**Name:** Dev Detroja, Darsh Jodhani

**Subject:** Graph Theory Innovative Assignment

**Topic:** Max-Flow

**Roll No.:** 22BCE065, 22BCE129

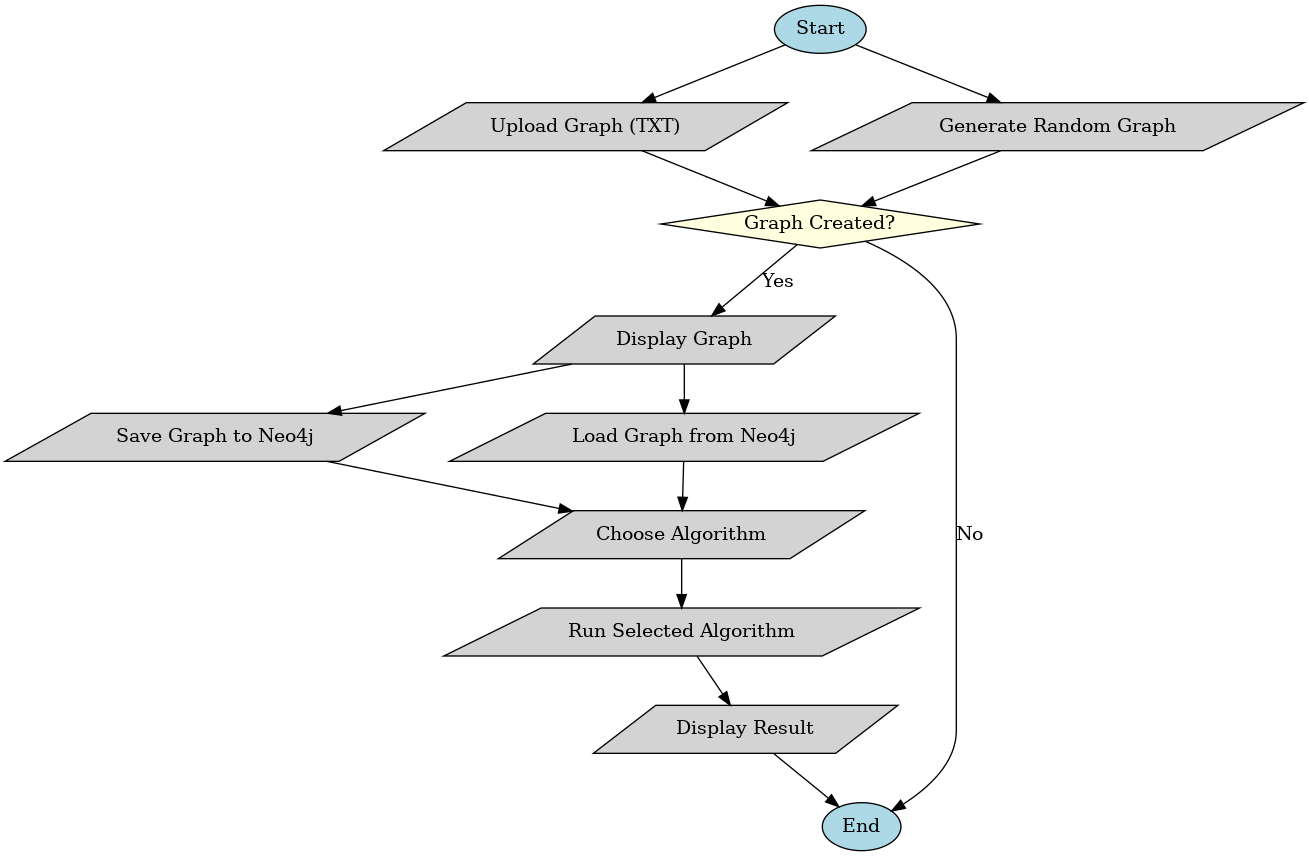
***1. Introduction***

This project focuses on developing a **Flow Management System** using graph-based algorithms to simulate flow distribution through a network of nodes. The system is designed with an interactive UI using Streamlit and stores graph data in Neo4j, enabling persistent storage and retrieval.

