1 3
7 n 8 3 3
8 n 9 4 3
9 e 0 3
10 e 0 5
11 e 1 3
12 e 2 3
13 e 2 5
14 e 3 8
15 e 3 7
16 e 5 7
17 e 5 9
18
Saved
Load Graph
Download File

3. Navigate to the save location then press the “Save” button.

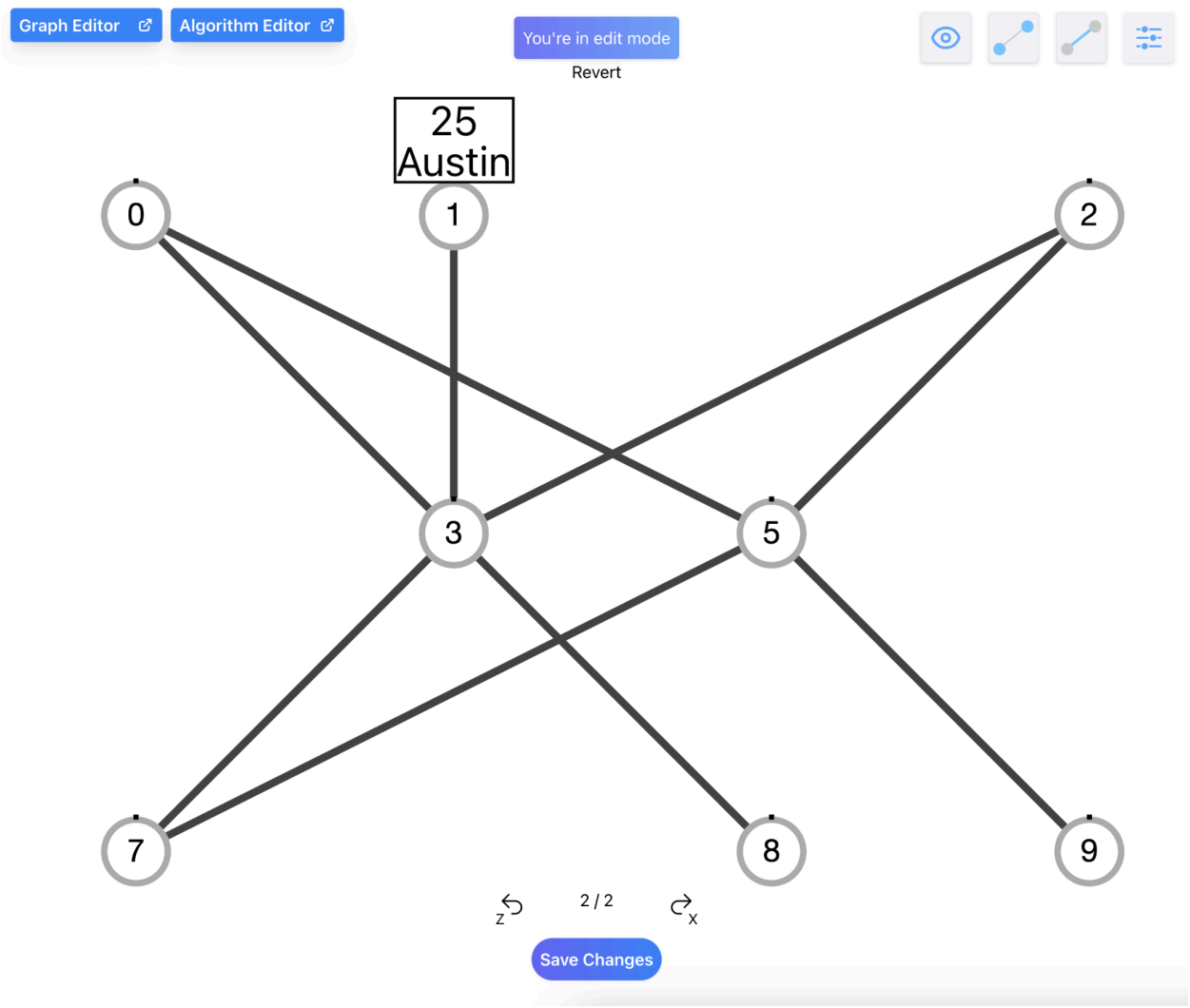


Adding Node Label and Weight

1. Ensure that a graph has been successfully loaded in the main window.
2. Right click on a node you wish to add a label or weight to.



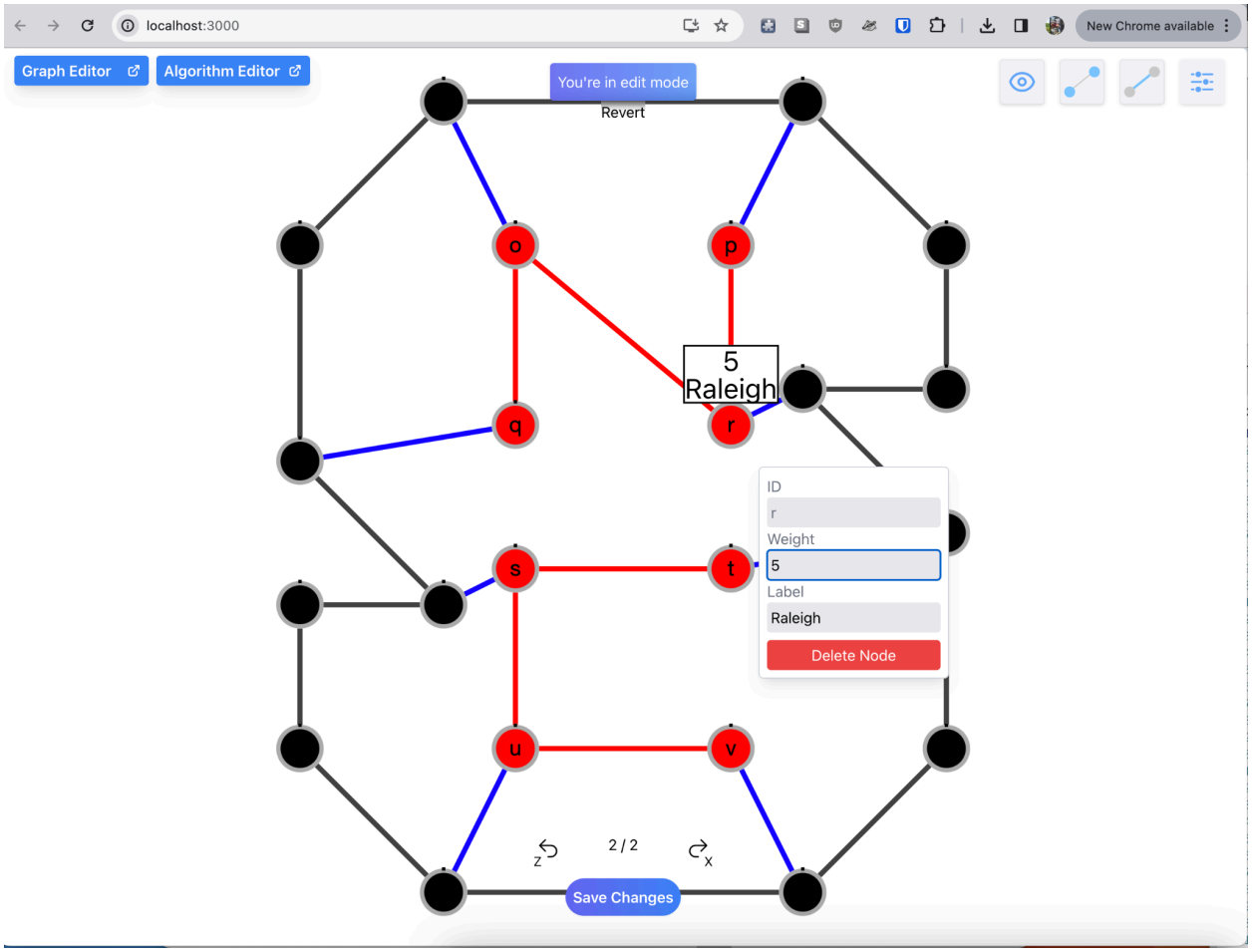
3. Enter the label name and weight of the node.
4. Click off to the side of the pop-up to save the node changes.



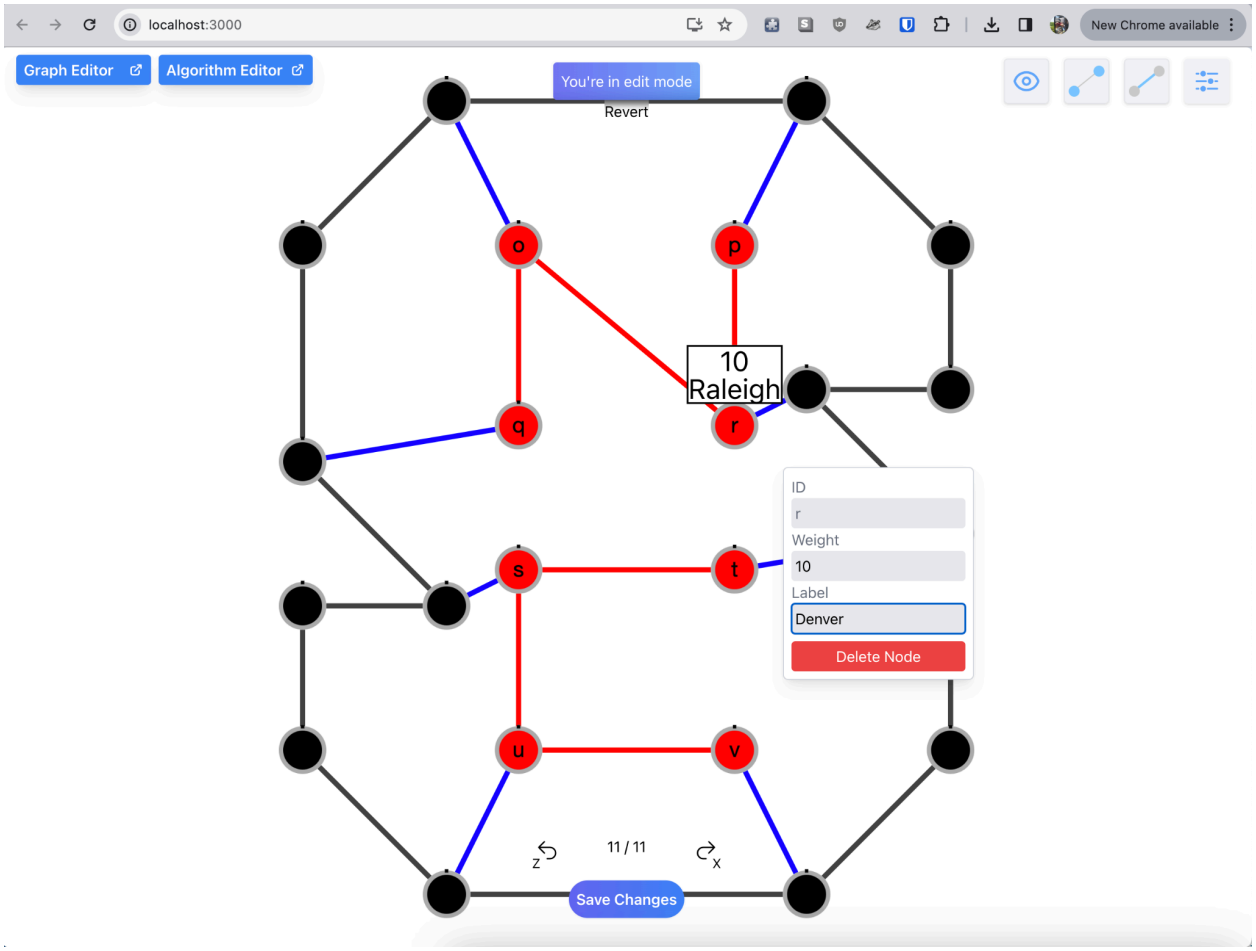
Editing Node Label and Weight

1. Ensure that a graph has been successfully loaded in the main window.

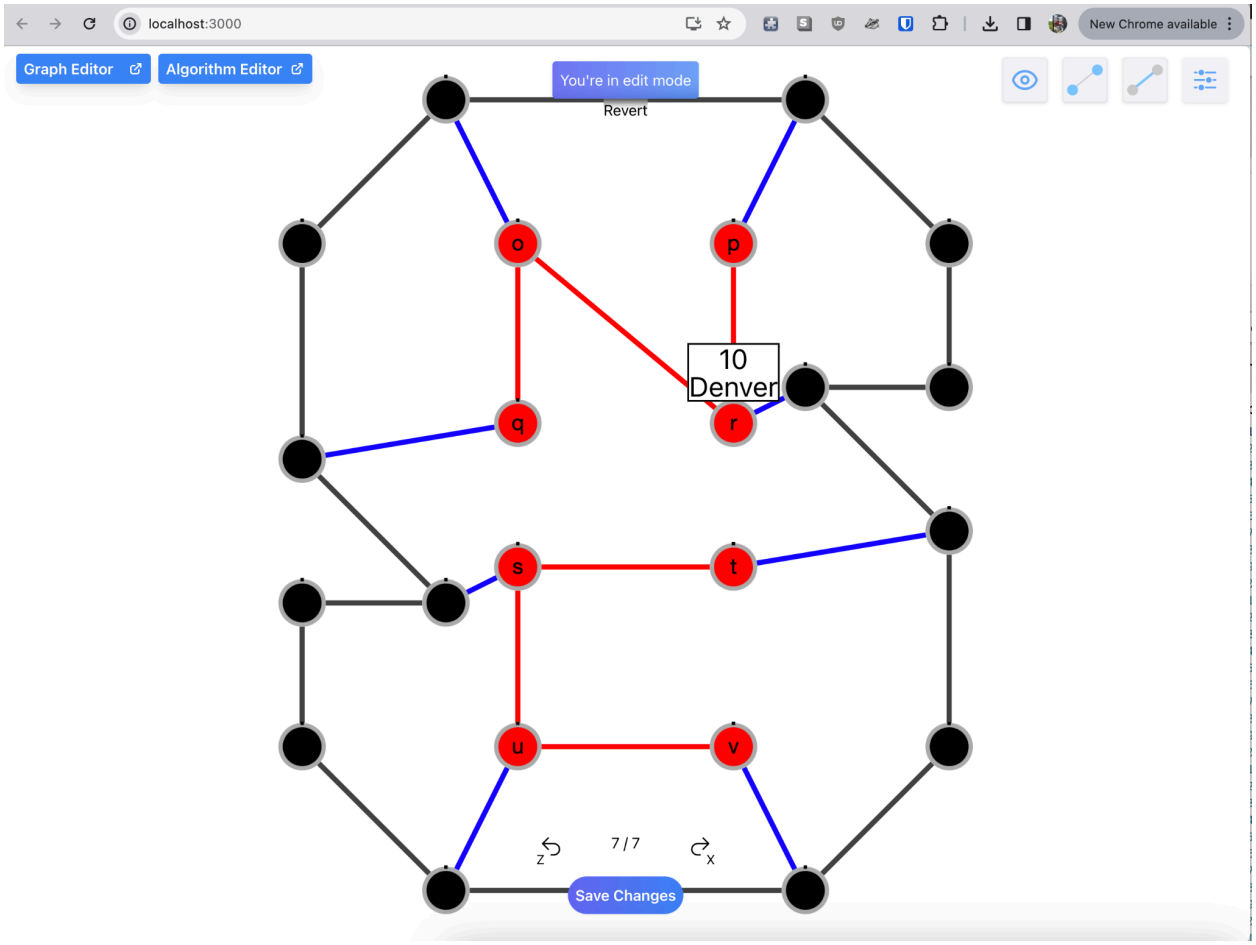
2. Right-click a node you wish to edit.



