1. **Node Embeddings**
   a. **For node embedding using LINE with first-order proximity, what's the time complexity of calculating all pairs of node similarities?**
   Answer**:** O(|E|)
   How about second-order proximity?
   Answer: O(|E| * d), where d is the average degree of a node in G
   Suppose we now use a negative sampling scheme for K << |V| negative samples, what's the time complexity in this case?
   Answer: O(|E| * K)
   b. **Reason about why second-order proximity is required on top of first-order proximity. You may find the discussion in the original paper fruitful.**
   Answer: With a first-order proximity, the amount of information we can gain from embeddings is very sparse; second-order proximities allow us to leverage a wider range of nodes to handle indirect relationships and capture local/global structure. As a result, we may have more informative embeddings for downstream tasks.
   c. **Read the note from CS224W about Random walk-based approach and reason about comparison with LINE; in particular, the advantages and disadvantages potentially associated with this approach.**
   Answer: Random walk-based approaches are more flexible in capturing local and global views of a graph; conversely, LINE only offers fixed first- or second-order proximities for each node. Furthermore, since we do not consider all node pairs when training (and only on pairs that co-occur on random walks), random walks are more computationally efficient. However, random walk approaches do not retain information about the structure of the graph, and they require parameter tuning in order to work properly.
   d. **Reason (shortly) about the relation between Node2Vec and Word2Ve, from the perspective of sequence vs. graph data.**
   Answer: Where Word2Vec is used for processing sequential data, Node2Vec is tailored for graph-structured data. Both learn embeddings based on "neighbors", be that the surrounding context of words or co-occurring nodes, and use them to make predictions about other words/structural relationships.