# Machine Learning on Graphs: Foundations and Applications

## EXERCISE 2

**Group 29**

Rishang Yadav 485041 | Risa Niranjan Chaudhari 485510 | Bhoomi Dhawan 485556

---

1. **Problem 1:** For $k > 1$, prove that the folklore $k$-WL is at least as expressive as the $\delta$-$k$-WL in distinguishing non-isomorphic graphs.

**Ans:** We prove that whenever two $k$-tuples $s, t$ receive the same stable colour under the folklore $k$-WL, then they also receive the same stable colour under the $\delta$-$k$-WL. The proof proceeds by induction using unrolling trees.

**Unrolling trees.** For a WL-type algorithm $\mathcal{A}$ and a tuple $v \in V(G)^k$, let $\mathrm{UNR}^{\mathcal{A}}[G, v, i]$ be the depth-$i$ unrolling tree. Standard WL theory gives:

$$C^k_{i,\mathcal{A}}(v) = C^k_{i,\mathcal{A}}(w) \iff \mathrm{UNR}^{\mathcal{A}}[G, v, i] \cong \mathrm{UNR}^{\mathcal{A}}[G, w, i].$$

Thus two tuples have the same stable WL-colour iff all their unrolling trees of all depths are isomorphic.

Suppose

$$\mathrm{UNR}^{\text{folklore}}[G, s, q] \cong \mathrm{UNR}^{\text{folklore}}[G, t, q]$$

for some $q$ exceeding the folklore WL stabilization time. We prove by induction on $i$ that

$$\mathrm{UNR}^{\delta}[G, s, i] \cong \mathrm{UNR}^{\delta}[G, t, i].$$

**Base case $i = 0$.** Zero depth unrollings consist only of the atomic type of the tuple. Since the folklore unrolling roots are isomorphic, $s$ and $t$ have the same atomic type, hence

$$\mathrm{UNR}^{\delta}[G, s, 0] \cong \mathrm{UNR}^{\delta}[G, t, 0].$$

**Inductive step.** Assuming the claim holds for depth $i - 1$, we now show it for depth $i$.

For each position $j \in \{1, \dots, k\}$ define

$$X_j = \{\varphi_j(s, w) : w \in V(G)\}, \qquad Y_j = \{\varphi_j(t, w) : w \in V(G)\},$$

i.e. all tuples obtained by replacing the $j$-th component. In the $\delta$-unrolling, the children of $s$ at position $j$ are exactly the depth-$(i-1)$ unrollings of elements of $X_j$, with edge label $(j, \mathrm{adj}(s_j, w))$, and similarly for $Y_j$.

**Folklore vectors determine adjacency.** In folklore $k$-WL, the neighbour of $s$ obtained by replacing $v_j$ with $w$ contributes the vector

$$\big(C(\varphi_1(s, w)), \dots, C(\varphi_k(s, w))\big).$$

The colour $C(\varphi_j(s, w))$ contains the atomic type of $\varphi_j(s, w)$, which encodes whether $w$ is adjacent to $s_j$. Hence the adjacency bit required by $\delta$-WL is already determined by the folklore colour vector.

**Matching children.** Since the folklore unrolling trees of $s$ and $t$ are isomorphic at depth $q$, the multisets of folklore colour-vectors obtained from $s$ and $t$ coincide. Consequently, for each position $j$ and each folklore colour-vector class $\mathbf{c}$, the number of elements of $X_j$ of class $\mathbf{c}$ equals that of $Y_j$. Because $\mathbf{c}$ determines the adjacency bit, the numbers of local and global $\delta$-children in each class match.

Thus for each $j$, we can have a bijection

$$\sigma_j : X_j \to Y_j$$

preserving adjacency bits and folklore colours.

**Applying the induction hypothesis.** If $x \in X_j$ and $\sigma_j(x) \in Y_j$ have the same folklore colour at depth $i-1$, then their folklore unrollings of depth $i-1$ are isomorphic. By the induction hypothesis,

$$\mathrm{UNR}^\delta[G, x, i-1] \cong \mathrm{UNR}^\delta[G, \sigma_j(x), i-1].$$

**Conclusion of the step.** The -children of $s$ and $t$ at position $j$ are exactly the trees rooted at $X_j$ and $Y_j$ with their adjacency labels. Since $\sigma_j$ preserves adjacency and pairs each child with an isomorphic $(i-1)$-subtree, we obtain an isomorphism

$$\mathrm{UNR}^\delta[G, s, i] \cong \mathrm{UNR}^\delta[G, t, i].$$

**Final conclusion.** By induction, all depths match. Hence the stable colours satisfy

$$C^k_{\infty,\delta}(s) = C^k_{\infty,\delta}(t).$$

Therefore **folklore $k$-WL is at least as expressive as $\delta$-$k$-WL for all $k > 1$.**

2. **Problem 2:** For $k > 1$, prove that the $\delta$-$k$-WL is at least as expressive as the local-$k$-WL in distinguishing non-isomorphic graphs. (2 Pt.)