3. In the pop-up box, enter your new desired weight and label for that node.

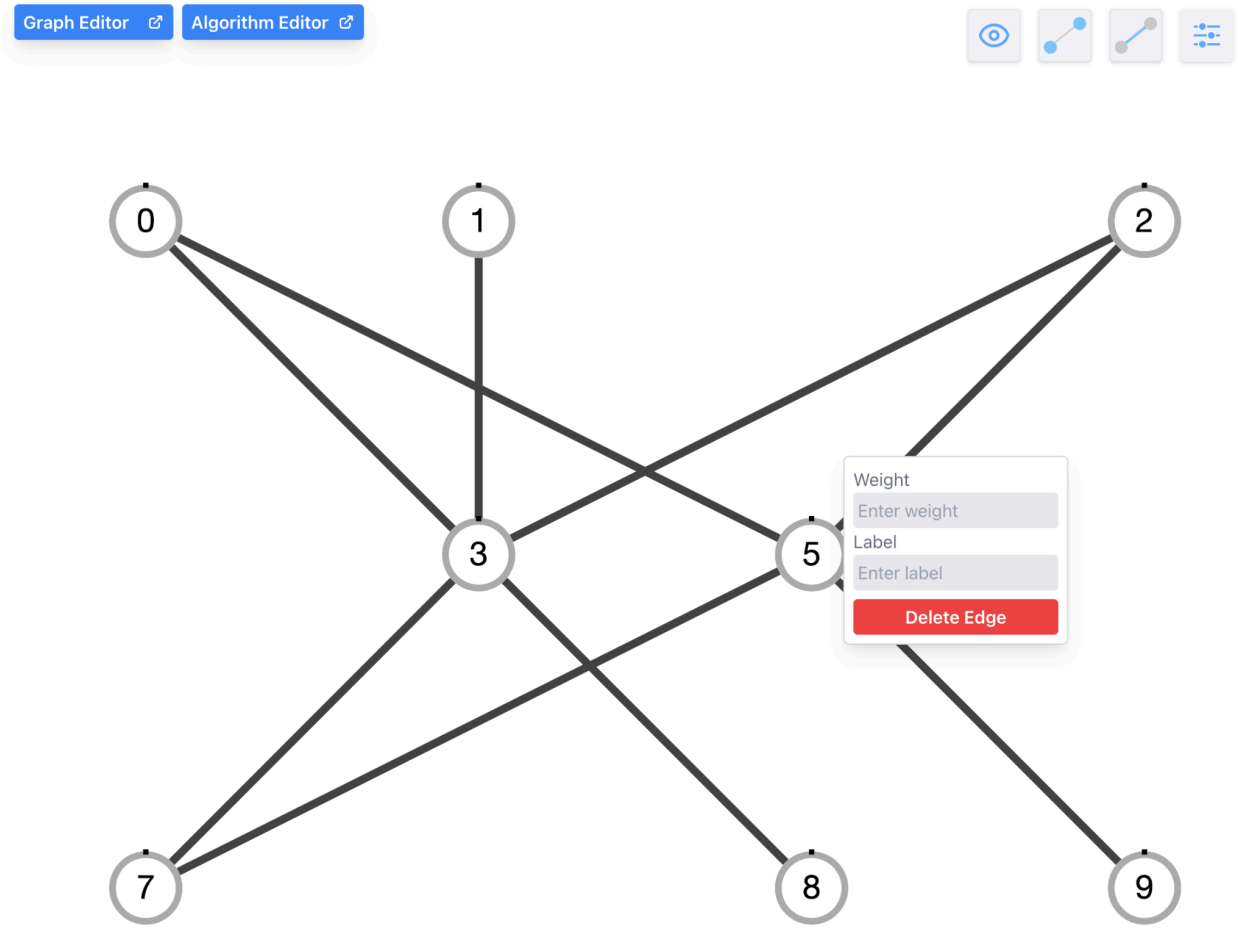


4. Click off the node and the changes will be updated on the node.



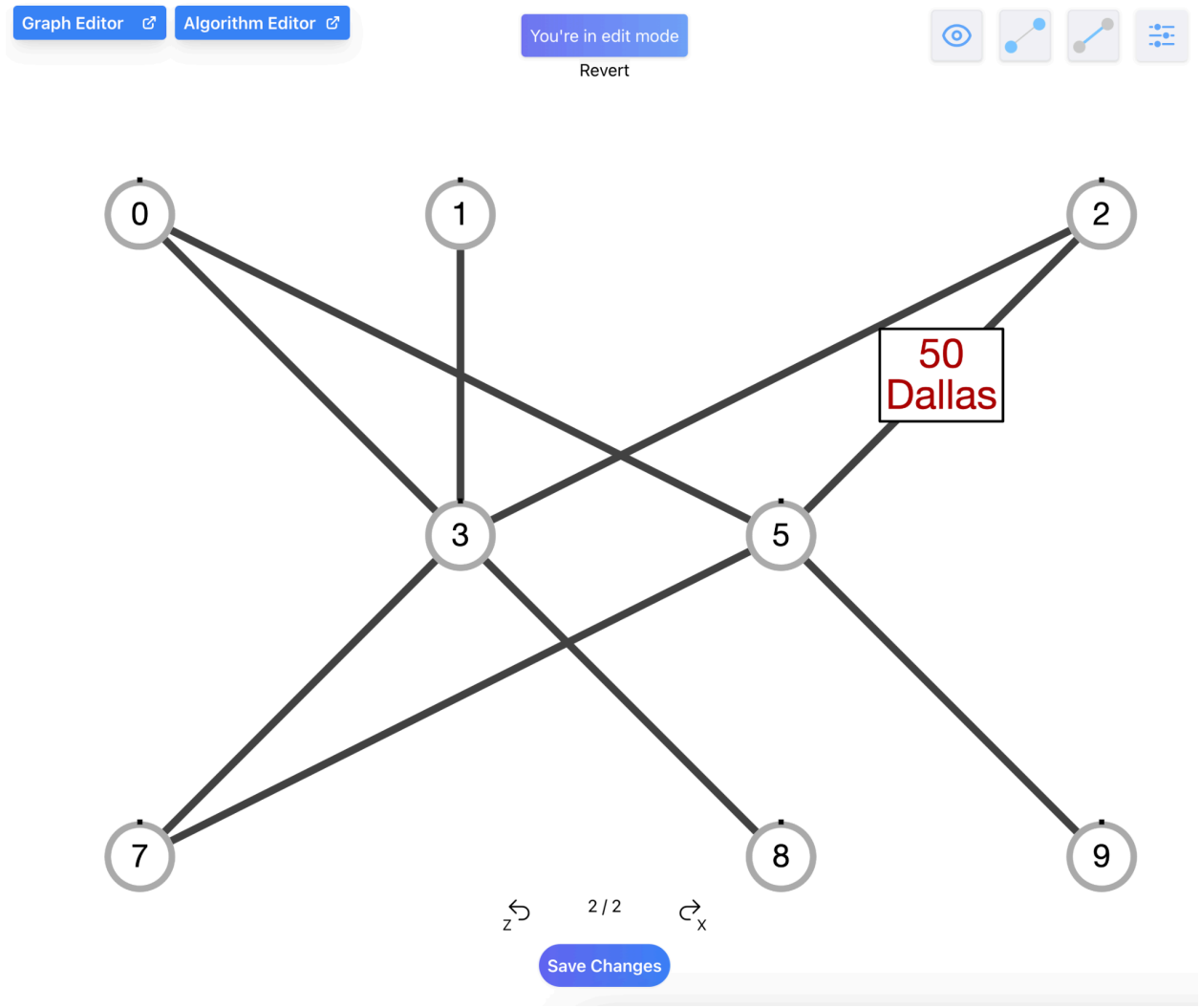
Adding Edge Label and Weight

1. Ensure that a graph has been successfully loaded in the main window.
2. Right-click on an edge to which you wish to add a label or weight.



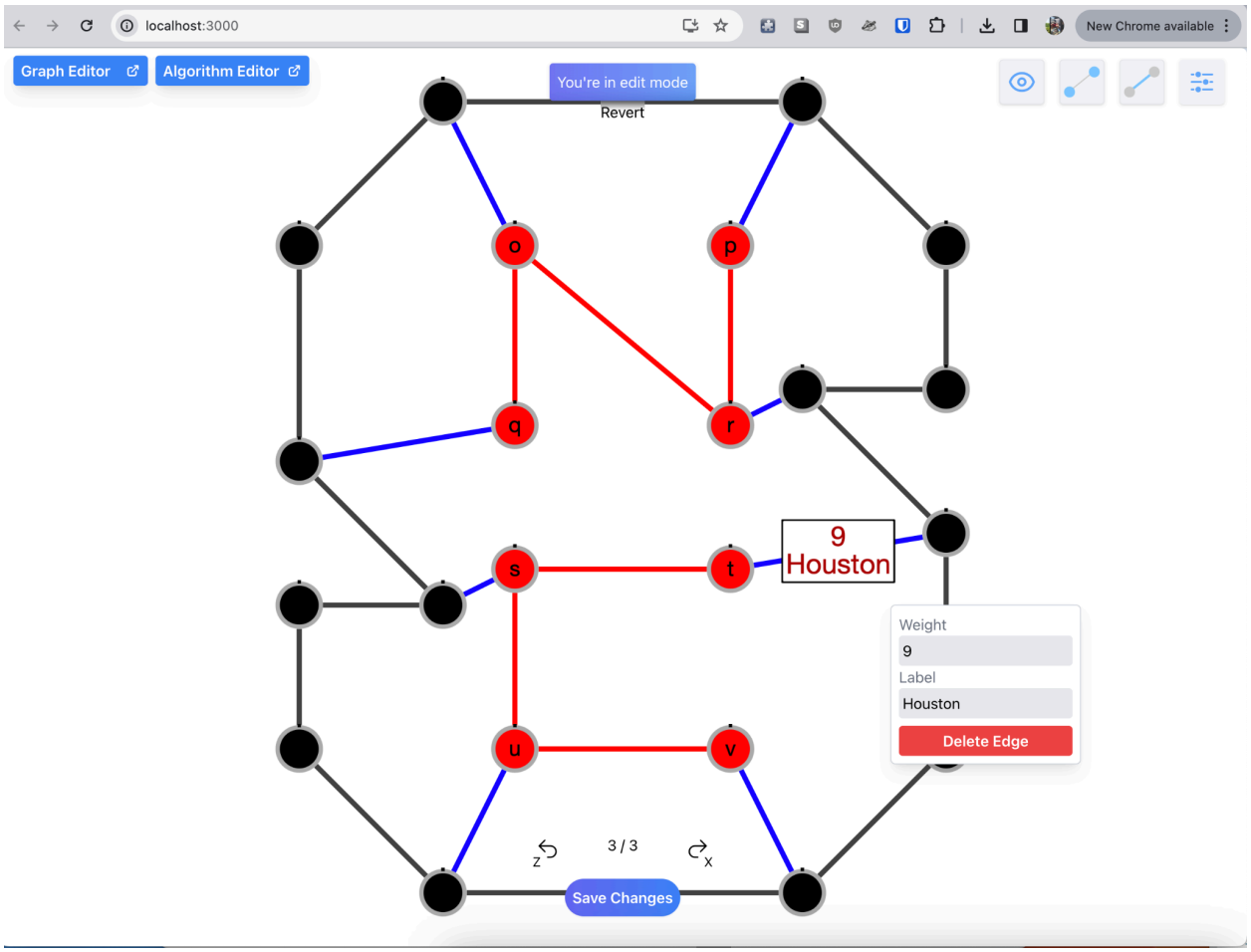
3. Enter the label name and weight of the edge.

4. Click off to the side of the pop-up to save the node changes.

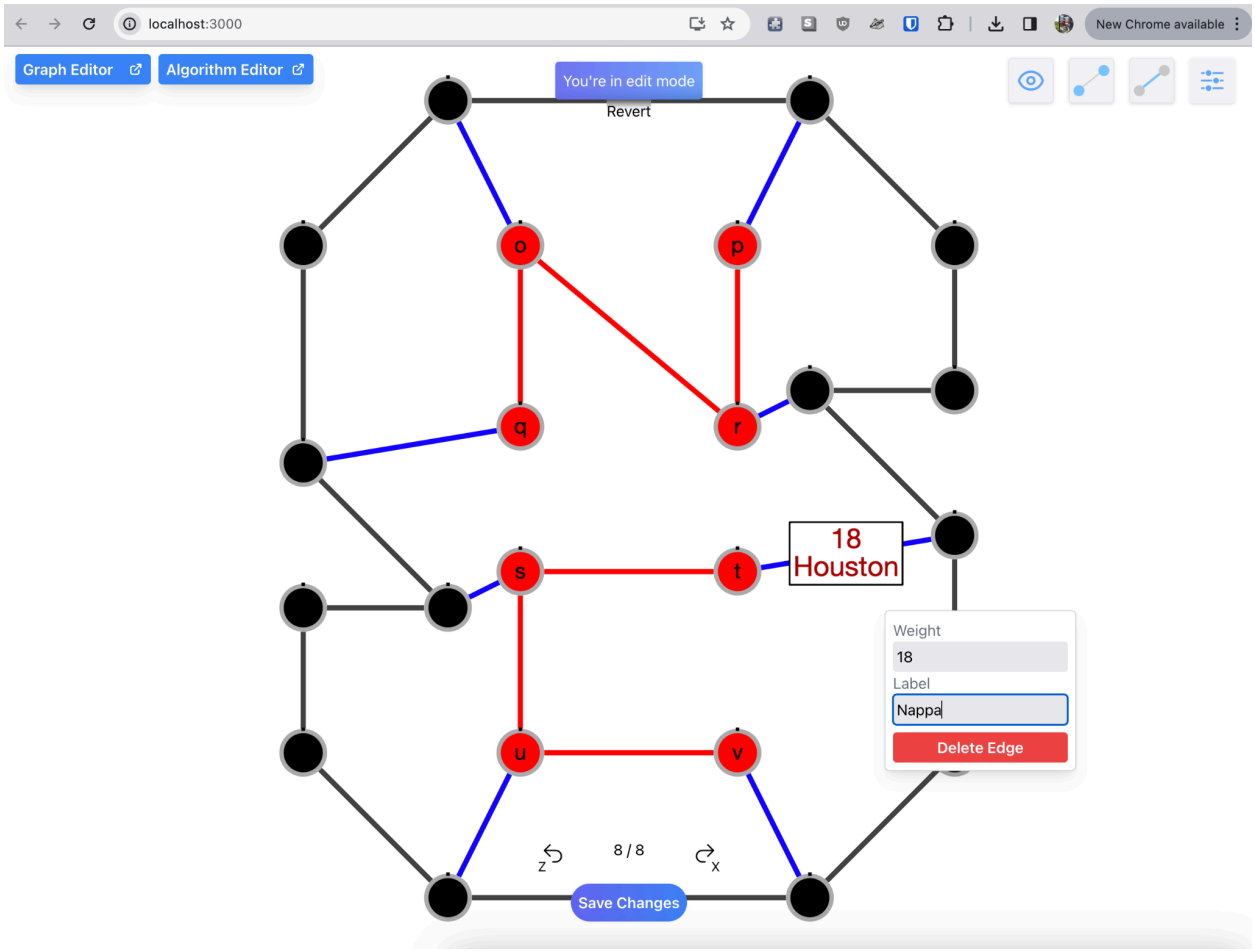


Editing Edge Label and Weight

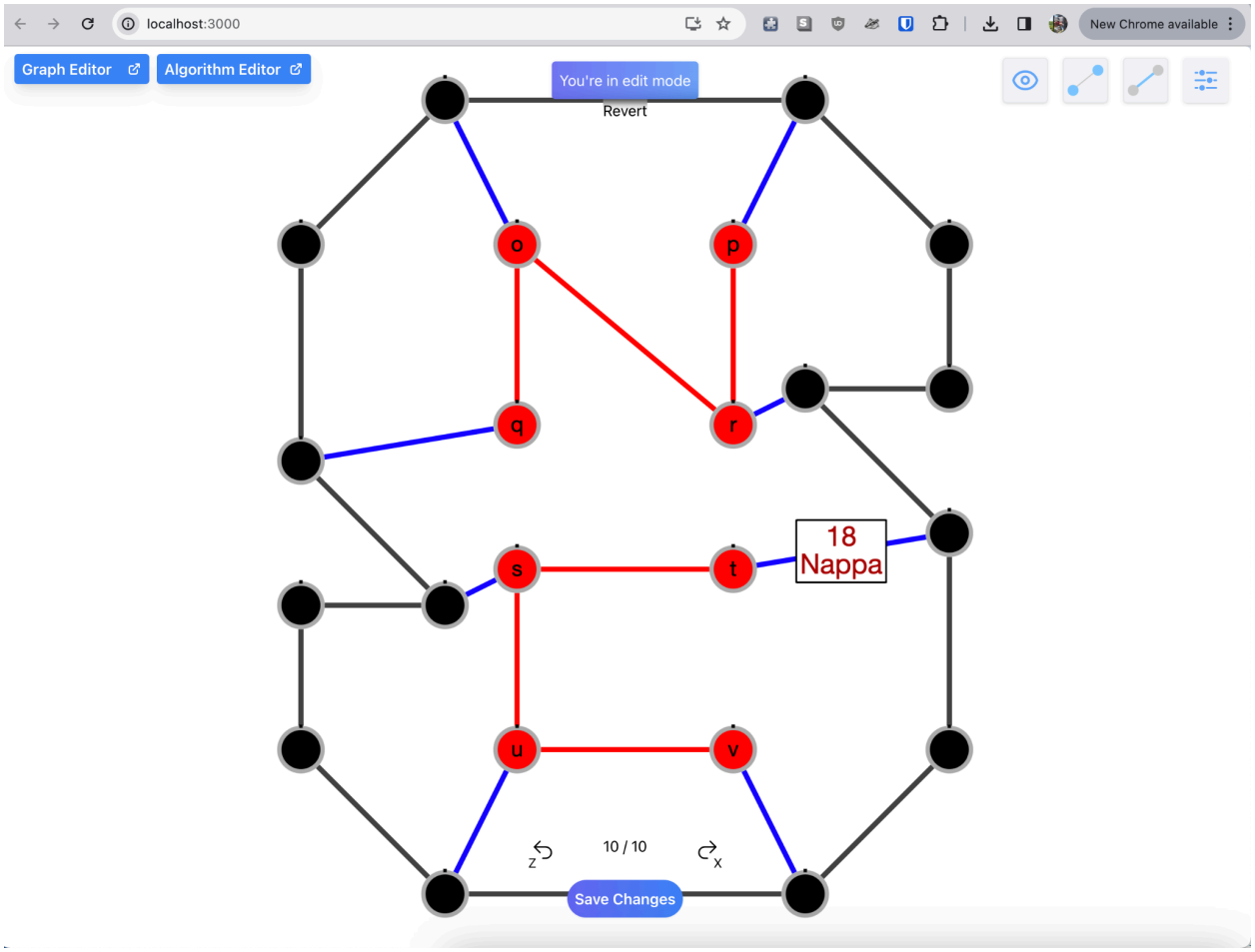
1. Ensure that a graph has been successfully loaded in the main window.
2. Right-click an edge you wish to edit.



