Assignment 1

Sudarshan S Harithas (2021701008)

1 Question 1

1.1 Question 1.1

Graph Name	Nodes	Edges	degree (mean , Std , min , max)	density	sparsity
Graph 1	27917	206259	(14.77658 , 22.97, 1, 678)	0.0005293	0.9994707
Graph 2	2642	3303	(2.5001891 , 0.7485 , 1, 5)	0.00094	0.99906

1.1.1 Analysis

- **Absolute Analysis**

  The density of an undirected graph is given by equation 1, where V is the number of vertices and $|E|$ is the number of edges and D is the density of the graph.

  $$D = \frac{2|E|}{V(V-1)} \tag{1}$$

  - **Graph 1**
    - We can observe that Graph 1 has a low density value of 0.00053. It is a large graph with more than 2 lakh edges. For graphs of this size an adjacency matrix (in traditional form) is an ineffective method as it increases the memory footprint. Sparse matrix representations [1] are more effective storage structures for such matrices.
    - The mean degree of the graph is close to 15, and the standard deviation is close to 23 and a maximum value is 678, therefore we can expect a tail in the *degree distribution graph*.
  - **Graph 2**
    - Although the size of Graph 2 is smaller in comparison to Graph 1, the adjacency matrix (traditional form) can still be an ineffective memory structure. Since the number of edges is in the order of thousands an adjacency list can be used or sparse matrix representations can be used for increased efficiency.
    - We observe that the standard deviation of the graph is low, and the maximum degree value is comparable to the mean hence we would not observe a tail in the degree distribution.

- **Comparative Analysis**
  - Since Graph 1 has a larger number of nodes and edges than Graph 2 it is larger.
  - A larger standard deviation in the degree score of *Graph 1* is an indicator of a rich distribution (i.e it spreads across a wider range of degrees) and the presence of a maximum value of 678 indicates that a particular node is dominating in the graph. On the other hand, we observe the standard deviation to be small value in graph 2 and the maximum value is comparable to the mean, this implies that there is a possibility that all nodes are uniformly connected to each other i.e the degree of most of the nodes are similar (or comparable).

[1] $https://en.wikipedia.org/wiki/Sparse_matrix$

1.2 Question 1.2

Cliques: A clique for node **v** is a largest complete subgraph containing **v**. A clique with K-vertices is known as a K-clique, the number of 3-cliques and 4-cliques for Graph 1 and 2 is given in the table below. A maximal clique is clique which cannot be expanded by adding another adjecent node.

Graph	#3 Clique	# 4 Clique
Graph1	387444	818754
Graph2	53	0

Graph	#3 maximal Clique	# 4 maximal Clique
Graph1	38002	20547
Graph2	53	0

Centrality Measure Here, we analyse the graph using 4 centrality measures namely the *Degree Centrality*, *EigenVector Centrality*, *Closeness Centrality* and *Betweenness Centrality*. The definitions of these measures are as follows.

- **Degree Centrality** The degree centrality of a node is the number of edges it has. **Note:** The values reported in the table below for degree centrality are the normalized values as explained here
- **Eigenvector Centrality** eigenvector centrality measures the influence of a node in the network.
- **Closeness Centrality** of a node is calculated as the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph.
- **Betweenness Centrality** of a given vertex is the number of shortest paths between any other two vertices of the graph, that pass through the vertex.

Graph		Degree	eigenvector	Closeness	Betweeness
Graph 1	Mean	0.000529	0.0023	0.2325	0.00012138
	Std	0.0008	0.0055	0.0317	0.00076
	Max	0.0242	0.1670	0.348	0.0433
Graph2	Mean	0.0009	0.00375	0.028	0.0129765
	Std	0.00028	0.0190	0.00441	0.0220
	Max	0.0018	0.2403	0.0380	0.19928

Clustering Coefficients Three measures for the clustering coefficients are used namely *No. of Triangles*, *Transitivity* and *Clustering Coefficient*. They are defined as follows

- **No. of Triangles** the number of triangles in a graph can be used as one of the parameters to quantify clustering. **Note:** during computation the networkx api counts each triangle three times once at each node.
- **Transitivity** is a measure proportional to the ratio of the number of triangles in the graph to the total number of possible triples of nodes (a node triple need not have an edge in between them).
- **Average Clustering:** In an unweighted graph the clustering coefficient is s a measure of the degree to which nodes in a graph tend to cluster together, it measures how connected the neighbours of a node **v** are. The average clustering coefficient is the average value of all the coefficients in the graph.

