1 Question 1

(1) To adapt the DeepWalk architecture to a directed graph, we should change the way we visit each node's neighbors in $random\_walk$ function. Neighbors will therefore be restricted to all visitable nodes, taking into account the direction of the edges.
(2) If the graph is weighted, we can choose each node's neighbors according to a certain probability, which will be proportional to the edge weigth.

2 Question 2

Consider the following reflection matrix :

$$S = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

This gives :

$$X_1 S = X_2$$

This reflects a flip across a line of symetry (the x axis).

3 Question 3

As we can see in this equation :

$$Z^0 = f(\hat{A}XW^0)$$

The new feature vector of each node is the sum of the feature vectors of its direct neighbors. We have thus the next equation :

$$Z^1 = f(\hat{A}Z^0W^1)$$

And as we can see, the new feature vector of each node is related to the neighbors of the neighbors of the node. Finally, we calculate Y as :

$$\hat{Y} = softmax(Z^1W^2)$$

To conclude, in this case, the maximal distance reached is the neighbors of the node's direct neighbors.

By generalizing, we can see that each message passing layer reaches the neighbors of the nodes already considered. So the maximal number of edges separating a given node i from the nodes the features of which are taken into account in the prediction $\hat{Y}i$ is k.

4 Question 4

4.1 Complete graph

First, let's consider the complete graph K_4, we have for adjacency matrix :

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

The normalized form is :

$$\hat{A} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

We apply f to the product $\hat{A}XW^0$:

$$Z^0 = f\left(\begin{bmatrix} -0.8 & 0.5 \\ -0.8 & 0.5 \\ -0.8 & 0.5 \\ -0.8 & 0.5 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \\ 0 & 0.5 \end{bmatrix}$$

Thus, we can compute Z^1

$$Z^1 = f\left(\begin{bmatrix} -0.28 & 0.06 & 0.29 \\ -0.28 & 0.06 & 0.29 \\ -0.28 & 0.06 & 0.29 \\ -0.28 & 0.06 & 0.29 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0.06 & 0.29 \\ 0 & 0.06 & 0.29 \\ 0 & 0.06 & 0.29 \\ 0 & 0.06 & 0.29 \end{bmatrix}$$

The feature vector of each node is the same. The reason for this is that each node has the same number of neighbors, so the adjacency matrix added to the identity matrix ($\bar{A}$) will be equal to a matrix full of ones, and the features vectors computed will be equal.

4.2 Star graph

Now, we consider the complete graph S_4, we have for adjacency matrix :

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The normalized form is :

$$\hat{A} = \begin{bmatrix} 0.25 & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & 0.5 & 0 & 0 \\ \frac{1}{\sqrt{8}} & 0 & 0.5 & 0 \\ \frac{1}{\sqrt{8}} & 0 & 0 & 0.5 \end{bmatrix}$$

We apply f to the product $\hat{A}XW^0$:

$$Z^0 = f\left(\begin{bmatrix} -1.04 & 0.65 \\ -0.68 & 0.42 \\ -0.68 & 0.42 \\ -0.68 & 0.42 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0.65 \\ 0 & 0.42 \\ 0 & 0.42 \\ 0 & 0.42 \end{bmatrix}$$

Thus, we can compute Z^1

$$Z^1 = f\left(\begin{bmatrix} -0.34 & 0.07 & 0.35 \\ -0.24 & 0.05 & 0.25 \\ -0.24 & 0.05 & 0.25 \\ -0.24 & 0.05 & 0.25 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0.07 & 0.35 \\ 0 & 0.05 & 0.25 \\ 0 & 0.05 & 0.25 \\ 0 & 0.05 & 0.25 \end{bmatrix}$$

Similarly to the previous case, the 3 nodes whose neighbors are only the central vertex will have the same feature vector (as we can see in the 2, 3, 4 rows of Z^1). As expected, the feature vector for the first node is different from the others.
If the node features X were randomly sampled from a random uniform distribution, we will see that $\hat{Y}$ is impacted randomly. But this will be countervailed by the weights training.

5 Figures

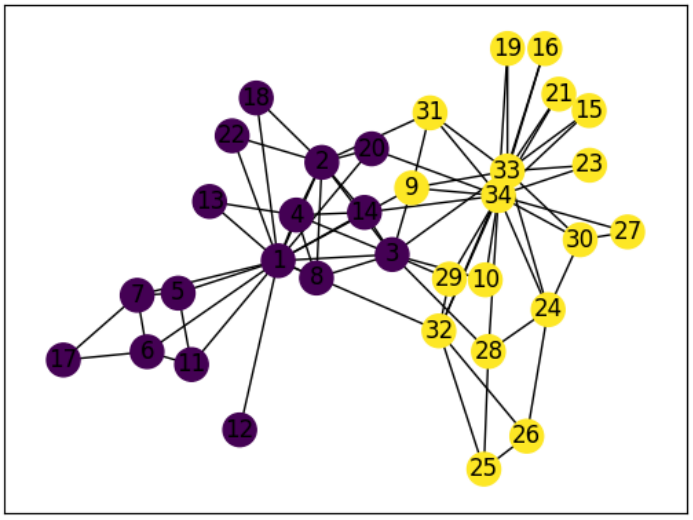

Figure 1: visualize the karate network

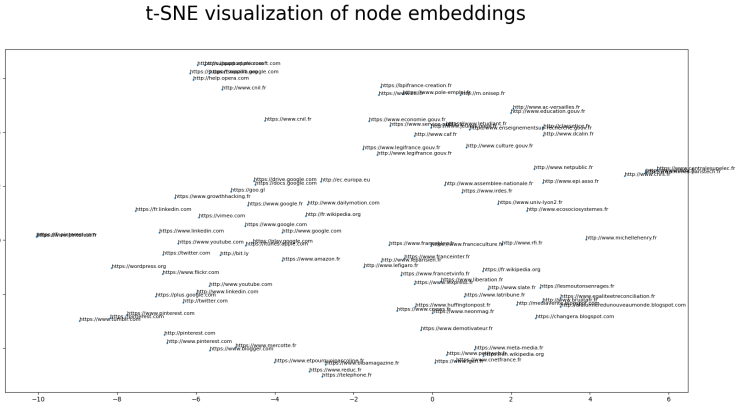

Figure 2: t-sne visualization of node embeddings

Figure 3: t-sne visualization of second message layer output