3. In the pop-up box, enter your new desired weight and label for that edge.

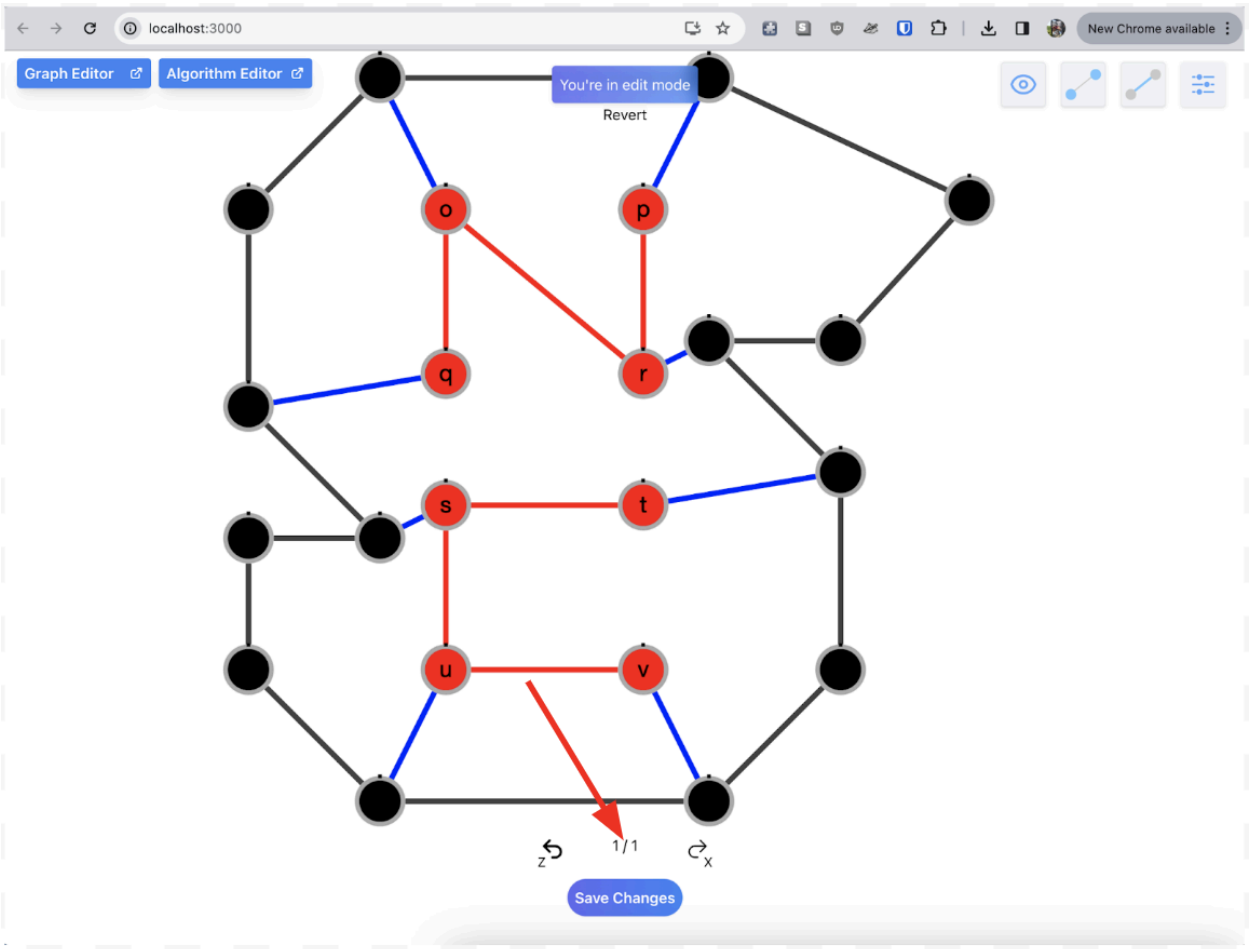


4. Click off the edge and the changes will be updated on the node.

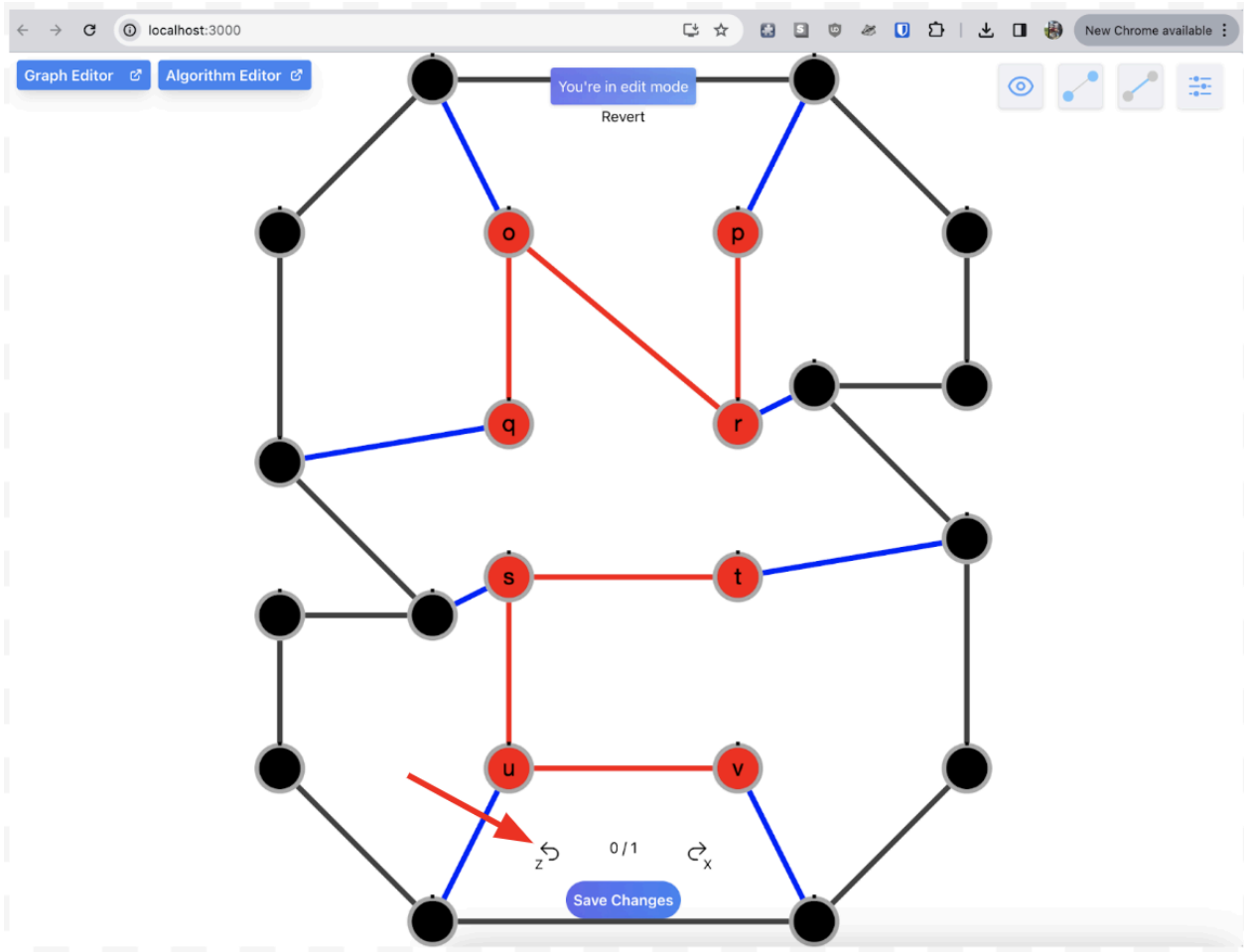


Editing a Graph in Edit Mode

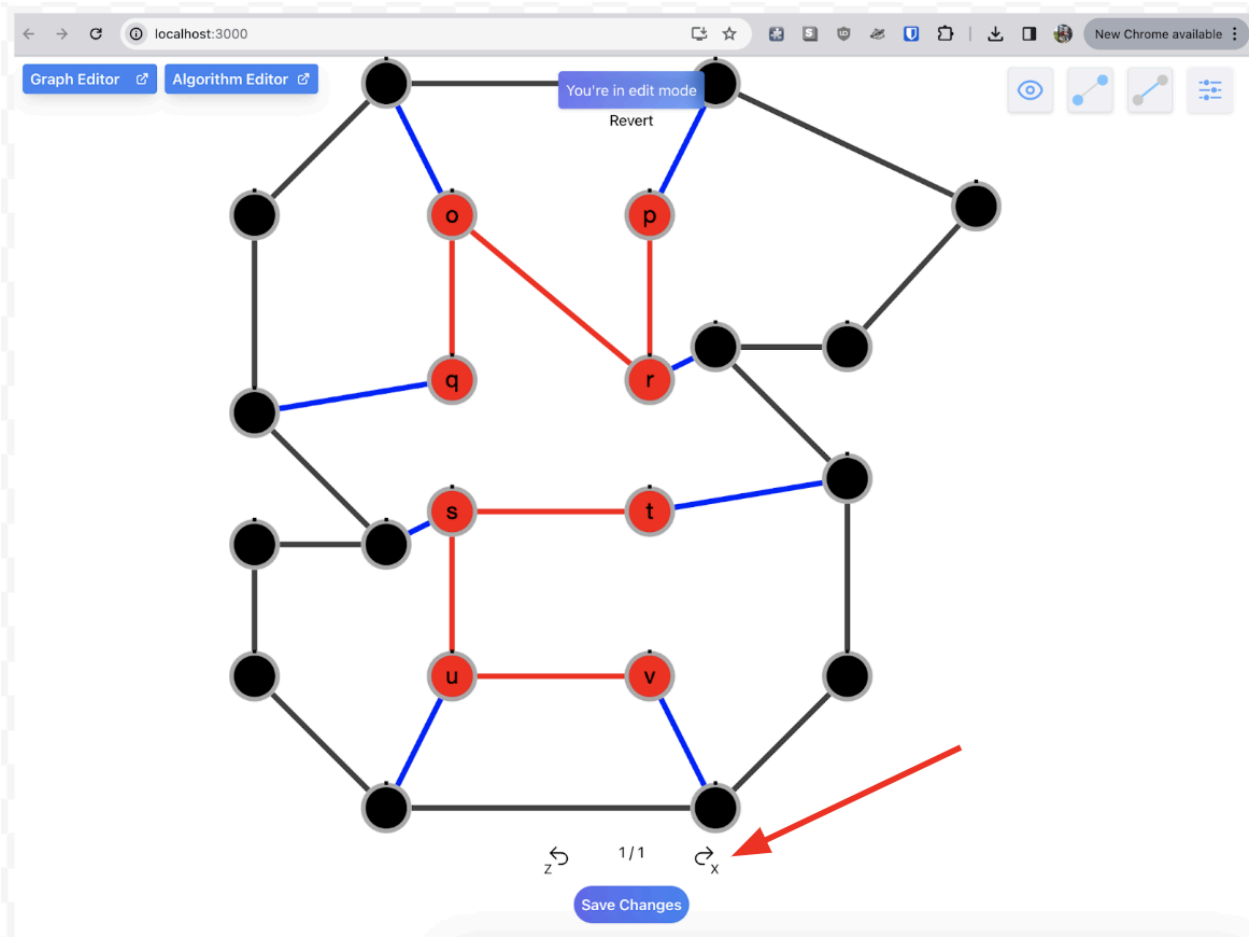
1. Ensure that a graph has been successfully loaded in the main window.
2. To make changes to a graph, you can drag nodes/edges, edit nodes/edges, and create nodes/edges. These changes will be reflected as a count below.



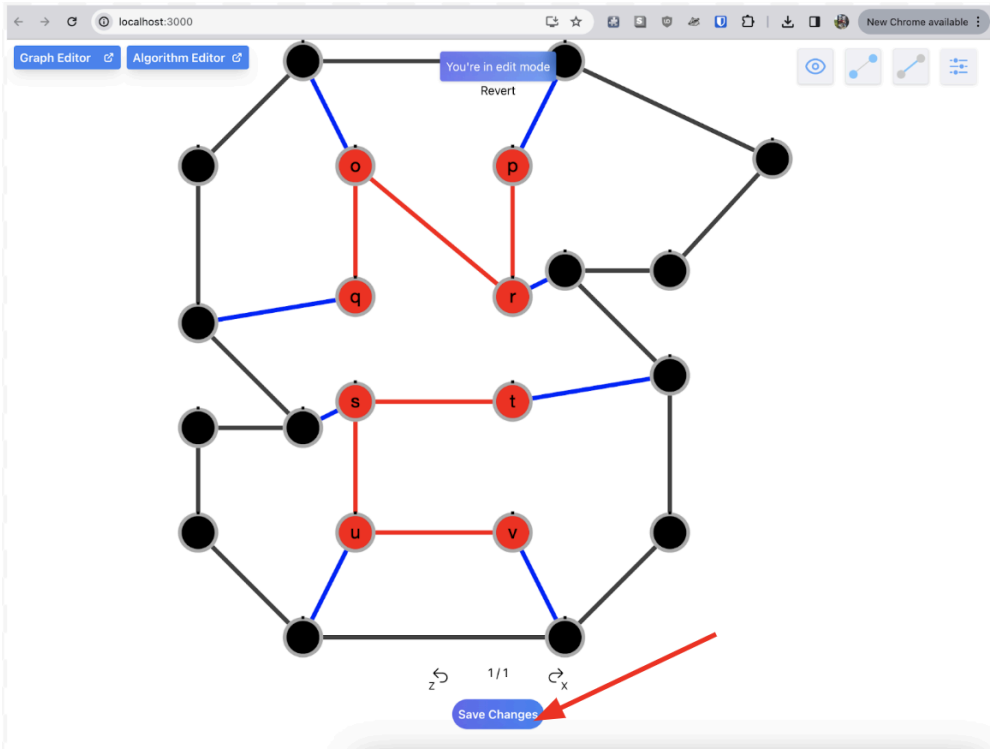
2. To undo any changes press the back button or the ‘z’ key on your keyboard.