Graph	No. of Triangles	transitivity	average clustering
Graph 1	1162332	0.11398	0.2953
Graph 2	159	0.0279	0.015954

Analysis

- **Graph 1**: From the number of cliques we can gain in intuition that the graph would have a large set of closely collected members and this also explains the relatively high value the Jaccard Coefficient (explained in section 1.3) .

  Regarding the centrality measures, we notice that the standard deviation of the degree centrality is low this would imply a high concentration of measurements close to the mean, furthermore the maximum value of the degree centrality is far-away from the mean this could be caused the node with the with 678 degrees (as seen earlier). The eigenvector centrality has a higher standard deviation and we can expect a larger distribution around the mean. The standard deviation of the closeness centrality is low, which indicates that nodes are closely connected and easily accessible.

  In the clustering coefficients we observe a large number of triangles that indicate the nodes are well connected at lower degrees, the presence of a considerable value of transitivity and a relatively high clustering coefficient further augments the observation.

- **Graph 2** is relatively a smaller graph with fewer nodes and edges in comparison to graph 1. The largest k-clique that can be formed in this graph is with 3 vertices (k=3).

  Using observations form the node centrality measures further provides us an intuition that most of the nodes have a similar degree measure (i.e. the graph has nodes with a close to uniform degree measure the distribution is not rich) .

  Since the largest clique is 3 and the clustering coefficients are low we can expect multiple nodes of degree 2 (this has been observed in Fig 1) and since there are limited paths between nodes we observe a relatively higher betweeness centrality.

1.3 Question 1.3

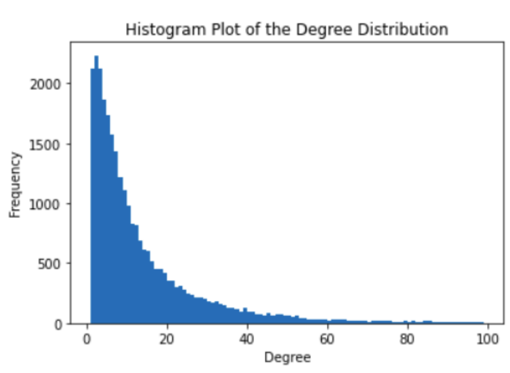
- **Neighbourhood Parameters**

Graph	Jaccard Coefficient	Adamic Adar Index
Graph1	0.11600	Undefined as network has loops
Graph2	0.0083	0.040

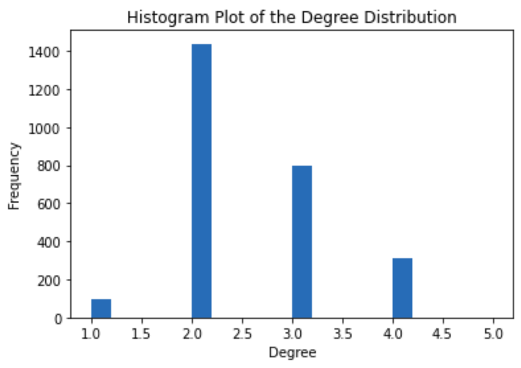
  - The Jaccard Coefficient is a similarity measure the finds the intersection over union of the input set. Therefore, a higher value of the Jaccard Coefficient between a pair of nodes indicates that the node has a higher number of common neighbours. In the above table we represent the average Jaccard Coefficient i.e. we sample all the edges (or pair of nodes) from the graph and determine the Jaccard Coefficient individually finally we take the mean of all the obtained values. We can observe that Graph 1 has a higher Jaccard Coefficient than Graph 2, this implies that on an average a pair of nodes of graph 1 have more common-known neighbours than a pair of nodes in graph 2.
  - The *Adamic Adar Index* is defined as the sum of the inverse logarithmic degree centrality of the neighbours shared by the two nodes. The Networkx Library defines this index to only graphs with no loops.

- **Histogram of Degrees**

  The degree distribution is shown in Fig. 1, it can be observed that the distribution of Graph 1 shown in Fig 1a, follows an exponential distribution and is richer (i.e it spreads across a wider range of degrees) in comparison to the histogram of Graph 2 Fig. 1b. It can be observed that graph 1 has a higher variance in the distribution and the variance in the distribution of graph 2 is quite low.