**Ans:** The proof relies on the equivalence between stable coloring and the isomorphism of the corresponding unrolling trees, $\mathrm{UNR}^{\mathcal{A}}[G, v, i]$.

**Induction on Depth $i$**

Let $q$ be a sufficiently large depth. The assumption $C^k_{\infty,L,+}(s) = C^k_{\infty,L,+}(t)$ implies $\mathrm{UNR}^{L,+}[G, s, q] \cong \mathrm{UNR}^{L,+}[G, t, q]$. We induct on $i$ to show $\mathrm{UNR}^{\delta}[G, s, i] \cong \mathrm{UNR}^{\delta}[G, t, i]$ for all $i$.

**Base Case $i = 0$:** The isomorphism of the roots of $\mathrm{UNR}^{L,+}[G, s, q]$ and $\mathrm{UNR}^{L,+}[G, t, q]$ implies they have the same atomic type. Since $\mathrm{UNR}^{\delta}[G, v, 0]$ is a single node with the atomic type, $\mathrm{UNR}^{\delta}[G, s, 0] \cong \mathrm{UNR}^{\delta}[G, t, 0]$.

**Inductive Step $i$:** Assume $\mathrm{UNR}^{\delta}[G, x, i-1] \cong \mathrm{UNR}^{\delta}[G, \sigma_j(x), i-1]$ for all $j$-neighbors $x$ and their image $\sigma_j(x)$.

1. **Matching via $\mathbf{L-WL^+}$:** The isomorphism $\mathrm{UNR}^{L,+}[G, s, q] \cong \mathrm{UNR}^{L,+}[G, t, q]$ ensures that for any stable color class $C$ and position $j$:

$$|X_j \cap C| = |Y_j \cap C|,$$

   and, crucially, the counts of **local** (adjacent) and **global** (non-adjacent) neighbors in $C$ are also individually equal for $s$ and $t$.

2. **Bijection $\sigma_j$:** This count equality allows us to construct a bijection $\sigma_j : X_j \to Y_j$ that pairs up $j$-neighbors $x$ and $\sigma_j(x)$ such that they are in the same stable $L$-WL$^+$ color class and share the same adjacency type (local/global).

3. **Isomorphism $\delta$-UNR:** Since $\sigma_j$ preserves the $L$-WL$^+$ color, by the induction hypothesis, the $\delta$-unrolling subtrees are isomorphic: $\mathrm{UNR}^{\delta}[G, x, i-1] \cong \mathrm{UNR}^{\delta}[G, \sigma_j(x), i-1]$. The $\delta$-WL uses the $j$-label and the adjacency type $adj(v, w) \in \{0, 1\}$ (corresponding to local/global) as its edge label. Since $\sigma_j$ preserves both the subtree isomorphism and the adjacency label, we obtain the overall isomorphism for depth $i$:

$$\mathrm{UNR}^{\delta}[G, s, i] \;\cong\; \mathrm{UNR}^{\delta}[G, t, i].$$

**Conclusion** By induction, $C^k_{\infty,\delta}(s) = C^k_{\infty,\delta}(t)$.

3. **Problem 3:** Show that the local $k$-WL$^+$ can be computed in the same asymptotic running time as the local $k$-WL.

**Ans:** We need to show that both algorithms have the same asymptotic complexity of $O(n^k \cdot k \cdot d)$ per iteration, where $n = |V(G)|$ is the number of vertices, $k$ is the tuple dimension, and $d$ is the maximum degree.

**Aggregation functions.** For local $k$-WL:

$$M_{t,L}(v) = \Big(\{\{C_t^{k,L}(\varphi_1(v,w)) \mid w \in N(v_1)\}\}, \ldots, \{\{C_t^{k,L}(\varphi_k(v,w)) \mid w \in N(v_k)\}\}\Big)$$

For local $k$-WL$^+$:

$$M_{t,L,+}(v) = \Big(\{\{(C_t^{k,L,+}(\varphi_j(v,w)), \#_t^j(v, \varphi_j(v,w))) \mid w \in N(v_j)\}\}_{j=1}^{k}\Big)$$

where

$$\#_t^j(v,x) = |\{w : w \sim_j v, C_t^{k,L,+}(w) = C_t^{k,L,+}(x)\}|.$$

**Running time analysis for local $k$-WL.** For each $k$-tuple $v \in V(G)^k$ at iteration $t$:

- For each position $j \in [k]$, iterate over the local neighbors $N(v_j)$.
- For each $w \in N(v_j)$, look up the color $C_t^{k,L}(\varphi_j(v,w))$.
- Collect these colors in a multiset.

Cost per tuple: $O(k \cdot d)$.