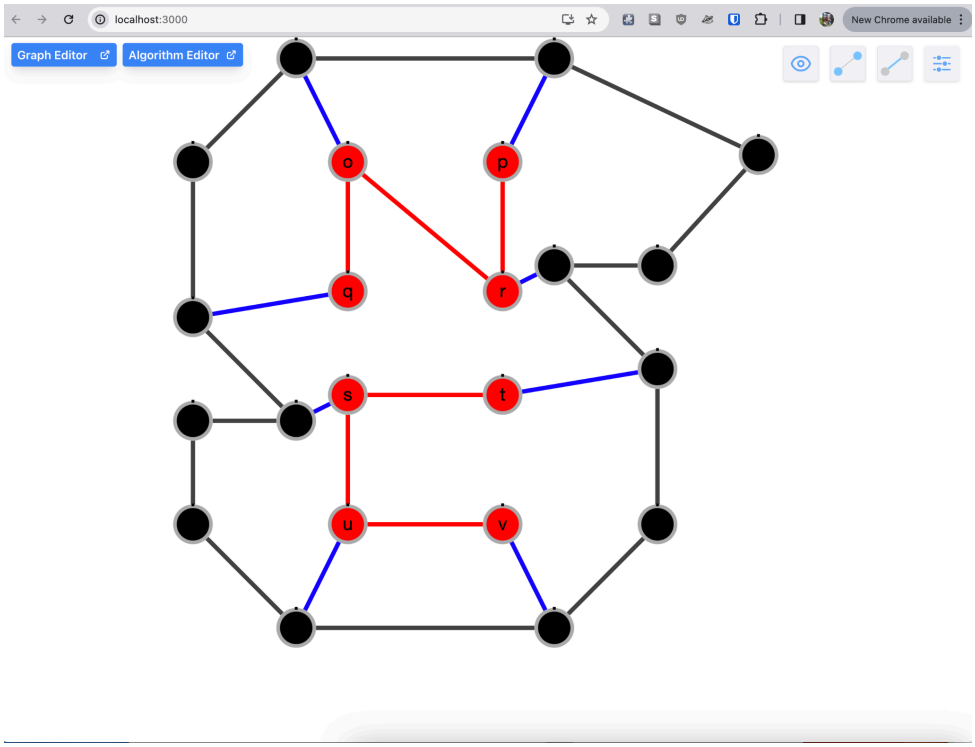


3. To redo any changes previously made press the forward button or the ‘x’ key on your keyboard.



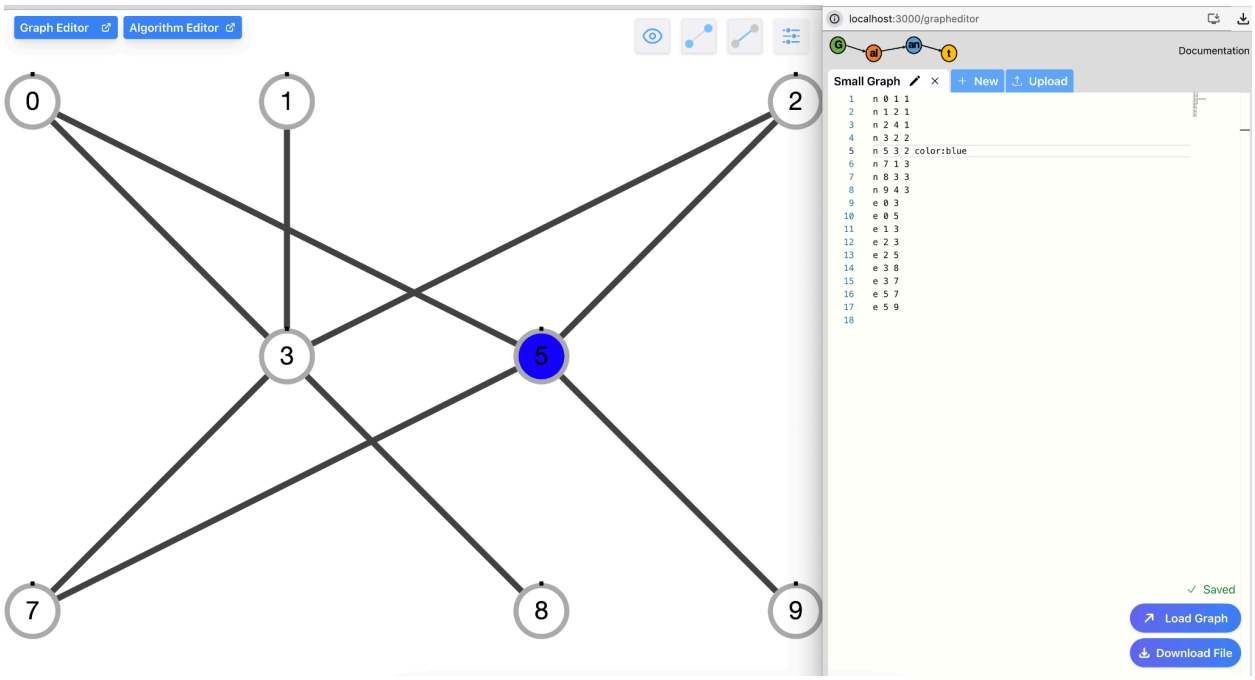
3. To save changes made to the graph, click the “Save Changes” button. This will bring you out of edit mode and into graph mode.



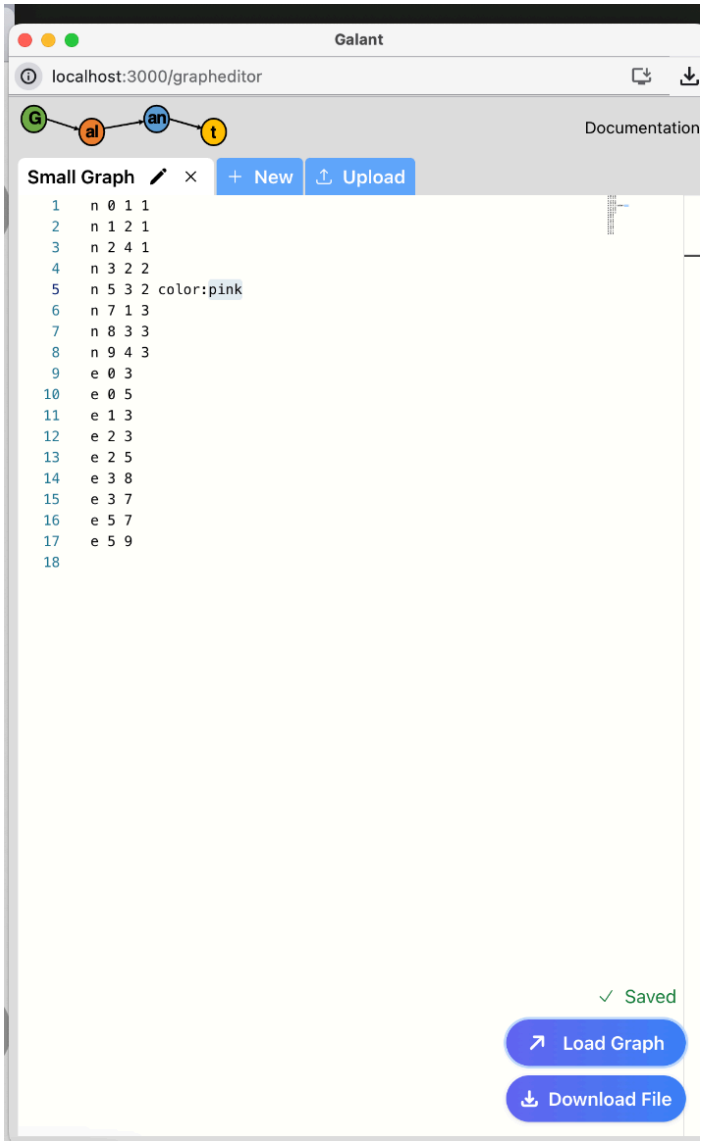


Editing Graph text

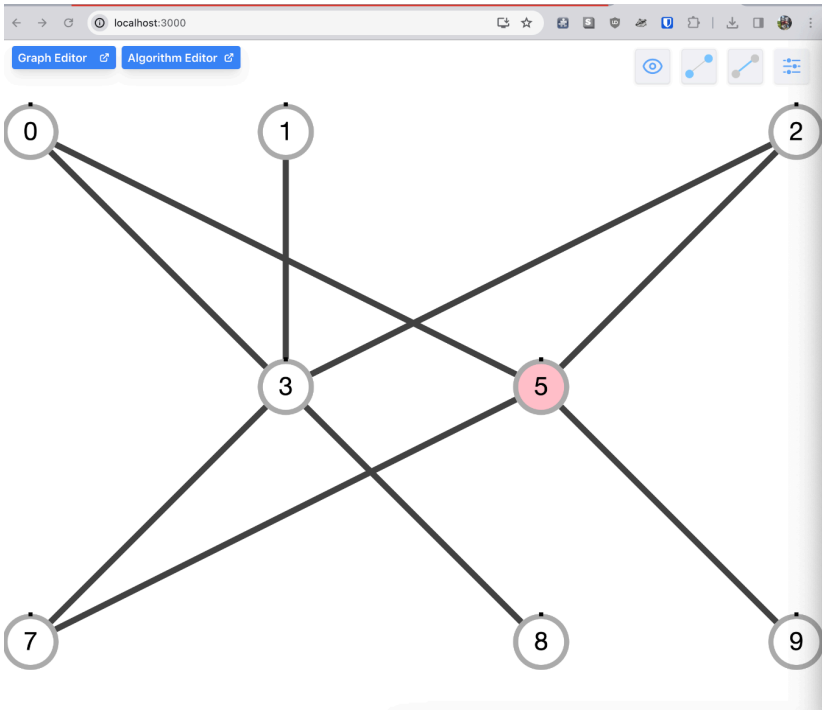
1. Ensure that a graph has been successfully loaded in the main window.
2. Any changes made in Graph Editor will be reflected in the graph currently loaded in the main window. As seen below, the color for node number five is blue. To change this color we need to update its node attribute of ‘color’.



3. In the Graph text editor, change the ‘color:blue’ to ‘color:pink’.

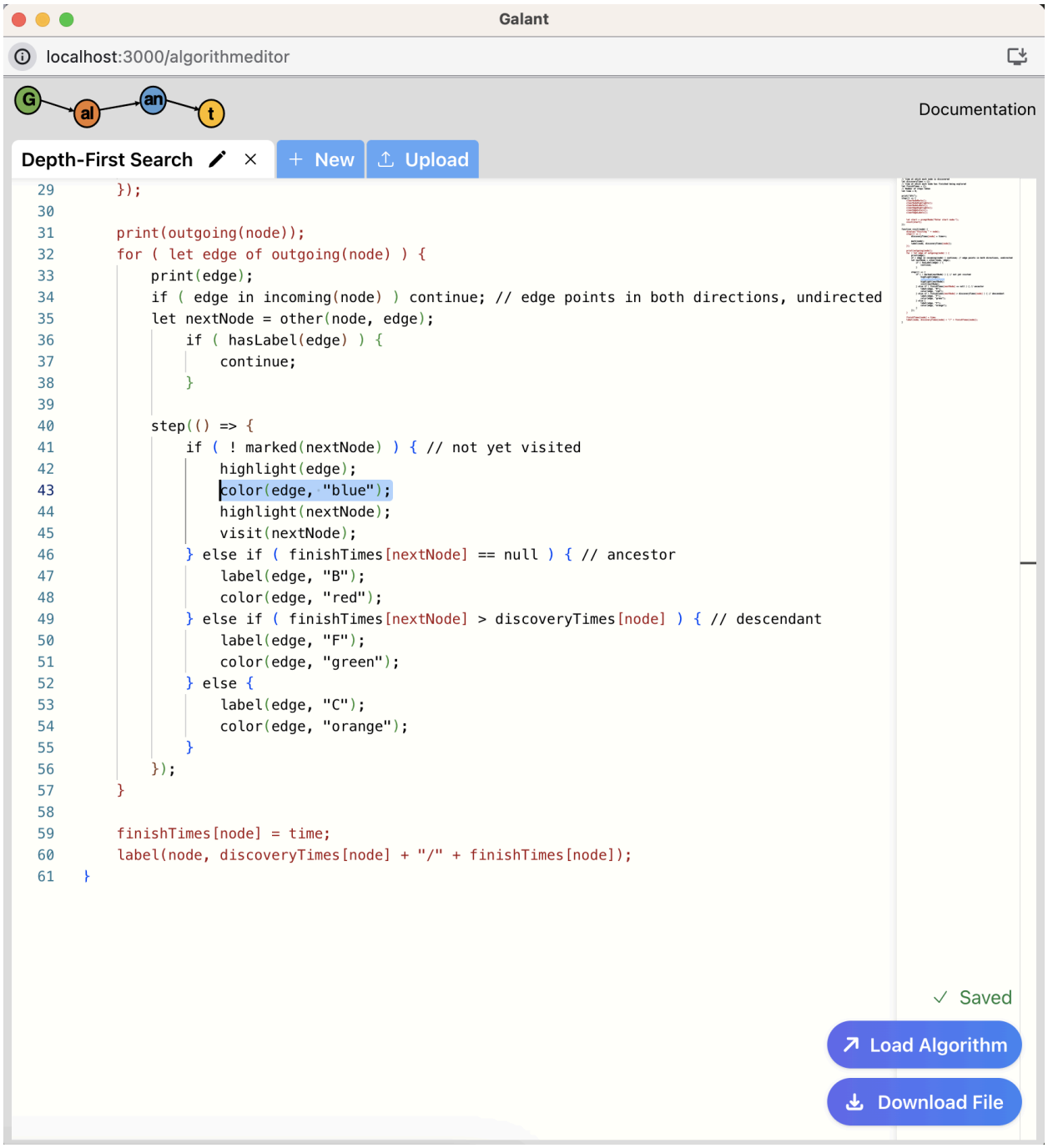


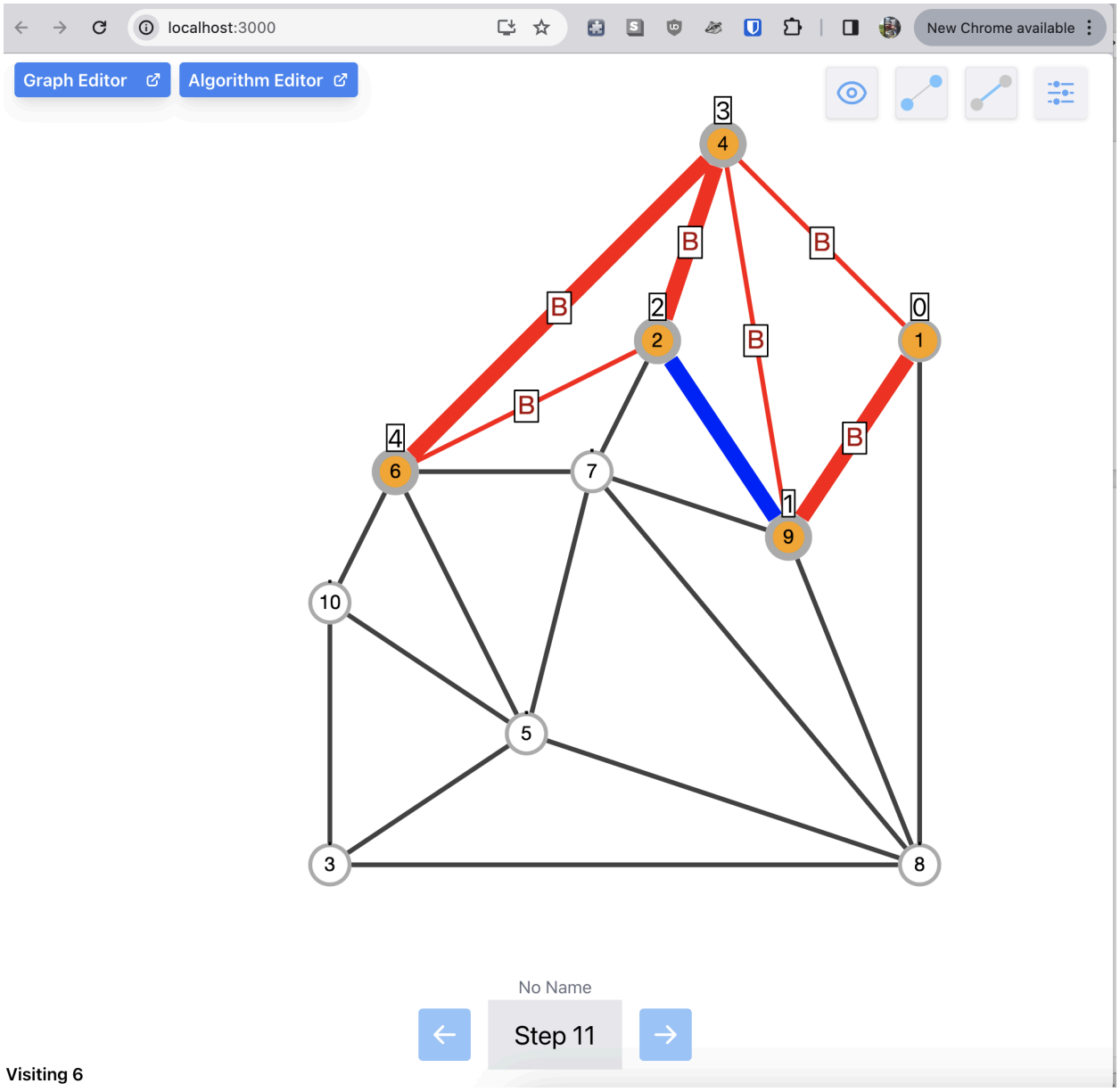
3. Load the graph and reflect on the changes.