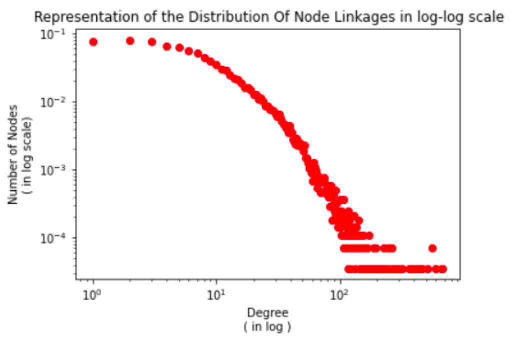


(a) Graph 1

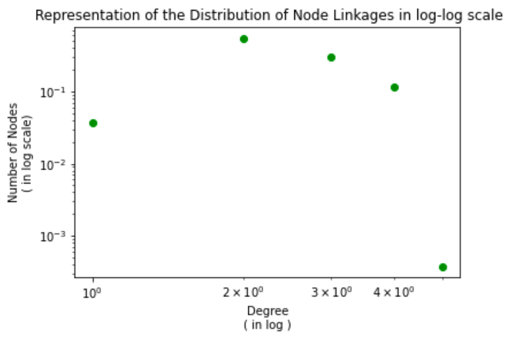


(b) Graph 2

Figure 1: Histogram Plot of Degree Distribution



(a) Graph 1 (Normalized)



(b) Graph 2(Normalized)

Figure 2: Log-Log Plot of the Degree Distribution

- **Log Log Distribution**
  An alternative representation of the distribution is the *Log-Log Plot* here both x and y axis is in logarithmic scale. We can learn from the graph that the majority of the population in Graph 1 Fig. 2a have very low degree (the same can be further confirmed from Fig. 1) and the fraction of the population with high degree is very low. Furthermore from the distribution of graph 2 Fig. 2b we observe that the population (of degrees) spread more uniform in comparison to graph 1 .

2 Question 2

2.1 Question 2.1

2.1.1 EigenVector Centrality

- **Example Application**: To determine influential people in a society.
- **Inputs**: Every individual corresponds to a node of the graph, and the edges represent connection between the members. **Output** Determine the node(s) that are more influential in the society.
- **Definition of Influence** : The influential ability of a member is proportional to the number of influential people he is connected to.
- **Intuition** The formulation must take into account of the importance of the node and the importance of its neighbouring nodes.
- **Reasoning**: Eigen Vector centrality is given by Eq. 2, here the score for node v is proportional to the scores of its neighbours. The formulation is inline with the intuition and hence it is an

appropriate choice for the application.

$$c_v = \frac{1}{\lambda} \sum_{u \in N(v)} c_u \tag{2}$$

- **Why not other metrics ?**
  - **Degree Centrality** It measures the number of people that the node is connected to but does not consider the "importance" of each of its connections.
  - **Betweenness Centrality** It measures the ability of the node to act as a bridge for the passing of information, it does not indicate the ability of a node to influence.
  - **Closeness Centrality**: This metric indicates the ease of accesablity to other nodes but does not consider the importance of a node in formulation hence it is not suitable.
- **Classifier**: A variant of the problem would be to classify a person as "influential" or "non-influential" based on the eigenvector centrality scores. A semi-supervised learning mechanism can be formulated and binary classifiers such as SVM (or kernel SVM) can be used. The model would essentially take the input feature vector (the eigenvector centrality score) and predict if a person is influential or not.

2.1.2 Closeness Centrality

- **Example Application**: Consider an engineering problem in the data communication network. A node can receive a message from an external source and it is expected to broadcast the same message throughout the network. Here a node is to be identified (to receive that initial message) such that it broadcasts the message with minimum effort. i.e. identify a node such that a message emanating from it would reach its destination with minimum effort.
- **Inputs** A network of computers modelled as a graph where an individual computer is a node and the connection between computers is an edge. (Note that the network is not fully connected and is sparse but there is at-least one way for a node to reach any other node within the network). **Output** The identified node to which the message is initially passed.
- **Definition of Effort** The effort is proportional to the length (or time) the message has to travel before it reaches its destination.
- **Reasoning** The closeness centrality by definition is the reciprocal of the sum of the length of the shortest paths between the node and all other nodes in the graph. Thus, the more central a node is, the closer it is to all other nodes. It is given by Eq. 3. This makes the closeness centrality the ideal metric for this application.