Total cost per iteration: $O(n^k \cdot k \cdot d)$.

**Running time analysis for local $k$-WL$^+$.** The # function can be computed efficiently using hash maps without changing the asymptotic complexity.

For each $k$-tuple $v \in V(G)^k$ at iteration $t$:

*First pass – count colors by position:*

- For each position $j \in [k]$, create a hash map $\mathrm{count}_j : \mathbb{N} \to \mathbb{N}$.
- For each $w \in N(v_j)$, compute $x = \varphi_j(v,w)$ and increment $\mathrm{count}_j[C_t^{k,L,+}(x)]$.
- Time per position: $O(|N(v_j)|) = O(d)$.

*Second pass – build multiset with counts:*

- For each position $j \in [k]$ and each $w \in N(v_j)$:
  - Compute $x = \varphi_j(v, w)$.
  - Look up $\#_t^j(v, x) = \text{count}_j[C_t^{k,L,+}(x)]$ (constant time hash lookup).
  - Add pair $(C_t^{k,L,+}(x), \#_t^j(v, x))$ to the multiset.
- Time per position: $O(|N(v_j)|) = O(d)$.

Cost per tuple: $O(k \cdot d) + O(k \cdot d) = O(k \cdot d)$.

Total cost per iteration: $O(n^k \cdot k \cdot d)$.

The # function counts *all* $j$-neighbors (local and global) with a given color, but we only iterate over *local* neighbors in $M_{t,L,+}$.
For a fixed tuple $v$ and position $j$, the set of all $j$-neighbors is fixed and can be processed once per iteration.
Counting colors via hash maps requires only $O(d)$ time per position, the same as collecting the colors themselves.
The additional bookkeeping (counting and lookup) adds only constant factor overhead per neighbor.

**Conclusion.** Both local $k$-WL and local $k$-WL$^+$ have the same asymptotic running time of

$$O(n^k \cdot k \cdot d) \text{ per iteration.}$$

4. **Problem 4:** Prove that the local $k$-WL induces a hierarchy. That is, for $k \geq 1$, show that there exists a pair of graphs that the local $k$-WL cannot distinguish but the local $(k+1)$-WL can.

**Ans:** We construct, for each $k \geq 1$, a pair of non-isomorphic graphs $(G_k, H_k)$ such that the local $k$-WL cannot distinguish them, but the local $(k+1)$-WL can.

**Construction of $G_k$ and $H_k$.** Let $K$ denote the complete graph on $k+1$ vertices (without self-loops). The vertices of $K$ are numbered from 0 to $k$. Let $E(v)$ denote the set of edges incident to $v$ in $K$. Clearly, $|E(v)| = k$ for all $v \in V(K)$.

*Graph $G_k$:*

1. For the vertex set $V(G_k)$, we add:

- $(v, S)$ for each $v \in V(K)$ and for each *even* subset $S$ of $E(v)$ (including $\emptyset$).
- Two vertices $e_0, e_1$ for each edge $e \in E(K)$.

2. For the edge set $E(G_k)$, we add:
   - An edge $\{e_0, e_1\}$ for each $e \in E(K)$.
   - An edge between $(v, S)$ and $e_1$ if $v \in e$ and $e \in S$.
   - An edge between $(v, S)$ and $e_0$ if $v \in e$ and $e \notin S$.

*Graph $H_k$:* Define $H_k$ similarly to $G_k$, with the following exception:

- For vertex $0 \in V(K)$, we choose all *odd* subsets of $E(0)$ instead of even subsets.

Note that $|V(G_k)| = |V(H_k)| = (k+1) \cdot 2^{k-1} + \binom{k+1}{2} \cdot 2$.

**The graphs are non-isomorphic.** We use the concept of a *distance-two-clique*: a set $S$ of vertices is a distance-two-clique if the distance between any two vertices in $S$ is exactly two.

1. There exists a distance-two-clique of size $(k+1)$ in $G_k$.
2. There does not exist a distance-two-clique of size $(k+1)$ in $H_k$.

Hence $G_k$ and $H_k$ are non-isomorphic.

*Part 1:* In $G_k$, consider $S = \{(0, \emptyset), (1, \emptyset), \ldots, (k, \emptyset)\}$ of size $k+1$. For any $i, j \in V(K)$ with $i \neq j$, vertex $(i, \emptyset)$ is adjacent to $\{i, j\}_0$, which is adjacent to $(j, \emptyset)$. Thus any two vertices in $S$ are at distance two.

*Part 2:* Suppose there exists a distance-two-clique $\{(0, S_0), (1, S_1), \ldots, (k, S_k)\}$ in $H_k$, where $S_i \subseteq E(i)$.

Computing the parity-sum of $|S_0|, \ldots, |S_k|$: Since $S_0$ is odd and all others are even, the total parity is 1.