localhost:3000
Graph Editor
Algorithm Editor
0
1
2
3
5
7
8
9

Editing Algorithm Text

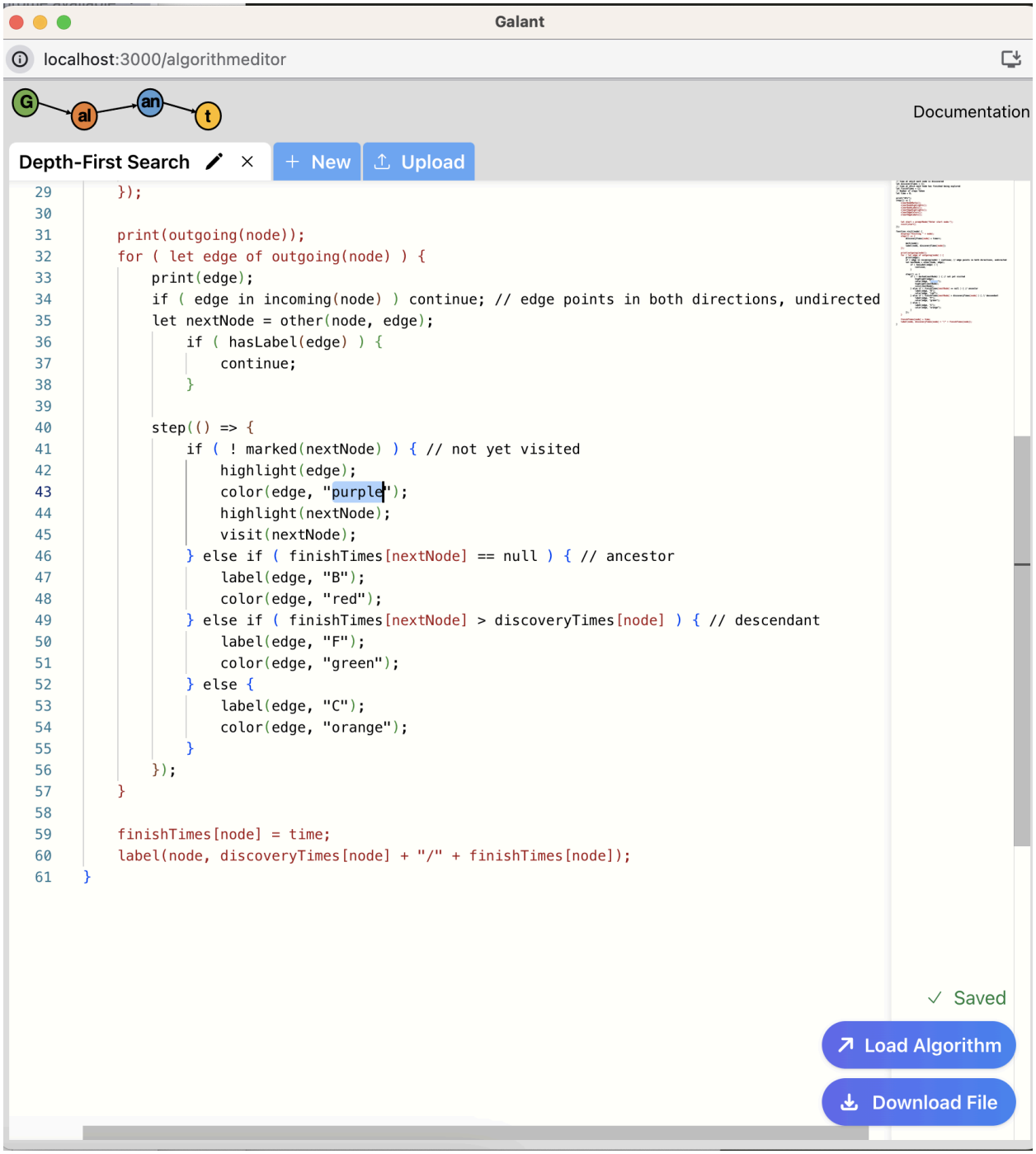
1. Ensure that an algorithm has been successfully loaded in the main window.
2. Any changes made in Algorithm Editor will be reflected in the graph currently loaded in the main window. As seen below, the edge highlight color for an edge that has not been visited yet is blue. But what if we want to change the color?

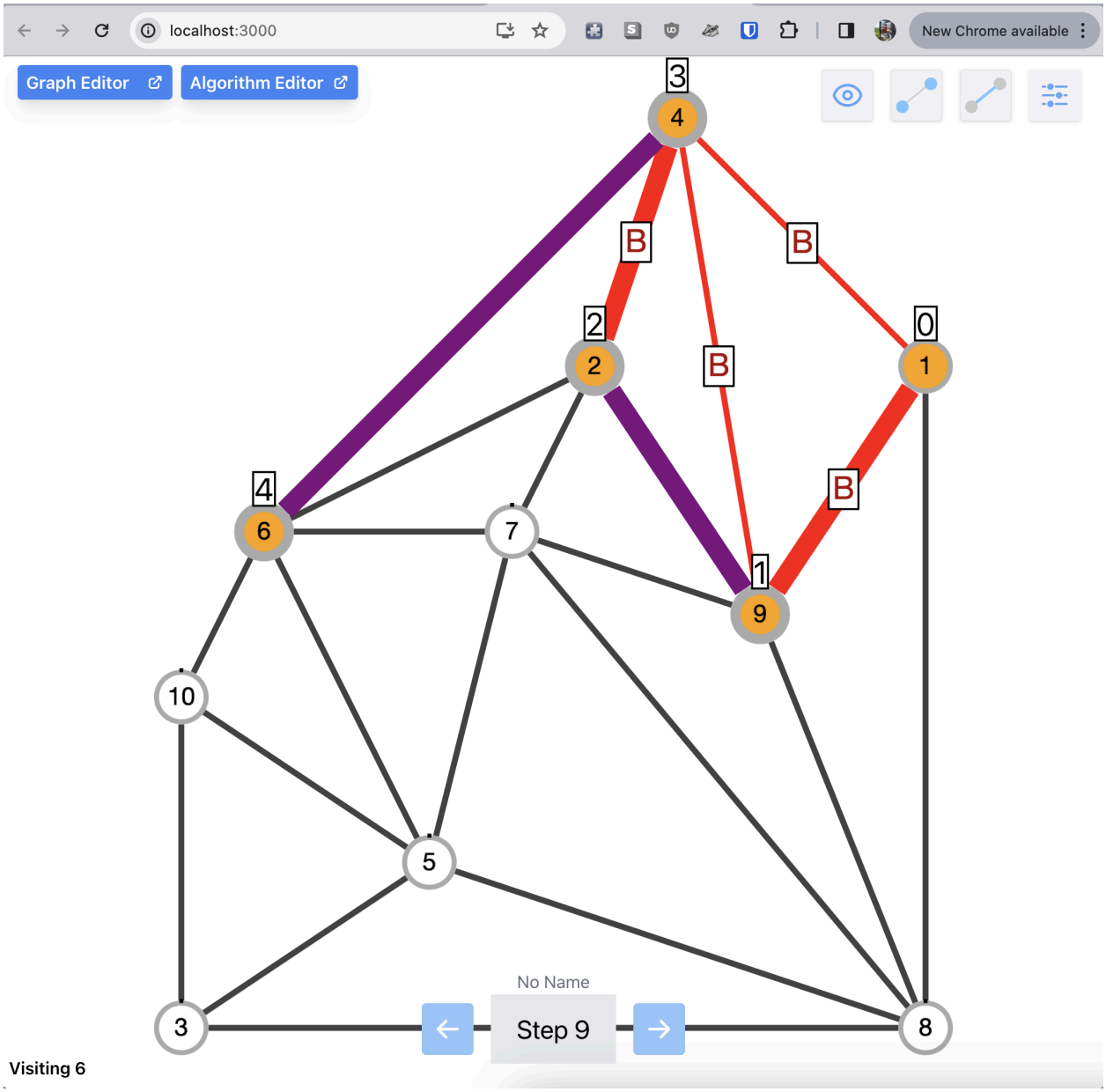



localhost:3000
New Chrome available
Graph Editor
Algorithm Editor
3
4
B
B
B
2
2
B
B
0
1
B
4
6
7
1
9
10
5
3
8
No Name
Step 11

Visiting 6

3. For example, to change the edge highlight color on an Unweighted graph from blue to purple we can edit the Depth-First Search algorithm in the Algorithm Editor window. Then click the Load Algorithm button again to reflect these changes.




localhost:3000
New Chrome available
Graph Editor
Algorithm Editor
3
4
B
B
2
2
B
0
1
4
6
7
B
1
9
10
5
No Name
3
Step 9
8

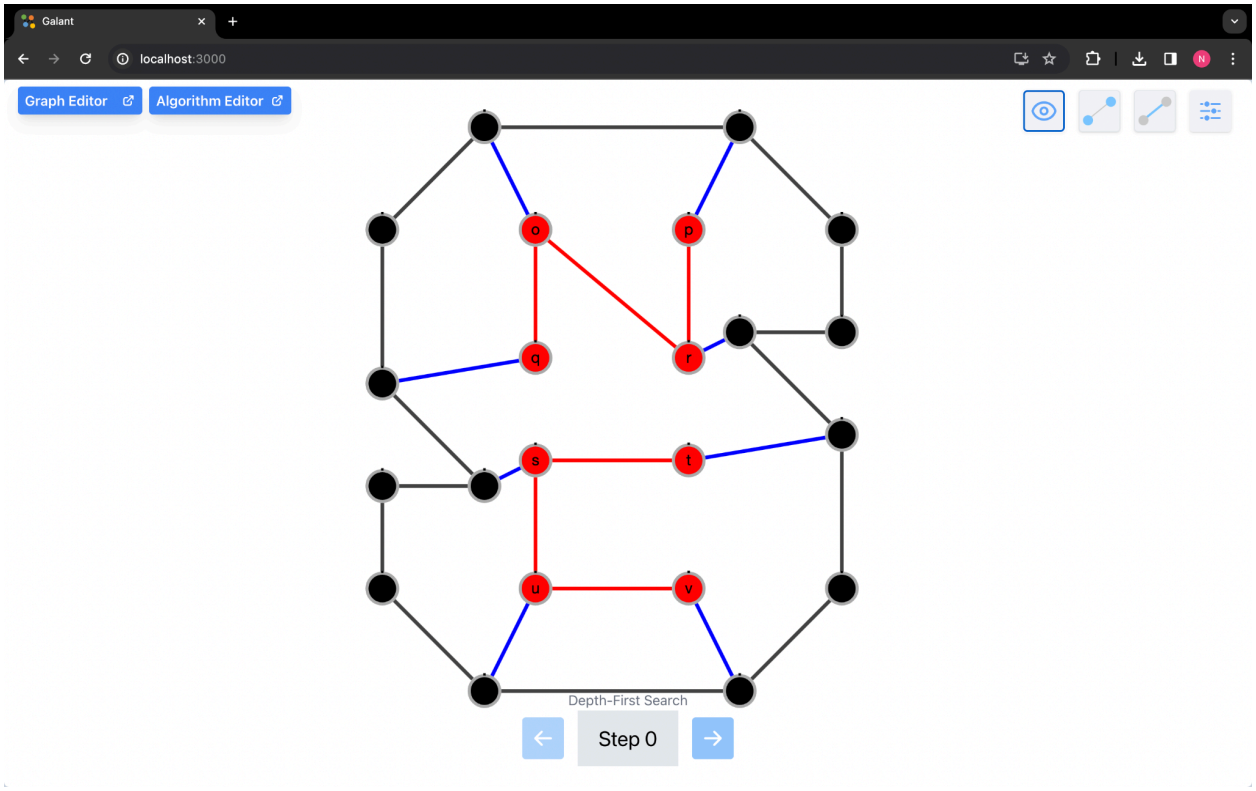
Visiting 6

Keyboard Shortcuts

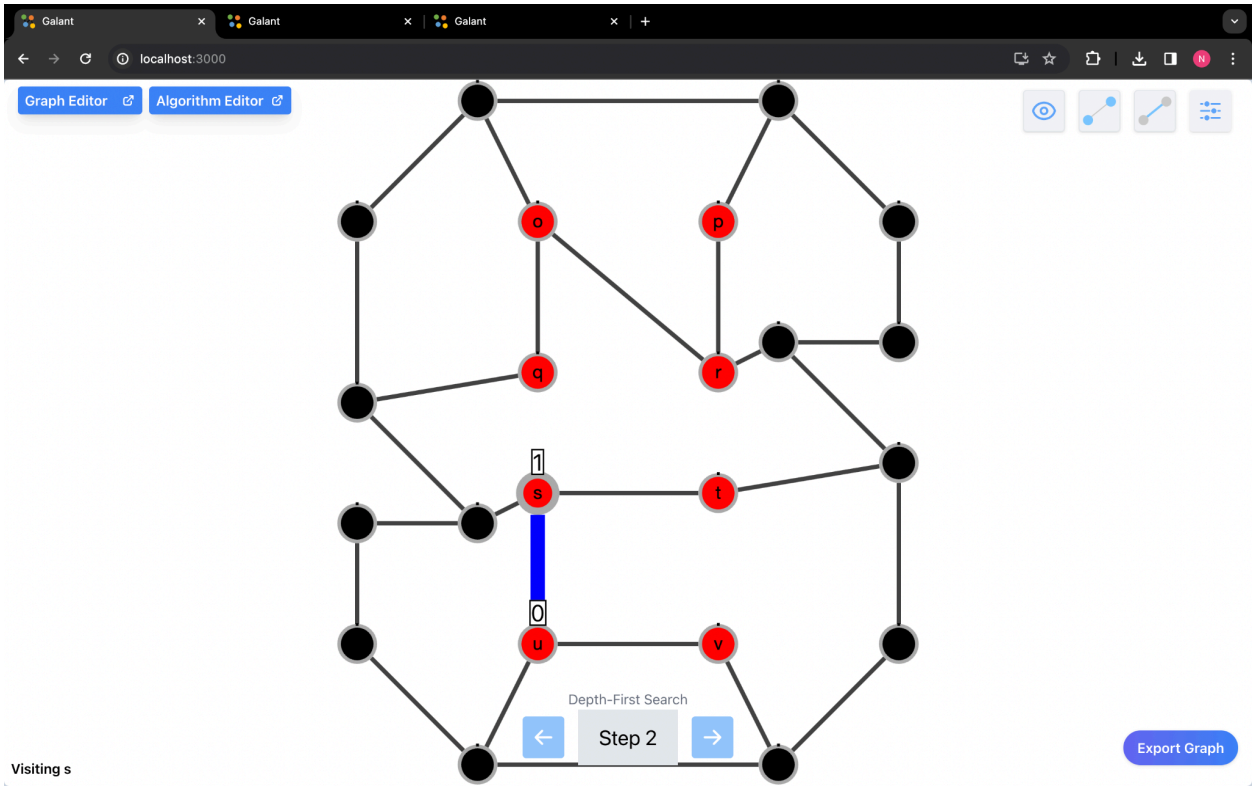
Table 1: Keyboard Shortcuts

Keyboard Shortcut	Action	Description
A	Opening Editors	Open Algorithm Editor from Main Page
G	Opening Editors	Open Graph Editor from Main Page
S	In-Editor Controls	Download Current Graph/Algorithm File in the Editor
L	In-Editor Controls	Load Current Graph/Algorithm File in the Editor
N	In-Editor Controls	Choose New Graph/Algorithm File in the Editor
Z	Edit Mode	Undo Changes
X	Edit Mode	Redo Changes
R	Edit Mode	Revert Graph back to original state
S	Edit Mode	Save Changes
Left Arrow	Algorithm Controls	Step Back
Right Arrow	Algorithm Controls	Step Forward
Escape Key	Algorithm Controls	Terminate Algorithm
‘S’ Key	Algorithm Controls	Export Graph
Cmd/Win + Right Arrow	Algorithm Controls	Skip to the End of the Algorithm
C	Control Settings Popover	Open popover for Layout Controls
N	Node	Open popover button for Node Settings

