Simon Queric
simon.queric@telecom-paris.fr

Part I

Question 1

Intuitively, I expect embeddings of two nodes belonging to the same component to be colinear so that the cosine similarity will be high. Embeddings of two nodes belonging to different components should be orthogonal and thus their cosine similarity should be equal to zero.

Question 2

According to [2], DeepWalk algorithm time complexity is bounded by $O(|V|)$.
The spectral embedding technique requires the diagonalization of the Laplacian matrix which is of dimension $|V|$ so that the time complexity of this technique is in $O(|V|^3)$ according to [1].

Part II

Question 3

If no self-loops were added, $D^{-1/2}AD^{-1/2}$ would be like a transition of a random walk that wouldn't allow to stay at the same place. Concretely, for a vector Y, the paper [3] explained that $(\widetilde{A}Y)_i = \sum_j \widetilde{A}_{ij}Y_j = Y_i + \sum_{j\in\Gamma(i)} Y_j$ and $(AY)_i = \sum_{j\in\Gamma(i)} Y_j$. Without self-loops, the embedding of node i contains information about the neighbors but not about the node i itself.

Question 4

In the case of the star graph S_4, we see that the 3 nodes connected to the center node have the same embedding, different from the one of the center node.
In the case of the cycle graph C_4, all the nodes have the same embedding.
This embedding respects the symmetry of the graph.

```python
W0 = np.array([0.5, -0.2])
W1 = np.array([[0.3, -0.4, 0.8, 0.5],
               [-1.1, 0.6, -0.1, 0.7]])
def forward(A, X) :
    X = np.ones(4)
    Z0 = np.maximum(np.einsum("i, j -> ij", A@X, W0), 0)
    Z1 = np.maximum(A@Z0@W1, 0)
    return Z1
# Star Graph S_4
A = np.array([[1, 1, 1, 1],
              [1, 1, 0, 0],
              [1, 0, 1, 0],
              [1, 0, 0, 1]])
D = np.sqrt(np.diag([4, 2, 2, 2]))
A = D@A@D
X = np.ones(4)
Z1 = forward(A, X)
print("Embedding for a star graph :")
print(Z1)
print("\n")
# Cycle graph C_4
A = np.array([[1, 1, 0, 1],
              [1, 1, 1, 0],
              [0, 1, 1, 1],
              [1, 0, 1, 1]])
D = np.sqrt(np.diag([3, 3, 3, 3]))
A = D@A@D
Z1 = forward(A, X)
print("Embedding for a cycle graph :")
print(Z1)
```

✓ 0.1s

```
Embedding for a star graph :
[[13.63675324  0.         36.3646753  22.72792206]
 [ 6.74558441  0.         17.9882251  11.24264069]
 [ 6.74558441  0.         17.9882251  11.24264069]
 [ 6.74558441  0.         17.9882251  11.24264069]]


Embedding for a cycle graph :
[[12.15  0.   32.4  20.25]
 [12.15  0.   32.4  20.25]
 [12.15  0.   32.4  20.25]
 [12.15  0.   32.4  20.25]]
```

Figure 1:

References

[1] James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, October 2007.

[2] Tiago Pimentel, Rafael Castro, Adriano Veloso, and Nivio Ziviani. Efficient estimation of node representations in large graphs using linear contexts, 2019.

[3] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification, 2018.