Computing edge-by-edge: For each edge $\{i, j\} \in E(K)$, since $(i, S_i)$ and $(j, S_j)$ are at distance two, either both $S_i$ and $S_j$ contain $\{i, j\}$ or neither does. Thus the parity contribution of $\{i, j\}$ to the sum is 0.

Since each edge contributes 0, the total parity-sum must be 0, a contradiction. Hence no such distance-two-clique exists in $H_k$.

**The local $k$-WL cannot distinguish $G_k$ and $H_k$.** This follows from classical WL theory. The graphs $G_k$ and $H_k$ are the standard Cai-Fürer-Immerman (CFI) construction, which is known to fool the $k$-WL. Since the local $k$-WL only uses local neighbors, which is a subset of the information used by the standard $k$-WL, it follows that:

$$\text{local } k\text{-WL} \preceq k\text{-WL}.$$

Therefore, the local $k$-WL also cannot distinguish $G_k$ and $H_k$.

**The local $(k+1)$-WL can distinguish $G_k$ and $H_k$.** We show that the local $(k+1)$-WL can detect the distance-two-clique in $G_k$.

Consider the $(k+1)$-tuple

$$P = ((0, \emptyset), (1, \emptyset), \ldots, (k, \emptyset)) \in V(G_k)^{k+1}.$$

*Claim:* There is no tuple $Q \in V(H_k)^{k+1}$ such that the local $(k+1)$-WL unrolling of $P$ is isomorphic to the unrolling of $Q$.

[Proof of Claim] Suppose, for contradiction, that such a $Q$ exists. By comparing atomic types, $Q$ must be of the form $((0, S_0), (1, S_1), \ldots, (k, S_k))$.

Consider the depth-two unrolling of $P$: From the root $P$, we can go down via two local edges labeled 1 to reach the tuple

$$P_2 = ((1, \emptyset), (1, \emptyset), (2, \emptyset), \ldots, (k, \emptyset)).$$

This is possible because $(0, \emptyset)$ and $(1, \emptyset)$ are at distance two via the edge-gadget.

For the unrolling of $Q$ to be isomorphic, vertices $(0, S_0)$ and $(1, S_1)$ must be at distance two in $H_k$. Repeating this argument for all pairs of positions, we find that $(i, S_i)$ and $(j, S_j)$ must be at distance two for all $i \neq j$.

Now consider the depth-four unrolling of $P$: From $P$, we can reach $R = ((0, \emptyset), (1, \emptyset), \ldots, (k, \emptyset))$ with position 1 replaced by $(0, \emptyset)$. This shows that $(0, \emptyset)$ is at distance two from all $(i, \emptyset)$ for $i \in \{1, \ldots, k\}$.

Similarly, in $H_k$, vertex $(0, S_0)$ must be at distance two from all $(i, S_i)$ for $i \in \{1, \ldots, k\}$. Combined with the pairwise distances, this means $\{(0, S_0), (1, S_1), \ldots, (k, S_k)\}$ forms a distance-two-clique of size $k + 1$ in $H_k$, contradicting the lemma.

Since no such $Q$ exists, the local $(k+1)$-WL assigns different colors to tuples in $G_k$ and $H_k$, and thus distinguishes the graphs.

**Conclusion.** For each $k \geq 1$, the pair $(G_k, H_k)$ witnesses that the local $(k+1)$-WL is strictly more powerful than the local $k$-WL. Therefore, the local $k$-WL induces a strict hierarchy:

$$\text{local 1-WL} \prec \text{local 2-WL} \prec \text{local 3-WL} \prec \cdots$$

5. **Problem 5:** The random-walk kernel counts how many walks two graphs have in common. To define it, we need to introduce the direct product graph of two graphs. Let $G$ and $H$ be two node-labeled graphs. Then the direct product graph $G \times H$ of the two graphs is the graph with the vertex set

$$V(G \times H) := \{(g, h) \mid g \in V(G), h \in V(H), l(g) = l(h)\},$$

and the edge set

$$E(G \times H) := \{((g, h), (s, t)) \mid (g, s) \in E(G), (h, t) \in E(H)\}.$$

Now given two graphs $G$ and $H$ and $k > 0$, the random-walk kernel

$$k_{\mathrm{RW}}(G, H) := \sum_{i,j=1}^{|V(G \times H)|} \left[ \sum_{l=1}^{k} \lambda_l A(G \times H)^l \right]_{ij},$$

where $\lambda_1, \ldots, \lambda_k$ with $\lambda_i > 0$ is a sequence of weights ensuring convergence, e.g., when $k = \infty$. Show that the random-walk kernel for any $k > 0$ cannot distinguish more non-isomorphic graphs than the 1-WL.

**Ans:** We need to prove that if the 1-WL cannot distinguish two graphs $G$ and $H$, then $k_{\mathrm{RW}}(G, G) = k_{\mathrm{RW}}(H, H)$ for any $k > 0$. This proves that the random-walk kernel cannot distinguish more non-isomorphic graphs than the 1-WL.