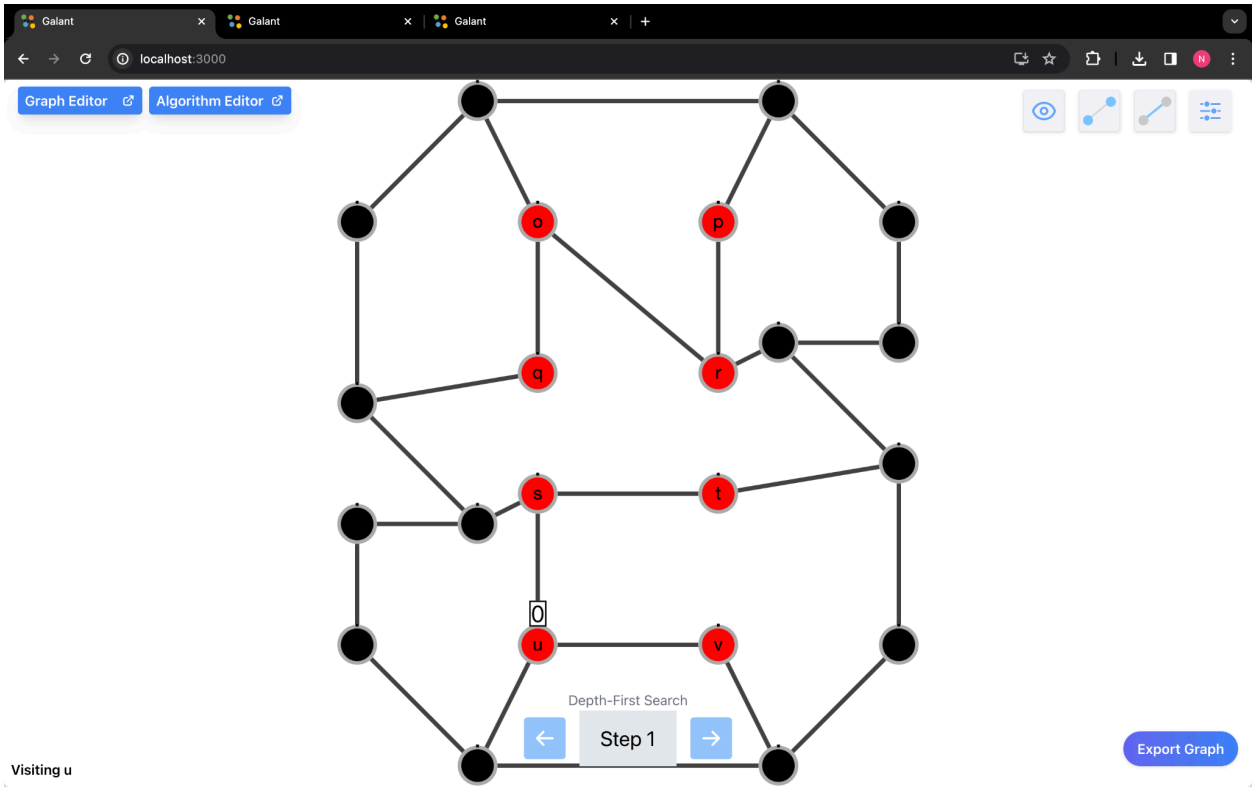
	Settings Popover	
E	Edge Settings Popover	Open popover button for Edge Settings
P	Preference Popover	Open popover button for Preference Panel
H	Help	Open help button



1. The left and right arrow keys can be used to step through the algorithm once it has been loaded. The right arrow key takes the graph a step further through the algorithm.

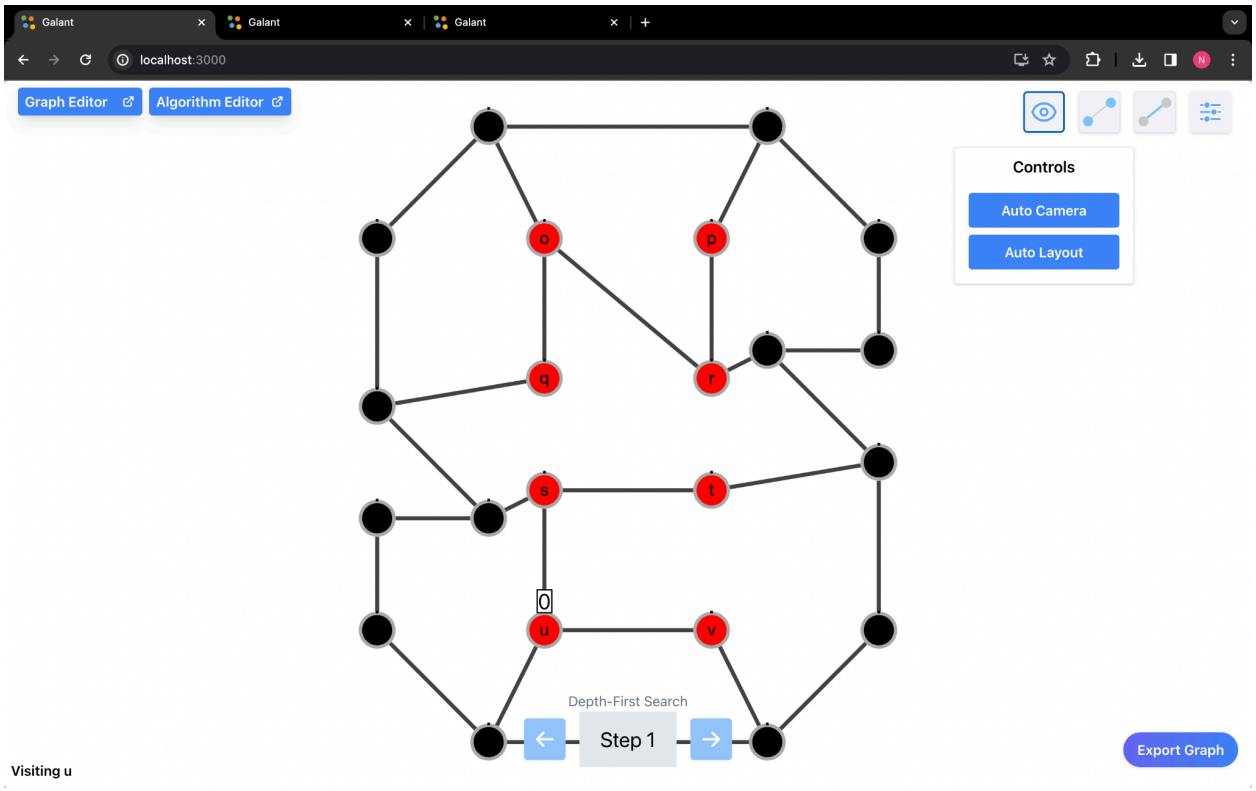


The left arrow key takes the graph a step prior to the current step in the algorithm.

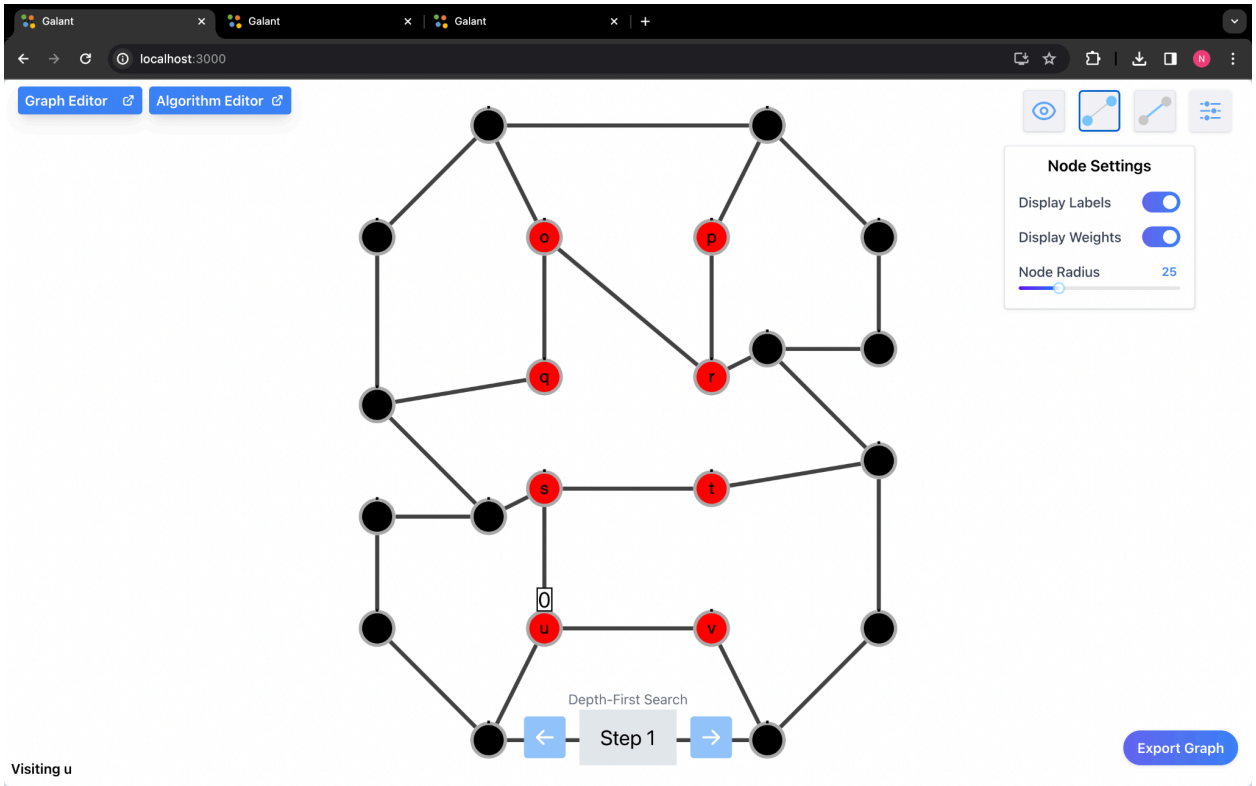


2. The Enter key allows the user to start executing the algorithm immediately after being loaded. The user can load a graph, then load an algorithm, and then press enter to start the algorithm.

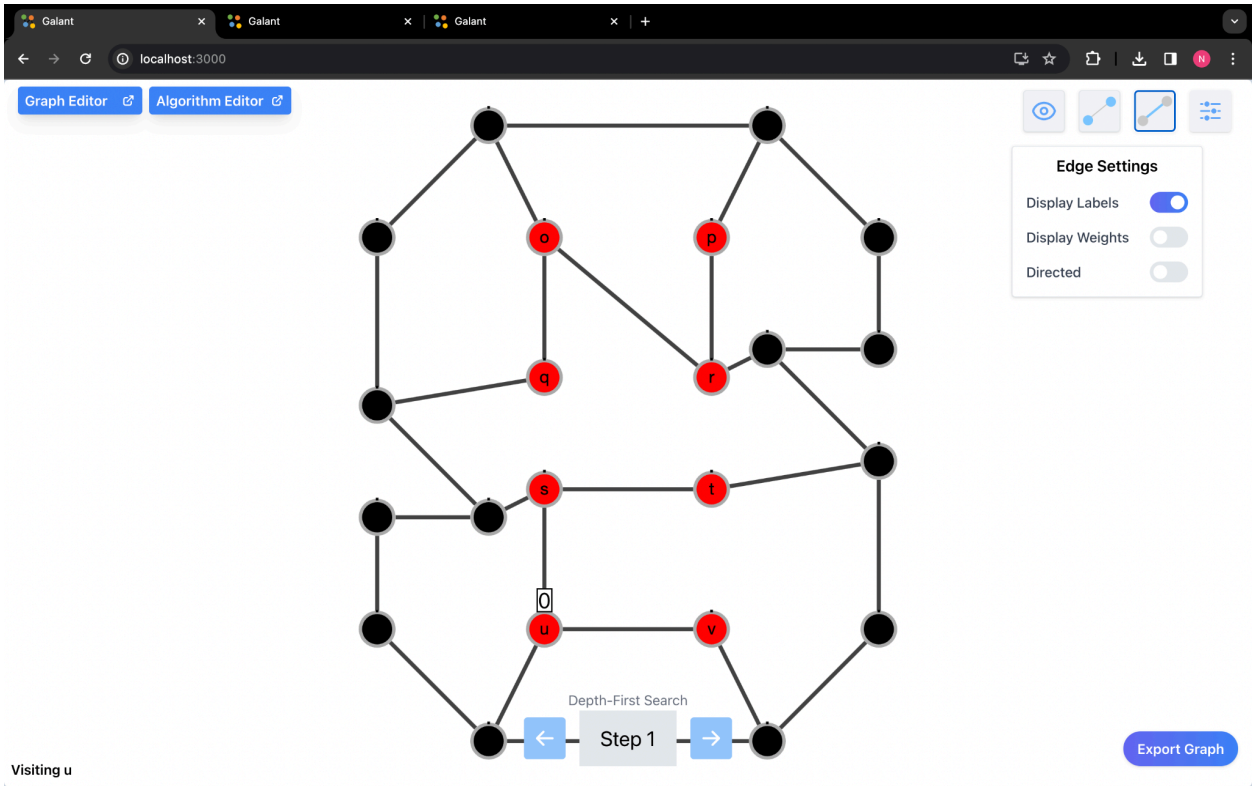
3. The toggles on the top right can be triggered by keyboard shortcuts. The controls can be seen by pressing the c key on the keyboard.



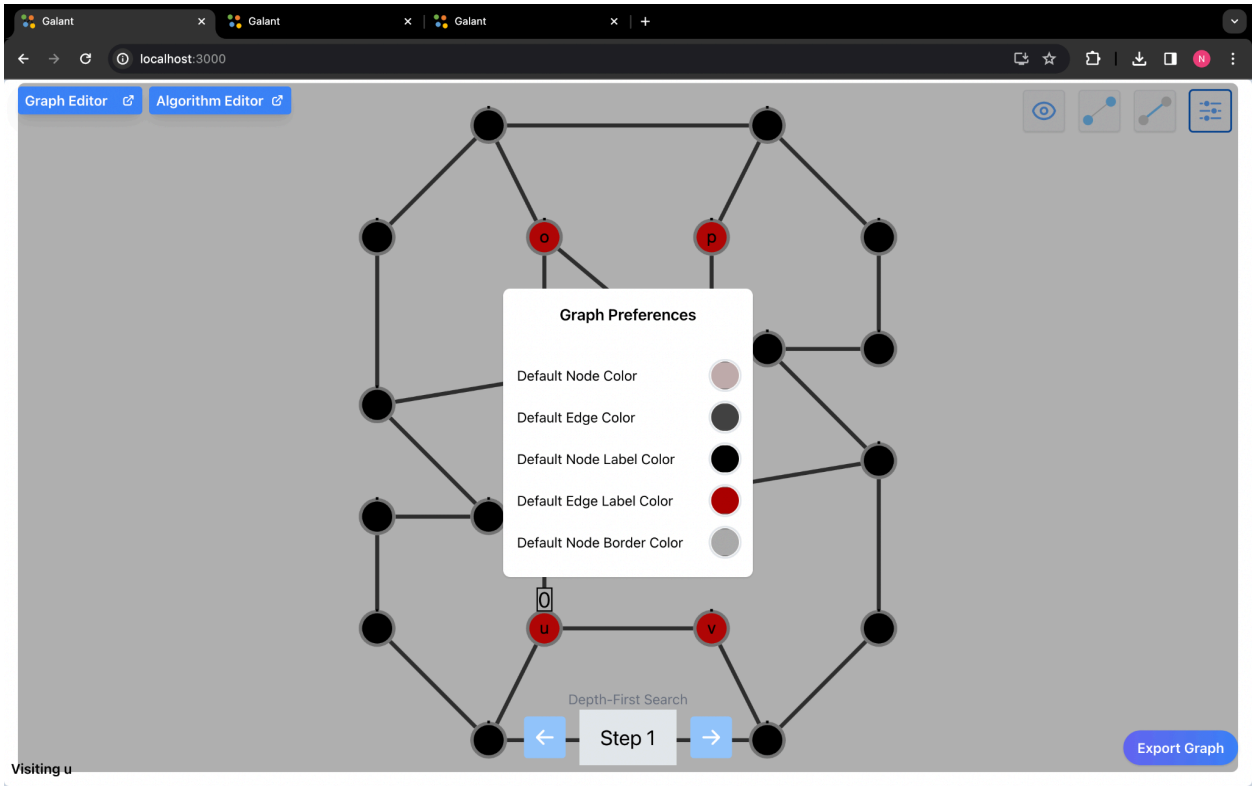
4. The node settings can be seen by pressing the n key on the keyboard