$$C(x) = \frac{1}{\sum_y d(x,y)} \tag{3}$$

- **Why not other metrics ?**
  - **EigenVector Centrality** It captures the importance of a node with respect to its neighbours and is not a distance measurement system. By The definition of effort we require a distance measurement system hence eigen vector centrality is not suitable.
  - **Betweeness Centrality** Although this measure considers the distance measurement, it mainly signifies the ability of a node to act as a bridge for message and does not represent the ease of accessibility of a node, hence it is not an appropriate measure.
- **Classifier** A variant of the problem is to classify if a given node is "fit" or "unfit" to receive the external message, a node is classified as unfit if it lies towards the end points of the graph (far away from other nodes) and the closer the nodes are towards the center they would be regarded as "fit". A semi-supervised binary classification mechanism using the closeness centrality as the feature can be formulated to perform classification.

2.1.3 Betweenness Centrality

- **Example Application** Identification of critical nodes in railway network.
- **Inputs** A railway network modelled as a graph, where every railway station is a node and the path between connecting railway stations is an edge. **Output** identify (or rank) railway stations (nodes) that are critical to the smooth functioning of the network.
- **Definition of Criticality of a node** The importance of a node is proportional to the traffic that flows through it.
- **Reasoning** The betweenness centrality measures the ability of a node to act as bridge between other nodes. Intutively, more frequently a given node is present in the way of shortest path between the other nodes the higher its ability to act as a bridge. This property makes the betweeness centrality an appropriate measure.

$$C(x) = \sum_{s \neq v \neq t} \frac{\#(shortest\ path\ between\ s\ and\ t\ that\ contain\ v)}{\#(shortest\ path\ between\ s\ and\ t)} \tag{4}$$

- **Why not the other metrics ?**
  - **EigenVector Centrality** It captures the importance of a node with respect to its neighbours and is not a measurement of the ability of a given node to act as a bridge between stations.
  - **Closeness Centrality** Despite being a distance measure, it caputers how accessible a node is from all the other nodes and does not account for its message passing (or traffic passing) ability hence its not an appropriate measure.
- **Classifier** A variant of the problem would be to classify nodes as "critical" or "non-critical". A semi-supervised binary classification problem using the betweenness centrality as the feature can be formulated to perform the task.

2.2 Question 2.2 WL Kernels for Isomorphism Test

2.2.1 Algorithm

The idea of color refinement can be extended to determine if two graphs are isomorphic, [1] suggests an algorithm for the same. In the context of this question the algorithm can be used as follows

Algorithm 1 explains the WL test for graph isomorphism, the two graphs on which the test is to be performed is the input to the program. Isomorphic graphs are expected to have equal number of nodes, the function *AreNodesEqual* checks if both the graphs have equal number of nodes and returns *True* on a positive result. (In the context of this question this function returns a *True* as both graphs have equal 6 nodes each).

The variables V_a^i and L_a^i are used to store the labels before and after the hashing operation respectively, the subscript **a** corresponds the graph number (1 or 2) and the superscript **i** denotes the i^{th} iteration. The *InitilizeLabels* function sets all the nodes to the label 1, i.e. all nodes have the same color.

The *AggregateNewLabels* function performs the aggregation step, here the node labels are augmented by the sorted set of node labels from the neighbouring nodes.

The *GenerateHashLabels* funciton is another key componenet of the algorithm, it compress the augumented labels into new, short label.

The ***TerminateConditions*** is a function that determines if the graphs are isomorphic. The WL test algorithm terminates if the newly created labels of *Graph 1 and Graph 2* are not identical to each other, the element wise comparison is performed using the *AreSame* function. Incase of an early termination it is guaranteed [1] that the graphs are non-isomorphic, if the sets are identical after N_{max} iterations it implies that the graphs are either isomorphic or the algorithm was not able to determine that they are non-isomorphic.