**Random walks and color refinement.** The random-walk kernel counts walks of different lengths in the direct product graph. The crucial insight is that walks in $G$ correspond to sequences of vertex labels and adjacencies, which are exactly the information captured by the 1-WL color refinement.

Let $C^1_t : V(G) \to \mathbb{N}$ denote the coloring of $G$ after $t$ iterations of the 1-WL, and similarly for $H$.

$$C^1_0(v) = l(v) \quad \text{(initial vertex label)}$$

$$C^1_{t+1}(v) = \mathrm{Recolor}\left(C^1_t(v), \{\{C^1_t(w) \mid w \in N(v)\}\}\right).$$

The 1-WL assigns the same stable coloring to $G$ and $H$ if and only if, for all $t \geq 0$, the multisets of colors $\{\{C^1_t(v) \mid v \in V(G)\}\}$ and $\{\{C^1_t(v) \mid v \in V(H)\}\}$ are equal.

**Relating walks to 1-WL colors.** A walk of length $l$ in a graph $G$ starting at vertex $v$ is a sequence $v = v_0, v_1, \ldots, v_l$ where $(v_i, v_{i+1}) \in E(G)$ for all $i$.

*Claim:* Two vertices $v \in V(G)$ and $w \in V(H)$ have the same 1-WL color after $t$ iterations if and only if they have the same distribution of walks of length up to $t$ (counted by the sequence of labels encountered).

[Proof] By induction on $t$.

*Base case $t = 0$:* $C_0^1(v) = C_0^1(w)$ if and only if $l(v) = l(w)$, which means they have the same "walk of length 0" (i.e., the vertex label itself).

*Inductive step:* Assuming that the claim holds for $t-1$. Then $C_t^1(v) = C_t^1(w)$ iff:

- $C_{t-1}^1(v) = C_{t-1}^1(w)$ (same walks up to length $t-1$), and
- $\{\{C_{t-1}^1(u) \mid u \in N(v)\}\} = \{\{C_{t-1}^1(u') \mid u' \in N(w)\}\}$ (same multiset of neighbor colors).

The second condition means that for every walk of length $t-1$ starting at a neighbor of $v$, there exists a corresponding walk starting at a neighbor of $w$ with the same label sequence (and vice versa). Combined with the first condition, this means $v$ and $w$ have the same distribution of walks of length up to $t$.

**Counting walks in the direct product.** The entry $[A(G \times H)^l]_{(g,h),(s,t)}$ counts the number of walks of length $l$ from $(g,h)$ to $(s,t)$ in the direct product graph $G \times H$.

A walk from $(g,h)$ to $(s,t)$ in $G \times H$ corresponds to a pair of walks:

- A walk $g = g_0, g_1, \ldots, g_l = s$ in $G$, and
- A walk $h = h_0, h_1, \ldots, h_l = t$ in $H$,

such that $l(g_i) = l(h_i)$ for all $i \in \{0, \ldots, l\}$ (since vertices in $G \times H$ require matching labels).

Therefore:

$$\sum_{i,j=1}^{|V(G \times H)|} [A(G \times H)^l]_{ij}$$

counts the total number of walks of length $l$ in $G \times H$, summed over all pairs of starting and ending vertices.

**1-WL indistinguishability implies equal kernel values.** Assume the 1-WL cannot distinguish $G$ and $H$, i.e., for all $t \geq 0$:

$$\{\{C_t^1(v) \mid v \in V(G)\}\} = \{\{C_t^1(v) \mid v \in V(H)\}\}.$$

We compute $k_{\text{RW}}(G, G)$ and $k_{\text{RW}}(H, H)$.

For $G \times G$:

- $V(G \times G) = \{(g, g') \mid g, g' \in V(G), l(g) = l(g')\}$.
- A walk of length $l$ in $G \times G$ from $(g_1, g_2)$ to $(s_1, s_2)$ corresponds to two walks in $G$: one from $g_1$ to $s_1$ and one from $g_2$ to $s_2$, where the label sequences match at each step.

Similarly for $H \times H$.

The number of walks of length $l$ in $G \times G$ depends only on:

1. The number of vertices of each label (given by $C_0^1$).
2. The distribution of walks of each length up to $l$ from vertices of each color (given by $C_t^1$ for $t \leq l$).

Since the 1-WL assigns the same color distribution to $G$ and $H$ at all iterations, the graphs have:

- The same number of vertices with each label.
- The same distribution of walks of each length from vertices of each 1-WL color.