5. The edge settings can be seen by pressing e on the keyboard

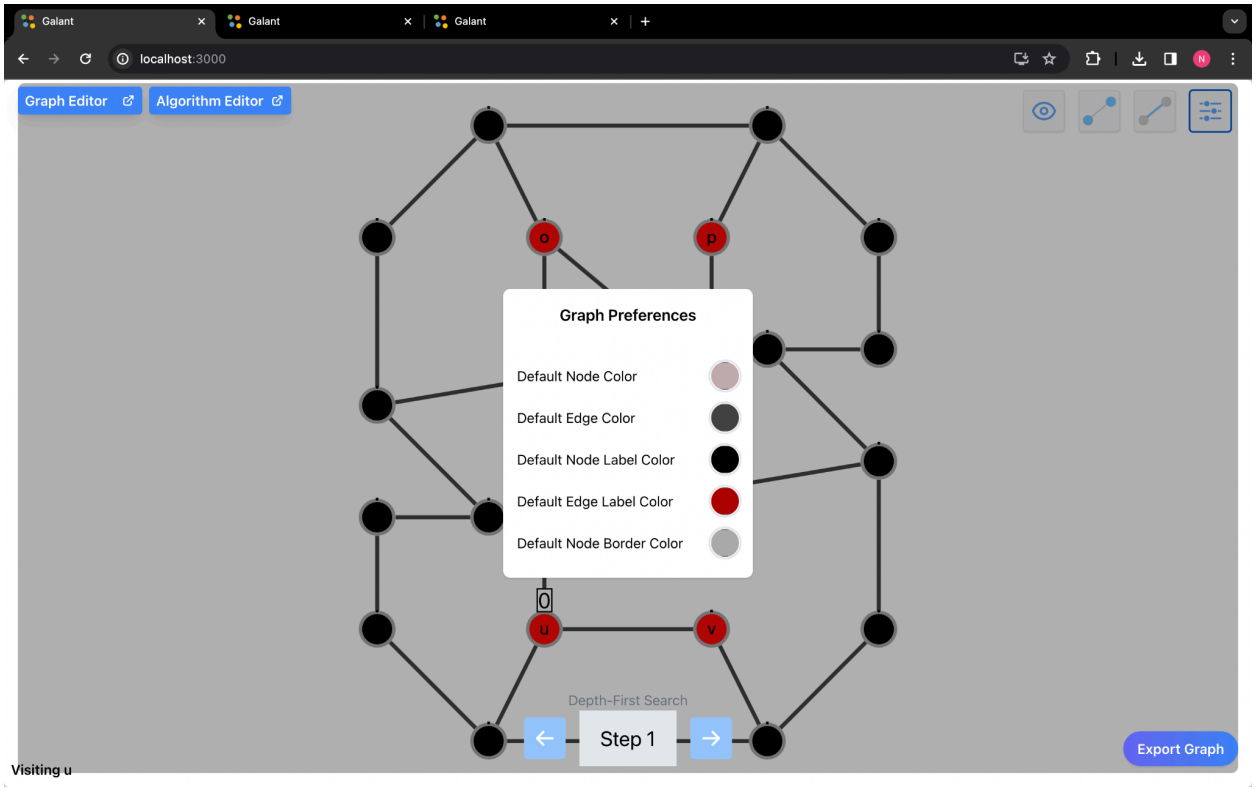


6. The graph preferences can be seen by pressing p on the keyboard

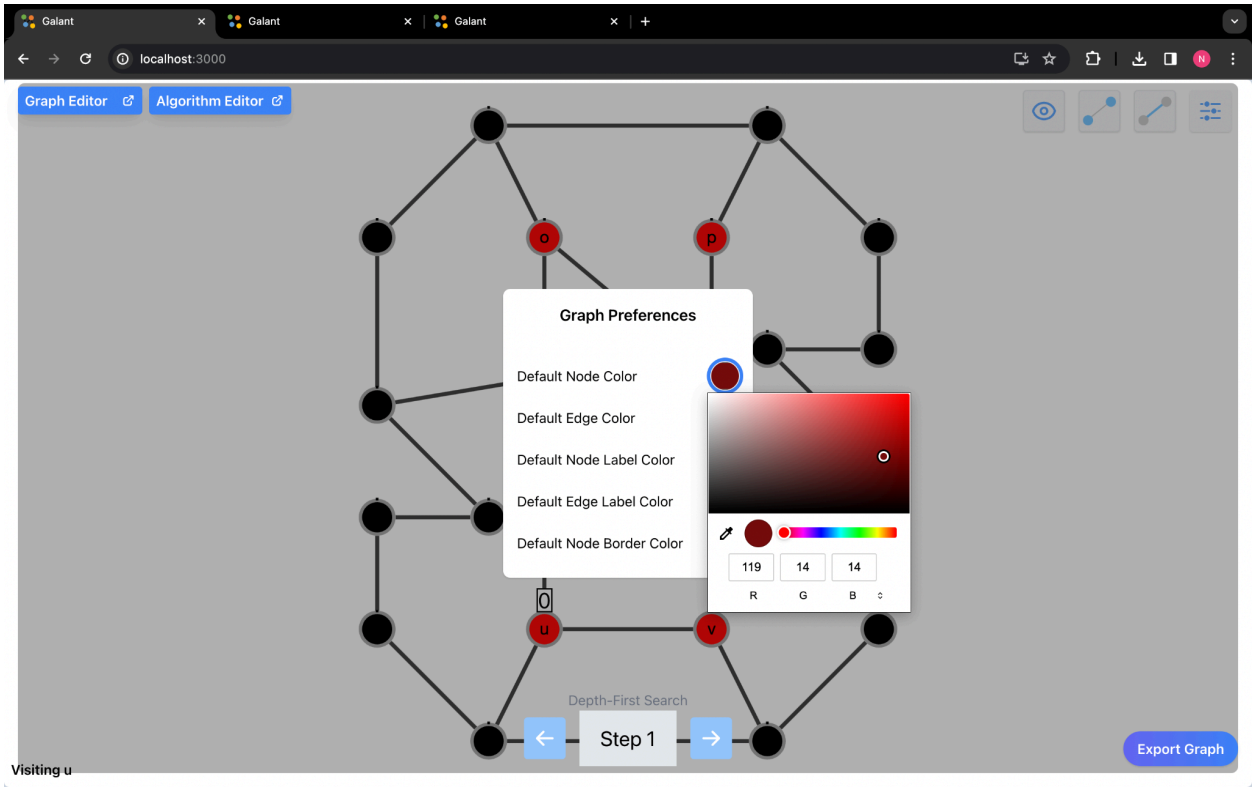

Galant
Galant
Galant
localhost:3000
Graph Editor
Algorithm Editor
o
p
Graph Preferences
Default Node Color
Default Edge Color
Default Node Label Color
Default Edge Label Color
Default Node Border Color
0
u
v
Depth-First Search
Step 1
Export Graph
Visiting u

Preference Panel

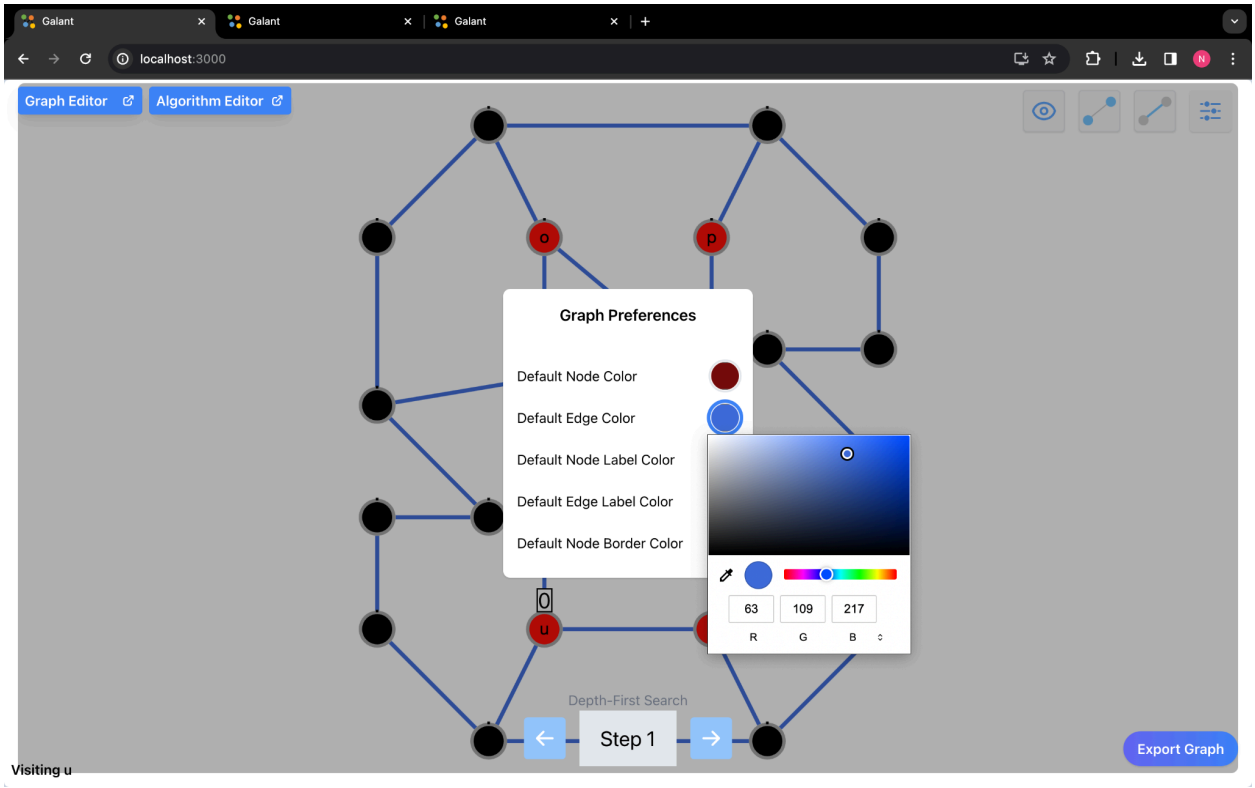
1.The Graph Preferences Panel allows the user to change the node and edge colors and labels. To access the panel you can either press p as demonstrated above or click on the icon with the slides



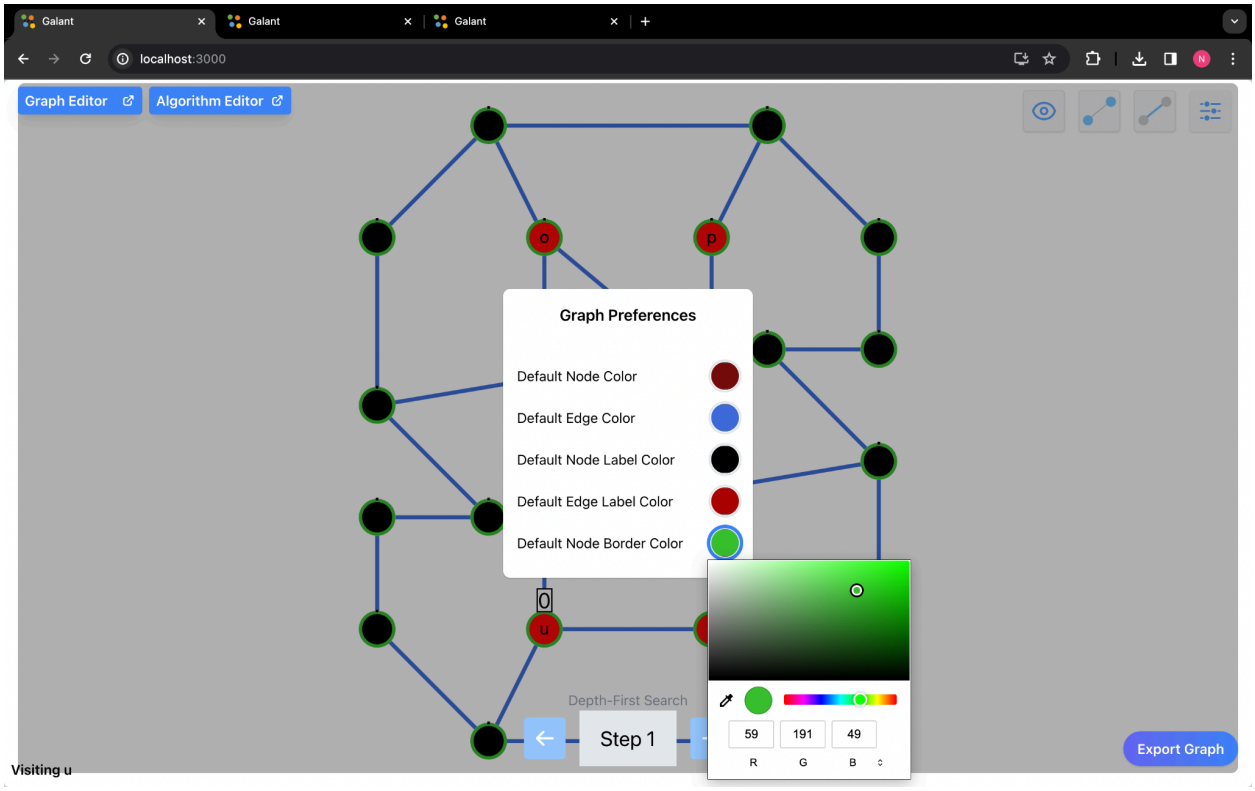
a. to change the color of the node you can click on the circle next to the label and choose the color you want to change it to



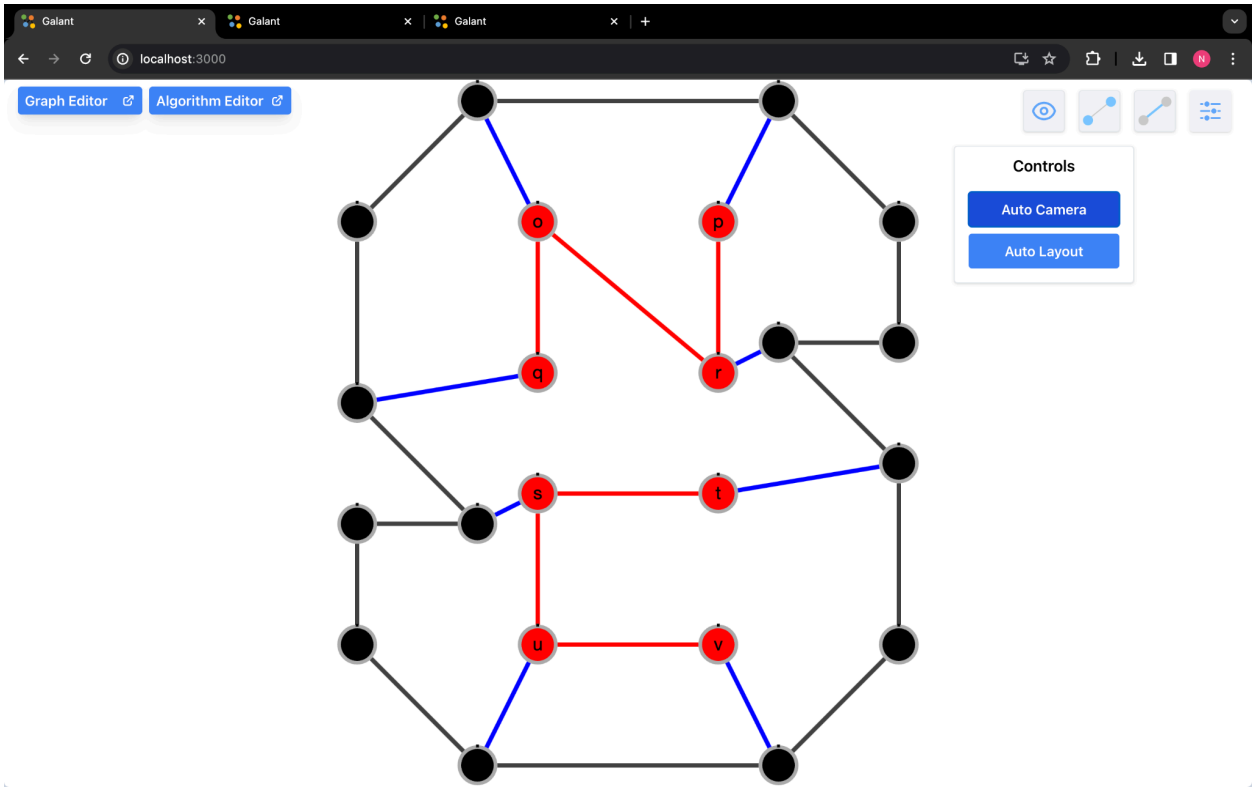
b. to change the color of the edge you can click on the circle next to the label and choose the color you want to change it to



