### **Algorithm / Flowchart for Interactive Graph Algorithms Visualizer**

The following steps outline the algorithm required to build the *Interactive Graph Algorithms Visualizer*.

**Step 1: Initialize Project Setup**

* Begin by setting up the project environment.
* Create a user interface where students can interact with nodes, edges, and algorithm options.
* Define the necessary graph data structure to hold node and edge information.

**Step 2: Enable Node Addition**

* Allow users to click on the canvas area to add nodes.
* Each new node should be assigned a unique identifier and a visual position on the canvas.
* Update the graph data structure to include the new node.

**Step 3: Enable Edge Creation**

* Provide an option for users to select two nodes to create an edge between them.
* Validate that the selected nodes are not the same (preventing self-loops, if applicable).
* Update the graph data structure to include the edge, storing both direction and weight if it’s a weighted graph.

**Step 4: Choose Algorithm**

* Present users with a list of algorithms (BFS, DFS, Dijkstra’s) to select from.
* Based on user selection, initialize the specific algorithm’s setup, which may include setting a starting node for traversal algorithms or setting edge weights for shortest path algorithms.

**Step 5: Display Real-Time Visualization Setup**

* Configure the canvas to visualize each step of the selected algorithm.
* Prepare necessary visual elements (e.g., colors or animations) to distinguish nodes and edges as they are processed by the algorithm.

**Step 6: Execute the Algorithm with Step-by-Step Visualization**

* Begin execution of the chosen algorithm, updating the visualization in real time:
  + **BFS:** Highlight each node and edge as they are visited in breadth-first order.
  + **DFS:** Highlight each node and edge in depth-first order.
  + **Dijkstra’s:** Display shortest path calculations by progressively updating edge weights and visited nodes.
* At each step, update the graph view to reflect current progress, making the algorithm's inner workings visible to the user.

**Step 7: Handle Completion of Algorithm**

* Upon completion, highlight the final path or traversal order depending on the algorithm:
  + For BFS/DFS, show the order in which nodes were visited.
  + For Dijkstra’s, highlight the shortest path found and display total path weight.

**Step 8: Enable User Interaction Post-Algorithm**

* Allow users to reset the graph for another run, clear current visualizations, or select a different algorithm for testing.
* Ensure users can add new nodes/edges or remove existing ones to create different graph configurations.

**Step 9: Display Summary of Results**

* Once the algorithm completes, present a summary:
  + For BFS/DFS, show the traversal sequence.
  + For Dijkstra’s, list the shortest paths from the start node to all reachable nodes.
* Optionally, provide brief feedback or tips for understanding the algorithm’s result and logic.

**Step 10: Offer Option to Save or Restart**

* Allow users to save the current graph and results for reference.
* Provide a reset button to clear the canvas and start over with new inputs and configurations.

This flow ensures a comprehensive and user-friendly experience, helping students visualize graph algorithms interactively while reinforcing their understanding of data structures and algorithmic logic. Each component of the algorithm is structured to facilitate smooth interaction, algorithm execution, and learning.