Therefore, for each $l$:

$$\sum_{i,j=1}^{|V(G \times G)|} [A(G \times G)^l]_{ij} = \sum_{i,j=1}^{|V(H \times H)|} [A(H \times H)^l]_{ij}.$$

**Conclusion.** Since the above equality holds for all $l$, and the kernel is a weighted sum over $l$:

$$k_{\text{RW}}(G, G) = \sum_{l=1}^{k} \lambda_l \sum_{i,j} [A(G \times G)^l]_{ij} = \sum_{l=1}^{k} \lambda_l \sum_{i,j} [A(H \times H)^l]_{ij} = k_{\text{RW}}(H, H).$$

Therefore, if the 1-WL cannot distinguish $G$ and $H$, then $k_{\text{RW}}(G, G) = k_{\text{RW}}(H, H)$, which means the random-walk kernel (as a graph feature for classification) cannot distinguish them either.

Hence, the random-walk kernel for any $k > 0$ cannot distinguish more non-isomorphic graphs than the 1-WL.

6. **Problem 6:** Given an $n$-order graph $G$ of maximum degree $d$, devise an efficient polynomial-time algorithm (in $n$ and constant $d$) for computing the feature map of the Graphlet kernel for $k = 4$. Simply enumerating all 4-Graphlets and then counting them is not allowed.

**Ans:** We devise an algorithm that computes the feature map for 4-node graphlets (connected subgraphs on 4 vertices) in time $O(n \cdot d^3)$, which is polynomial in $n$ when $d$ is constant.

**Overview of 4-node connected graphlets.** There are exactly 6 non-isomorphic connected graphs on 4 vertices:

1. $P_4$: Path with 3 edges (1-2-3-4)
2. $C_4$: 4-cycle (square)
3. $K_{1,3}$: Star with 3 edges (center connected to 3 leaves)
4. $P_3 + e$: Path with one chord (triangle + tail)
5. $K_4 - e$: Complete graph minus one edge (5 edges)
6. $K_4$: Complete graph (6 edges)

**Key idea: Avoid explicit enumeration.** Instead of enumerating all $\binom{n}{4}$ possible 4-vertex subsets (which would be $O(n^4)$), we exploit the bounded degree to enumerate only **relevant** 4-vertex sets—those that actually form connected subgraphs.

**Strategy**: For each vertex $v$, consider all connected 4-graphlets containing $v$ by exploring the local neighborhood up to distance 3 from $v$. Due to bounded degree $d$, this neighborhood has size at most $O(d^3)$.

**Algorithm: Efficient 4-Graphlet Counting**

**Input:** Graph $G = (V, E)$ with $|V| = n$ and maximum degree $d$
**Output:** Feature vector $\phi(G) \in \mathbb{R}^6$ counting each graphlet type
Initialize $\phi(G) = [0, 0, 0, 0, 0, 0]$ (counts for the 6 graphlet types)

**for** each vertex $v \in V(G)$ **do**
    `/* Explore the local neighborhood of` $v$ `*/`
    $N_1(v) \leftarrow N(v)$ `/* 1-hop neighbors */`
    $N_2(v) \leftarrow \{w \mid \exists u \in N_1(v), w \in N(u), w \neq v\}$ `/* 2-hop neighbors */`
    $N_3(v) \leftarrow \{w \mid \exists u \in N_2(v), w \in N(u), w \notin N_1(v) \cup N_2(v) \cup \{v\}\}$ `/* 3-hop neighbors */`

$\mathcal{N}(v) \leftarrow N_1(v) \cup N_2(v) \cup N_3(v)$ `/* All vertices within distance 3 */`

`/* Enumerate all 4-vertex sets containing` $v$ `within` $\mathcal{N}(v) \cup \{v\}$ `*/`

**for** each triple $\{a, b, c\} \subseteq \mathcal{N}(v)$ with $a < b < c$ (lexicographically) **do**
  $S \leftarrow \{v, a, b, c\}$
  **if** $G[S]$ is connected **then**
    Determine the isomorphism type $\tau$ of $G[S]$ among the 6 graphlet types
    $\phi(G)[\tau] \leftarrow \phi(G)[\tau] + 1$
  **end if**
**end for**
**end for**

`/* Correct for overcounting: each graphlet counted once per vertex in it */`
$\phi(G) \leftarrow \phi(G)/4$
**return** $\phi(G)$

**Why this works:**

- **Completeness**: Every connected 4-graphlet has diameter at most 3 (e.g., path $P_4$ has diameter 3). Therefore, all 4 vertices of any connected graphlet lie within distance 3 of at least one of its vertices. By considering all vertices $v$ and their 3-hop neighborhoods, we find all connected 4-graphlets.
- **Correctness of counting**: Each 4-graphlet is counted exactly 4 times—once for each vertex in it (when that vertex is chosen as the "center" $v$). Dividing by 4 gives the correct count.