Algorithm 1: WL Test for Graph Isomorphism

```
1  Input: Two graphs $G_1$ and $G_2$ on which the WL test is to be performed
2  $N_{max}$ = maximum number of iterations
3  bool IsIsomorphic = True
4  if !AreNodesEqual( $G_1, G_2$)) then
5      IsIsomorphic = False
6      return IsIsomorphic
7  end
8  $V_1^0$ , $V_2^0$ ← InitilizeLabels( $G_1, G_2$ )
9  $L_1^0$ , $L_2^0$ = $V_1^0$ , $V_2^0$
10 for int i =0 ; i ¡ $N_{max}$ ; i++ do
11     $V_1^i$ , $V_2^i$ ← AggregateNewLabels( $L_1^i$ , $L_2^i$ )
12     $L_1^i$ , $L_2^i$ ← GenerateHashLabels( $V_1^i$ , $V_2^i$ )
13     if !AreSame( $L_1^i$ , $L_2^i$ ) then
14         IsIsomorphic = False
15         return IsIsomorphic
16     end
17 end
18 TerminateConditions(IsIsomorphic)
```

2.2.2 Example 1

The working of the WL algorithm to determine Graph isomorphisim is shown in Fig 3. The two input graphs $G_1 and G_2$ shown in Fig. 3a, the initialization step is performed where all nodes are assigned the same label 3b, the aggregation step performs the operation shown in Fig 3c where the labels of neighbouring are aggregated to form a new label. The result of this step is the input to the hashing function given by Eq. 5 and the output color is used for the next iteration. The *HASH* function in this step performs the following mapping operation

$$(1, 11)- > 2$$

. The iteration is repeated and the aggregation and Hashing steps are performed, in the second iteration the *HASH* function performs mapping between

$$(2, 22)- > 3$$

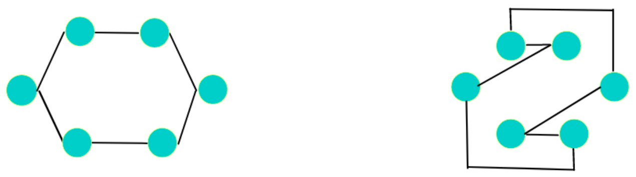
.

$$c^{k+1}(v) = HASH\{c^k(v), \{c^k(u)_{u \in N(v)}\}\} \tag{5}$$

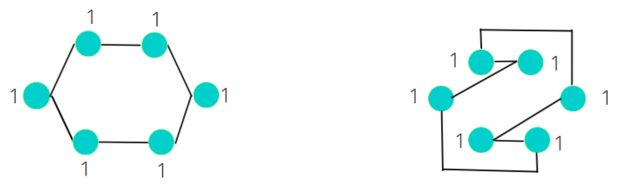
At the end of two iterations the WL kernel counts the number of nodes with the given color the results are given in Eq. 6. The vector [6, 6, 6] is the result of the observation that there are 6 units of color 1 , 6 of color 2 and 6 of color 3.

$$Colorcode[1, 2, 3] \tag{6a}$$
$$\phi(G_1) = [6, 6, 6] \tag{6b}$$
$$\phi(G_2) = [6, 6, 6] \tag{6c}$$

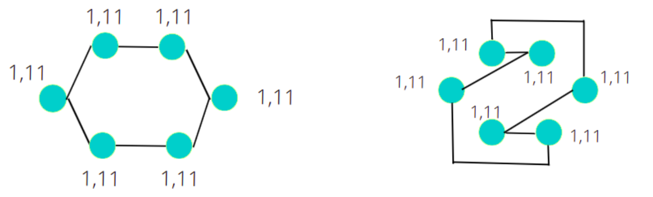
It can be observed that this trend of aggregation and hashing would continue until N_{max} iterations and the labels would be returned the equal to each other after every iteration i.e $\phi(G_1) = \phi(G_2)$. Under such circumstances the WL test can be terminated and the algorithm would conclude that the graph can be either isomorphic or the test did could not conclusively determine that the graph is non-isomporphic (In case of this illustration we can conclude that the graph is isomorphic).



