Find the Most Influential Node in the Graph using Brandes Algorithm

1. **Objective:**

   My objective is to find the most **influential node** in the graph by calculating the **Betweenness Centrality** of each vertex using **Brandes' Algorithm**.

   - What is Centrality?
     Centrality indicates the most important vertices in a graph. Centrality plays a vital role in graph theory or network analysis. A vertex can reach other vertices quickly or the vertex lies on the paths connecting others if the centrality score is higher.

     There are different ways of calculating centrality:
     - Closeness Centrality
     - Graph Centrality
     - Stress Centrality
     - Betweenness centrality

$$C_C(v) = \frac{1}{\sum_{t \in V} d_G(v,t)} \qquad \textit{closeness centrality} \text{ (Sabidussi, 1966)}$$

$$C_G(v) = \frac{1}{\max_{t \in V} d_G(v,t)} \qquad \textit{graph centrality} \text{ (Hage and Harary, 1995)}$$

$$C_S(v) = \sum_{s \neq v \neq t \in V} \sigma_{st}(v) \qquad \textit{stress centrality} \text{ (Shimbel, 1953)}$$

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \qquad \textit{betweenness centrality} \text{ (Freeman, 1977; Anthonisse, 1971)}$$

2. **Applications:**
    - Find the most influential person in the Business Organization.
    - Find the key infrastructure node in Urban Network.
    - Find the cell which is responsible for the spread of virus to other cells.

3. **Variations**
    - As of now, I'm trying to calculate the Betweenness centrality to find the most influential node in the graph. If the time permits I'll try to incorporate the other centralities namely
        - Closeness Centrality
        - Graph Centrality
        - Stress Centrality
    - As of now, I'm considering undirected graphs. If possible I'll try for directed graphs as well.
    - I'll try to add weights to the edges. I'll try to find the shortest paths based on weights.

4. **Progress**

    As of now, I've tried to read the csv file. I'm still figuring out which algorithm to use; to find the shortest paths as well as find the vertices which lie on the shortest paths. I've understood and trying to implement the Brandes' algorithm which can find the distance from each vertex as well as find the predecessors of the destination vertex.