**Running time analysis:**

For each vertex $v \in V(G)$:

- Computing $N_1(v)$: $O(d)$
- Computing $N_2(v)$: $O(d \cdot d) = O(d^2)$ (at most $d$ neighbors, each with degree $\leq d$)
- Computing $N_3(v)$: $O(d^2 \cdot d) = O(d^3)$
- $|\mathcal{N}(v)| \leq d + d^2 + d^3 = O(d^3)$
- Enumerating triples from $\mathcal{N}(v)$: $\binom{|\mathcal{N}(v)|}{3} = O(d^9)$

- Checking connectivity and isomorphism type of $G[S]$: $O(1)$ (4 vertices, constant time)

Cost per vertex: $O(d^9)$ (dominated by triple enumeration)

Total cost: $O(n \cdot d^9)$

**Important note**: While this is $O(d^9)$ per vertex, for **constant $d$**, this is $O(n)$ overall, which is polynomial in $n$.

**Optimization to reduce the exponent:**

We can improve the algorithm by being smarter about enumeration. Instead of considering all triples in $\mathcal{N}(v)$, we can:

1. **For each pair** $\{a, b\} \subseteq N_1(v)$ (the 1-hop neighbors):
   - Enumerate $c \in N_1(v) \cup N(a) \cup N(b)$
   - Check if $\{v, a, b, c\}$ is connected
   - Cost: $O(d^2 \cdot d) = O(d^3)$ per vertex
2. **For each** $a \in N_1(v)$ **and each pair** $\{b, c\} \subseteq N(a) \setminus \{v\}$:
   - Check if $\{v, a, b, c\}$ is connected
   - Cost: $O(d \cdot d^2) = O(d^3)$ per vertex

This improved algorithm achieves **$O(n \cdot d^3)$ total time**.

**Detailed optimized algorithm:**

```
Initialize φ(G) = [0, 0, 0, 0, 0, 0]
Initialize hash table H to track processed 4-sets

for each vertex v ∈ V(G) do
  /* Case 1: v and two of its neighbors + one more */
  for each pair {a, b} ⊆ N(v) with a < b do
    C ← N(v) ∪ N(a) ∪ N(b)
    for each c ∈ C with c > max(v, a, b) do
      S ← {v, a, b, c} (in sorted order)
      if S ∉ H and G[S] is connected then
        Add S to H
        Identify isomorphism type τ of G[S]
        φ(G)[τ] ← φ(G)[τ] + 1
      end if
    end for
```

        **end for**

        /* Case 2: $v$, one neighbor $a$, and two neighbors of $a$ */
        **for** each $a \in N(v)$ **do**
            **for** each pair $\{b, c\} \subseteq N(a) \setminus \{v\}$ with $b < c$ **do**
                $S \leftarrow \{v, a, b, c\}$ (in sorted order)
                **if** $S \notin \mathcal{H}$ and $G[S]$ is connected **then**
                    Add $S$ to $\mathcal{H}$
                    Identify isomorphism type $\tau$ of $G[S]$
                    $\phi(G)[\tau] \leftarrow \phi(G)[\tau] + 1$
                **end if**
            **end for**
        **end for**
    **end for**
    **return** $\phi(G)$

**Analysis of optimized algorithm:**

- Case 1: $O(d^2)$ pairs $\{a, b\}$, each considers $O(d)$ candidates for $c$: $O(d^3)$ per vertex
- Case 2: $O(d)$ neighbors $a$, each has $O(d^2)$ pairs: $O(d^3)$ per vertex
- Hash table operations: $O(1)$ expected time
- Isomorphism type identification: $O(1)$ for 4 vertices (check edge pattern)

Total time: $O(n \cdot d^3)$, which is **polynomial in $n$ for constant $d$**.

**Identifying isomorphism types efficiently:** For a 4-vertex induced subgraph $G[S]$ with $m$ edges:

- $m = 3$: Check structure (path vs star vs triangle+tail)
- $m = 4$: Must be $C_4$ (4-cycle)
- $m = 5$: Must be $K_4 - e$
- $m = 6$: Must be $K_4$

This can be done in $O(1)$ time by checking the degree sequence and specific edges.

7. **Problem 7:** Let $G$ be a graph. Consider the following variant of the 1-WL, computing a coloring $D^1_t : V(G) \to \mathbb{N}$, for $t > 0$, where

$$D^1_t(v) = \text{Recolor}\left(D^1_{(t-1)}(v), \{\{D^1_{(t-1)}(w) \mid w \in N(v)\}\}, \{\{D^1_{(t-1)}(w) \mid w \notin N(v)\}\}\right).$$

Show that the above variant of the 1-WL has the same expressivity as the ordinary 1-WL in distinguishing non-isomorphic graphs.

**Ans:** We need to show that this variant (which we call D-1-WL) and the ordinary 1-WL have the same expressivity, i.e., they distinguish exactly the same pairs of non-isomorphic graphs.

- Let $C_t^1(v)$ denote the color of vertex $v$ after $t$ iterations of the ordinary 1-WL.
- Let $D_t^1(v)$ denote the color of vertex $v$ after $t$ iterations of the D-1-WL variant.

