1 Question 1

We can modify the random walk as follows:

- Start from a starting vertex v_1
- Initialize $i = 1$ and $v_0 = \emptyset$
- While ($length\_of\_walk < L$) and ($neighbor(v_i) \setminus \{v_{i-1}\}$ is not empty):
  - Choose v_{i+1} randomly from $neighbor(v_i) \setminus \{v_{i-1}\}$
  - Add v_{i+1} to the walk
  - Set $i = i + 1$

where L is the maximum length of the random walk. By this way we can be sure that v_{i+1} will be chosen from a subset not containing v_{i-1}, hence we will always have $v_{i+2} \neq v_i$ for all i.

2 Question 2

If we want to do a graph classification, we can use a graph kernel matrix $\mathbf{K}$ instead of the adjacency matrix $\mathbf{A}$. Suppose now that we have n graphs to classify, then we can compute the output of the network similar to the node classification as follows:

$$\widehat{\mathbf{Y}} = softmax\left(f\left(\widehat{\mathbf{K}}f\left(\widehat{\mathbf{K}}\mathbf{X}\mathbf{W}^0\right)\mathbf{W}^1\right)\mathbf{W}^2\right),$$

where $\widehat{\mathbf{K}}$ is the normalized matrix of graph $\mathbf{K}$, $\mathbf{X}$ is a matrix whose i^{th} row contains the feature vector of graph i, the matrices $\mathbf{W}^0, \mathbf{W}^1, \mathbf{W}^2$ are the trainable weight matrices and f is the activation function.

3 Question 3

We observe that both the two methods can attain the accuracy of 1.00 for the default train test ratio (0.8). However, while the GNN always gives the absolute correct accuracy just after a very short time, the DeepWalk sometimes drops below to 0.85.
If we decrease the train test split ratio to 0.3, we obtain

```
Accuracy DeepWalk:  0.4166666666666667
Accuracy GNN: 0.8333
```

We deduce that GNN outperforms the DeepWalk approach. This is normal because GNN is a supervised method, while DeepWalk is an unsupervised method to learn graph representations, hence it's normal that GNN works better than DeepWalk for this node classification task.

4 Question 4

If we set the nodes to have the same feature vector, the accuracy of GNN drops from 1.00 to 0.2857. This is due to the fact that the dimentionality of the input reduced from n to 1.
More precisely, when we set $\mathbf{X} = \mathbf{I}_n$, then $\hat{\mathbf{A}}\mathbf{X} = \hat{\mathbf{A}}$, thus each input vector of the GNN is one row of $\hat{\mathbf{A}}$, which contains the information of all the neighborhoods of each node.
However, if we reduce $\mathbf{X} = (1, 1, \ldots, 1)^\top$, then the i^{th} row of $\hat{\mathbf{A}}\mathbf{X}$ will only contains the sum of i^{th} row of $\hat{\mathbf{A}}$, which is only equivalent to the degree of each node.