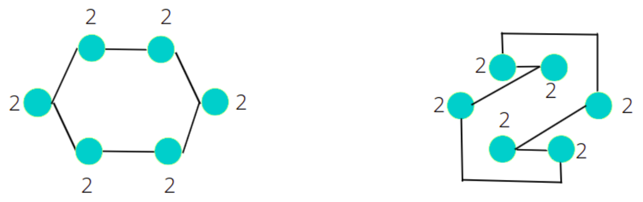
(a) The two graphs G_1 (left) and G_2 (right) on which the WL test is to be performed



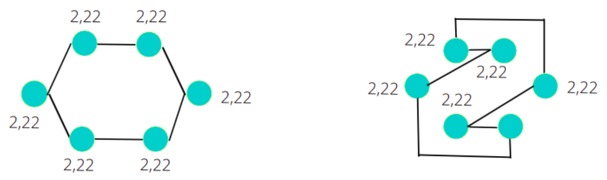
(b) Initilize the Graphs with all nodes belonging to the same color



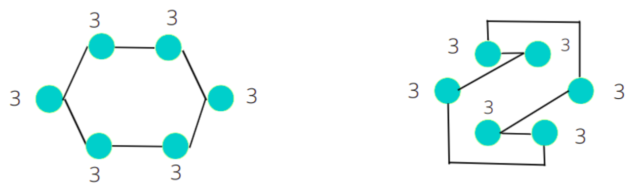
(c) Perform the aggregated label generation step



(d) Hash the aggregated labels to a short label (1,1) -> 2

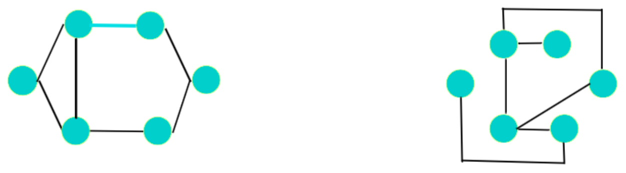


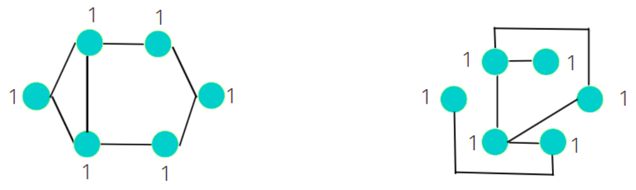
(e) Second Iteration of the label aggregation



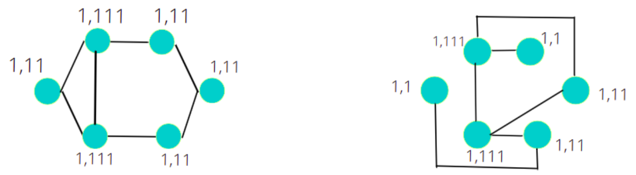
(f) Second Iteration of the hash label generation step (2,2) -> 3

Figure 3: Example 1: Illustration of the WL test for isomorphisim for a pair of isomorphic graphs

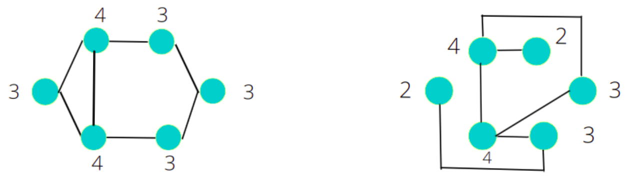

(a) The two graphs G_1 (left) and G_2 (right) on which the WL test is to be performed



(b) Initilize the Graphs with all nodes belonging to the same color



(c) Perform the aggregated label generation step



(d) Hash the aggregated labels to a short label

Figure 4: Example 2: Illustration of the WL test for isomorphisim for a pair of non-isomorphic graphs

2.2.3 Example 2

Fig. 4 is an illustration where the WL test is performed for a pair of non-isomorphic graphs. As explained in Example 1 (2.2.2) we perform the label aggregation and the hashing step as given by Eq 5 Fig 4c and Fig. 4d respectively. The hash mapping function is given below

$$(1,1)->2$$
$$(1,11)->3$$
$$(1,111)->4$$

As defined in 2.2.2, at the end of the iteration the WL kernel counts the number of nodes with the given color the results are given in Eq. 7.

$$Colorcode[1,2,3,4] \tag{7a}$$
$$\phi(G_1) = [6,0,4,2] \tag{7b}$$
$$\phi(G_2) = [6,2,2,2] \tag{7c}$$

It can be observed that the graph kernel values are not equal (element-wise) hence it can be concluded with guarantee that the two graphs are non-isomorphic [1].

References

[1] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.