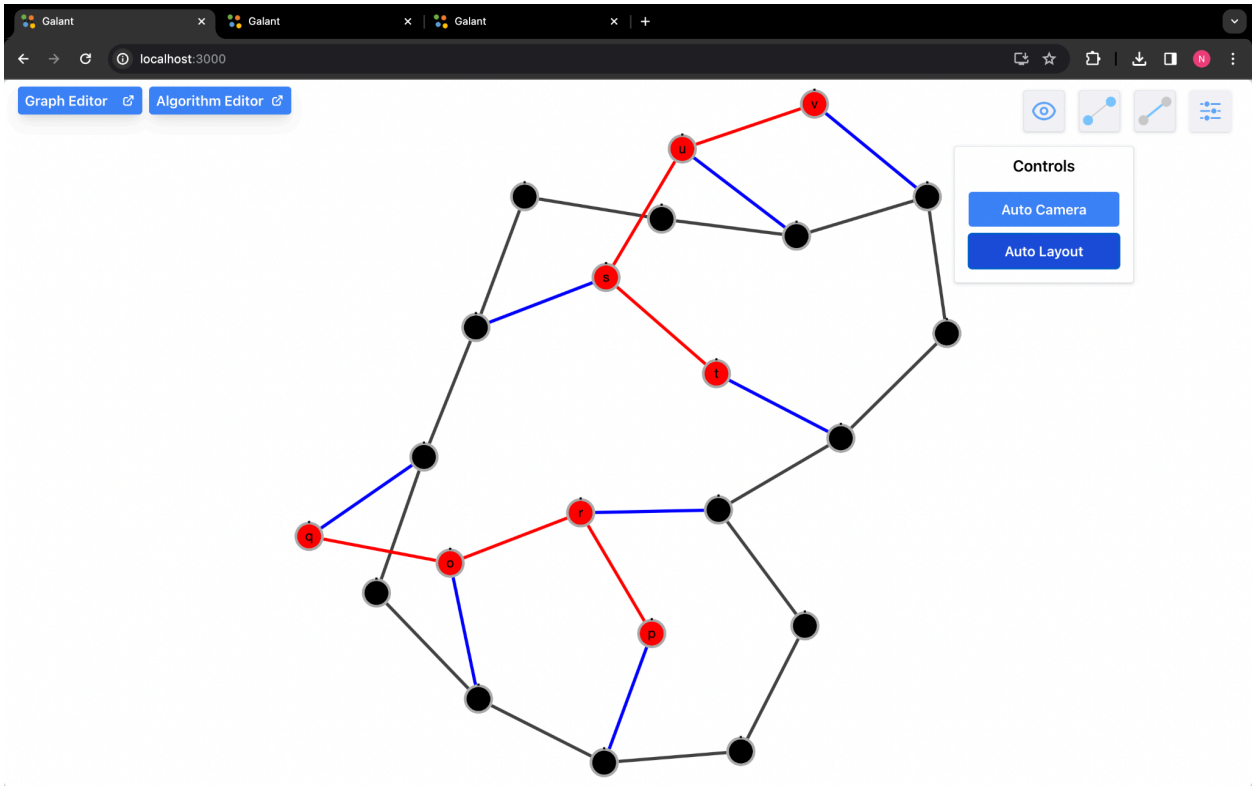
c. to change the color of the label you can click on the circle next to the label and choose the color you want to change it to



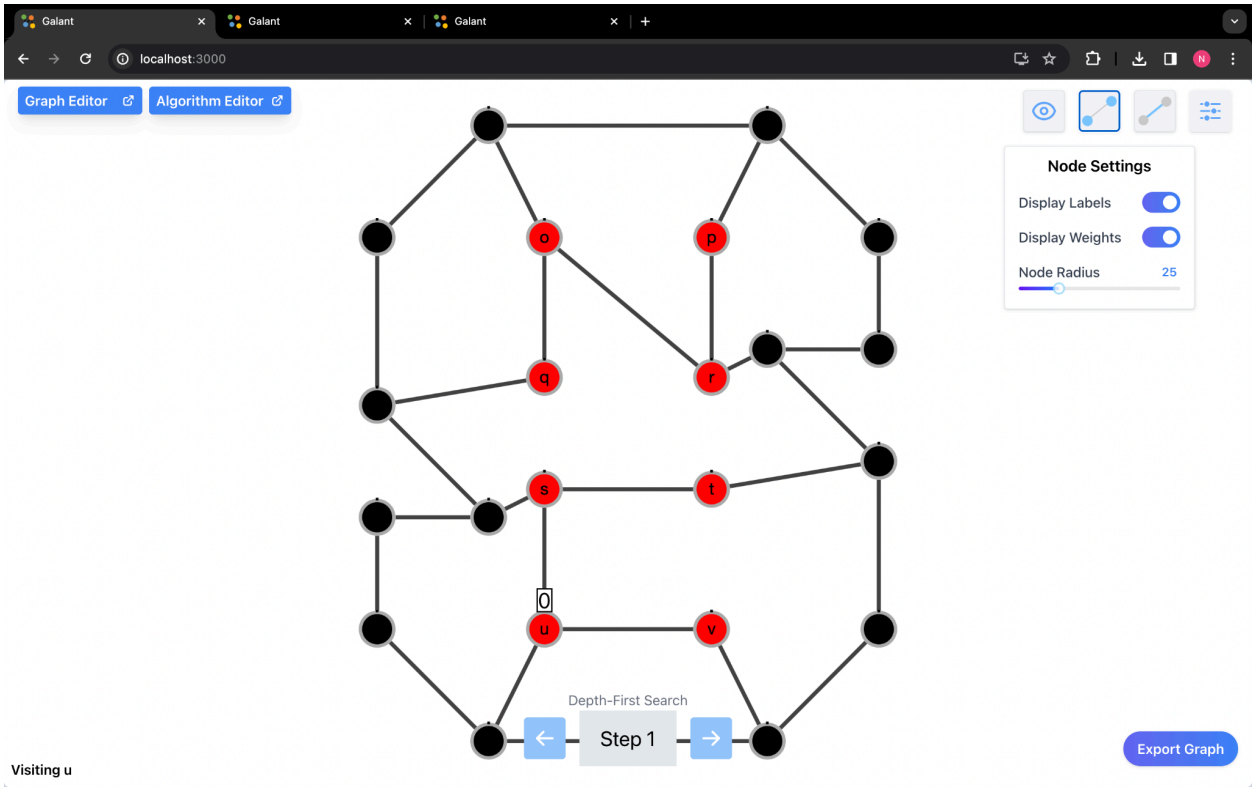
2. The user can access the controls by pressing c or clicking on the eye icon.
    a. Clicking on auto camera fits the graph to the page



    b. Auto layout changes the layout of the graph



3. The user can access the node settings by pressing the n key or clicking on the line segment icon with the endpoints highlighted.



Algorithm Methods

List Getters

These functions get lists of all nodes or edges in the graph to iterate over.

Function Name	Description
getNodes()	Returns a list of all nodes in the graph.
getNumberOfNodes()	Returns the number of nodes in the graph. Can also use getNodes().length.
getEdges(	Returns a list of all edges in the graph to iterate over.
getNumberOfEdges()	Returns the number of nodes in the graph. Can also use getEdges().length

Lists Example Algorithm

```
	//Prints each node and then each edge to the console.
for (let node of getNodes()) {
	print(node);
}
for (let edge of getEdges()) {
	print(edge);
}
```

Source and Target

These functions select nodes or edges based on the source and target of edges.
Note that for undirected graphs, the source and target are interchangeable.

Function Name	Description
source(edge)	Returns the source node of the specified edge.
target(edge)	Returns the target node of the specified edge.

getEdgeBetween(source, target)	Returns an edge between two nodes. For directed graphs, only the edge in the direction specified will be returned. If no edge exists between the specified nodes, returns null.
getEdgesBetween(source, target)	Returns a list of edges between two nodes. For directed graphs, only the edges in the direction specified will be returned. This function is only needed if a graph contains multiple edges between the same nodes.
other(node, edge)	Returns the other node attached to the edge. The node and edge parameters to this function may be provided in either order.

Source and Target Example Algorithm

```
//This function returns the target of edge A B by finding the node opposite its source.
let edge = getEdgeBetween(“A”, “B”);
let source = source(edge);
print(other(source, edge));
```

Adjacencies

These functions get lists of edges incident to a node to iterate over. In an undirected graph, all incoming and outgoing functions are equivalent to their non-directional counterparts.

Function Name	Description
incident(node)	Returns all edges (incoming and outgoing) incident to a node.
incoming(node)	Returns all incoming edges of a node (the node is their target).
outgoing(node)	Returns all outgoing edges of a node (the node is their source).

	These functions get lists of nodes adjacent to a node to iterate over.
adjacentNodes(node)	Returns all nodes adjacent to the node.
incomingNodes(node)	Returns all nodes connected to the node by an incoming edge.
outgoingNodes(node)	Returns all nodes connected to the node by an outgoing edge. These functions get the number of edges incident to a node.
degree(node)	Returns the number of edges incident to a node.
inDegree(node)	Returns the number of incoming edges of a node.
outDegree(node)	Returns the number of outgoing edges of a node.

Adjacencies Example Algorithm

```
//This function prints each edge out of the node with the max number of outgoing edges.
let max = 0;
let maxNode = null;
for (let node of getNodes()) {
        if (outDegree(node) > max) {
        max = outDegree(node);
        maxNode = node;
        }
}
for (let edge of outgoing(maxNode)) {
        let node = other(maxNode, edge);
        print(edge);
        print(node);
}
```

Marks

Function Name	Description
mark(node)	Marks the specified node
unmark(node)	Unmarks the specified node
marked(node)	Returns true if the node is marked, false otherwise
clearNodeMarks()	Clears marks on all nodes

Marks Example Algorithm

```
for (let node of getNodes()) {
        mark(node);
}
```

Highlights

Function Name	Description
highlight(id)	Highlights the specified node or edge based on the id
unhighlight(id)	Unhighlights the specified node or edge based on the id
highlighted(id)	Returns true if the node or edge is highlighted, false otherwise
clearNodeHighlights()	Clears highlights on all nodes

Highlights Example Algorithm

```
        //Highlights all edges
for (let edge of getEdges()) {
        highlight(edge);
}
```

Colors

Function Name	Description
color(id, color)	Colors the specified node or edge based on its id and given color
uncolor(id)	Uncolors the specified node or edge based on the id
getColor(id)	Returns true if the node or edge is colored, false otherwise
hasColor(id)	Returns true if the node or edge based on the id is colored
clearNodeColors()	Clears colors on all nodes
clearEdgeColors()	Clears colors on all edges

Colors Example Algorithm

clearNodeColors()

Labels

Function Name	Description
label(id, label)	Labels the specified node or edge based on its id and given label
unlabel(id)	Unlabels the specified node or edge based on the id
getLabel(id)	Returns the labels of the specified node or edge based on its id
hasLabel(id)	Returns true if the node or edge has a label based on the id

clearNodeLabels()	Clears labels on all nodes
clearEdgeLabels()	Clears labels on all edges

Labels Example Algorithm

```
        //Un-labels each label of all edges (40 and 39 are weights on the edges, not labels)
for (let edge of getEdges()) {
        unlabel(edge);
}
display("Removed all edge labels")
```

Weights

Function Name	Description
setWeight(id, weight)	Sets the weight of the specified node or edge based on its id and given weight
clearWeight(id)	Clears the weight of the node or edge based on the id
weight(id)	Returns the weight of the specified node or edge based on the id
hasWeight(id)	Returns true if the node or edge based on the id has a weight
clearNodeWeights()	Clears weights on all nodes
clearEdgeWeights()	Clears weights on all edges

Weights Example Algorithm

```
        //Sets the weight of each node to 10
for (let node of getNodes()) {
        setWeight(node, 10);
}
```

Hide/Show Node Properties

Function Name	Description
hideNode(node)	Hides the node based on the specified node id (also hides connecting edges to that node)
showNode(node)	Shows the node based on the specified node id
hideNodeWeight(node)	Hides the node weight based on the specified node id
hideAllNodeWeights()	Hides all node weights
showNodeWeight(node)	Shows the node weight based on the specified node id
showAllNodeWeights()	Shows all node weights
hideNodeLabel(node)	Hides the node label based on the specified node id
hideAllNodeLabels()	Hides all node labels
showNodeLabel(node)	Shows the node label based on the specified node id
showAllNodeLabels()	Shows all node labels

Node Properties Example Algorithm

```
	//Hides each node in the graph individually.
for (let node of getNodes()) {
	hideNode(node);
}
display("Hid every node");
```

Hide/Show Edge Properties

Function Name	Description
hideEdge(edge)	Hides the edge based on the specified edge id
showEdge(edge)	Shows the edge based on the specified edge id
hideEdgeWeight(edge)	Hides the edge weight based on the specified edge id
hideAllEdgeWeights()	Hides all edge weights
showEdgeWeight(edge)	Shows the edge weight based on the specified edge id
showAllEdgeWeights()	Shows all edge weights
hideEdgeLabel(edge)	Hides the edge label based on the specified edge id
hideAllEdgeLabels()	Hides all edge labels
showEdgeLabel(edge)	Shows the edge label based on the specified edge id
showAllEdgeLabels()	Shows all edge labels

Edge Properties Example Algorithm

```
//Hides each edge weight in the graph individually.
for (let edge of getEdges()) {
        hideEdgeWeight(edge);
}
display("Hid every edge weight");
```

Node Shape

Function Name	Description
shape(id)	Returns the shape of the specified node based on the id
setShape(id, shape)	Sets the shape of the specified node based on its id and given weight
clearShape(id)	Clears the shape of the node based on the id
hasShape(id)	Returns true if the node based on the id has a shape
clearNodeShapes()	Clears shapes on all nodes

Node Shape Example Algorithm

```
for (let nodeId of getNodes()) {
        setShape(nodeId,”star”);
}
```

Node Border Width

Function Name	Description
borderWidth(id)	Returns the border width of the specified node based on the id
setBorderWidth(id, borderWidth)	Sets the border width of the specified node based on its id and given weight
clearBorderWidth(id)	Clears the border width of the node based on the id
hasBorderWidth(id)	Returns true if the node based on the id has a border width
clearNodeBorderWidth()	Clears border width on all nodes

Node Border Width Example Algorithm

```
for (let nodeId of getNodes()) {
        setShape(nodeId,5);
}
```

Node Shading

Function Name	Description
backgroundOpacity(id)	Returns the background opacity of the specified node based on the id
setBackgroundOpacity(id, backgroundOpacity)	Sets the background opacity of the specified node based on its id and given weight
clearBackgroundOpacity(id)	Clears the background opacity of the node based on the id
hasBackgroundOpacity(id)	Returns true if the node based on the id has a background opacity
clearNodeBackgroundOpacity()	Clears background opacity on all nodes

Node Shading Example Algorithm

```
for (let nodeId of getNodes()) {
        setShape(nodeId,0.5);
}
```

Node Size

Function Name	Description
size(id)	Returns the size of the specified node based on the id
setSize(id, size)	Sets the size of the specified node based on its id and given weight
clearSize(id)	Clears the size of the node based on the id
hasSize(id)	Returns true if the node based on the id has a size
clearNodeSizes()	Clears size on all nodes

Node Size Example Algorithm

```
for (let nodeId of getNodes()) {
        setShape(nodeId,40);
}
```

Edge Width

Function Name	Description
edgeWidth(id)	Returns the edge width of the specified edge based on the id
setEdgeWidth(id, width)	Sets the edge width of the specified edge based on its id and given weight
clearEdgeWidth(id)	Clears the edge width of the edge based on the id
hasEdgeWidth(id)	Returns true if the edge based on the id has a edge width
clearNodeEdgeWidth()	Clears edge width on all edges

Edge Width Example Algorithm

```
for (let edgeId of getEdges()) {
        setShape(edgeId,10);
}
```