***2. Methodology***

1. **Graph Representation**
2. **Algorithm Implementation** - The project integrates flow algorithms such as Ford-Fulkerson, Push-Relabel, and Dinic’salgorithm to determine maximum flow between a source and a sink.
3. **Data Storage** - Neo4j is used to store graph structures persistently.
4. **User Interface** - Streamlit provides an interactive environment for users to generate, visualize, and store graphs.
5. **Testing** - Different graph sizes (small, medium, and large) are used to validate system efficiency.

***3. Flowchart***

******

***4. Implementation Details***

* **Graph Creation**: Users can create graphs manually or upload a .txt file containing edges and capacities.
* **Random Graph Generation**: Ensures no isolated nodes exist.
* **Graph Visualization**: Displays edges with capacities, with an option to open a full-screen view.
* **Saving & Loading Graphs**: Graphs are stored in **Neo4j**, each assigned a unique ID.
* **Algorithms Used**:
  + Ford-Fulkerson (Flow Augmenting Path)
  + Push-Relabel
  + Dinic’s Algorithm (Advanced Blocking Flow)

***5.Algorithms for all of three Step-by-step:***

Ford-Fulkerson:

- Initialize flow = 0 on all edges.

- While there exists a path from source to sink in the residual graph:

* Find a path using DFS or BFS.
* Compute bottleneck capacity = minimum residual capacity along the path.
* Augment flow along the path:
  + Add flow on forward edges.
  + Subtract flow on backward edges.

Repeat until no more augmenting paths exist.

Push-Relabel:

Initialization:

* Set all flow = 0
* Height of source = number of nodes (n)
* All other heights = 0
* Pre-flow: push as much flow as possible from source to its neighbors (based on capacities)

Maintain excess flow at each node.

* A node is active if it has excess flow > 0 (except source and sink)

While active nodes exist:

* Pick any active node u (≠ source/sink)
* Try to push to a neighbor v if:
  + residual capacity > 0
  + height[u] == height[v] + 1
* If push is not possible to any neighbor → Relabel:
  + height[u] = min(height[v]) + 1 for all v where residual capacity > 0

Dinic’s:

Initialize total\_flow = 0

While there exists a path from source to sink:

* Use BFS to build level graph (shortest distance from source)
* If sink not reachable, break

Use DFS to send blocking flows from source to sink along level graph

* Continue until no more flow can be sent

Add flow from DFS to total flow

***6. Testing (Small, Medium, Large Graphs)***

1. **Small Graph:** 5-10 nodes, 8-15 edges
   * Expected: Quick processing, accurate max flow calculation
2. **Medium Graph:** 15-30 nodes, 20-50 edges
   * Expected: Slight delay, results still efficient
3. **Large Graph:** 50+ nodes, 100+ edges
   * Expected: Increased computation time, ensures scalability

***6. Results & Conclusion***

The Flow Management System successfully simulates and evaluates flow distribution across varying network sizes. The integration of Neo4j enables persistent storage, and the Streamlit UI ensures user-friendly interaction. The project effectively demonstrates the application of network flow algorithms in practical scenarios, offering a valuable tool for flow analysis and management.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Method** | **Time Complexity** | **Best For?** |
| Ford-Fulkerson | BFS + Augmenting Path | O(E\*Max Flow) | Small Graphs |
| Dinic’s Algo | BFS + Level Graph | O(V²E) | Large Graphs |
| Push-Relabel | Preflow + Relabeling | O(V²E) | Dense Graphs |

***7.Practical Applications of Max Flow Algorithms****:*

1. **Network Traffic Optimization**
   * Managing data flow efficiently through computer networks to avoid congestion.
2. **Supply Chain and Logistics**
   * Maximizing the transportation of goods through supply routes while minimizing bottlenecks.
3. **Airline Scheduling**
   * Optimizing flight schedules and gate assignments at airports based on available resources.
4. **Sports Tournament Scheduling**
   * Deciding if a team can still qualify by modeling match outcomes as flow networks.
5. **Project Planning** 
   * Managing critical paths and maximizing resource allocation in project timelines.