Recall the ordinary 1-WL refinement:

$$C_t^1(v) = \text{Recolor}\left(C_{t-1}^1(v), \{\{C_{t-1}^1(w) \mid w \in N(v)\}\}\right).$$

The D-1-WL additionally considers the multiset of colors of **non-neighbors**:

$$D_t^1(v) = \text{Recolor}\left(D_{t-1}^1(v), \{\{D_{t-1}^1(w) \mid w \in N(v)\}\}, \{\{D_{t-1}^1(w) \mid w \notin N(v)\}\}\right).$$

### Direction 1: Ordinary 1-WL $\preceq$ D-1-WL

This direction is immediate since D-1-WL uses more information than ordinary 1-WL. Specifically:

- D-1-WL considers both the neighbor multiset and the non-neighbor multiset.
- Ordinary 1-WL only considers the neighbor multiset.

Therefore, if ordinary 1-WL can distinguish two graphs $G$ and $H$, then D-1-WL can also distinguish them, since it has access to all the information that ordinary 1-WL uses (and more).

Formally: If $C_\infty^1(v) \neq C_\infty^1(w)$ for some vertices $v \in V(G)$ and $w \in V(H)$ that would otherwise correspond under an isomorphism, then the same reasoning applies to $D_\infty^1$, implying $D_\infty^1(v) \neq D_\infty^1(w)$.

### Direction 2: D-1-WL $\preceq$ Ordinary 1-WL

We need to show that the additional information (non-neighbor multiset) does not increase the distinguishing power.

**Key observation**: The non-neighbor multiset is redundant given the neighbor multiset and the total color distribution in the graph.

**Why the non-neighbor information is redundant:**

For a vertex $v$ in a graph $G$ with $n$ vertices, let:

- $M_N(v) = \{\{D^1_{t-1}(w) \mid w \in N(v)\}\}$ be the neighbor multiset.
- $M_{\overline{N}}(v) = \{\{D^1_{t-1}(w) \mid w \notin N(v)\}\}$ be the non-neighbor multiset.
- $M_{\text{total}}(G) = \{\{D^1_{t-1}(w) \mid w \in V(G)\}\}$ be the total multiset of colors in $G$.

Then we have the fundamental relation:

$$M_{\text{total}}(G) = \{\{D^1_{t-1}(v)\}\} \uplus M_N(v) \uplus M_{\overline{N}}(v),$$

where $\uplus$ denotes multiset union.

Therefore:

$$M_{\overline{N}}(v) = M_{\text{total}}(G) \setminus \left(\{\{D^1_{t-1}(v)\}\} \uplus M_N(v)\right),$$

where $\setminus$ denotes multiset difference.

This shows that $M_{\overline{N}}(v)$ is completely determined by:

1. The total color distribution $M_{\text{total}}(G)$ (global information)
2. The vertex's own color $D^1_{t-1}(v)$ (local information)
3. The neighbor multiset $M_N(v)$ (local information)

**Simulating D-1-WL with ordinary 1-WL:**

We can simulate the D-1-WL using a modified version of ordinary 1-WL that includes global color histograms.

Define an augmented 1-WL variant where the color refinement is:

$$\tilde{C}^1_t(v) = \text{Recolor}\left(\tilde{C}^1_{t-1}(v), \{\{\tilde{C}^1_{t-1}(w) \mid w \in N(v)\}\}, M_{\text{total}}(G, t-1)\right),$$

where $M_{\text{total}}(G, t-1)$ is the color histogram of $G$ at iteration $t-1$.

**Claim**: This augmented variant has the same distinguishing power as ordinary 1-WL.

[Proof] The ordinary 1-WL on the disjoint union $G \sqcup H$ implicitly captures the global color distributions. When we run ordinary 1-WL on $G \sqcup H$:

- Two vertices $v \in V(G)$ and $w \in V(H)$ can only get the same stable color if they have the same neighborhood structure.

- The stable partition of $V(G) \sqcup V(H)$ ensures that if $G$ and $H$ have different color distributions at any level, this will eventually be reflected in different stable colors.

More precisely, if ordinary 1-WL distinguishes $G$ and $H$, it means either:

1. The color histograms differ: $|\{v \in V(G) \mid C^1_\infty(v) = c\}| \neq |\{w \in V(H) \mid C^1_\infty(w) = c\}|$ for some color $c$, or
2. There exist $v \in V(G)$ and $w \in V(H)$ with the same initial label but different stable colors due to neighborhood differences.

In both cases, the D-1-WL will also distinguish them, because:

- If the color histograms differ, then $M_{\text{total}}(G) \neq M_{\text{total}}(H)$.
- If neighborhood structures differ for some vertices, the neighbor multisets already capture this (the non-neighbor information is redundant as shown above).

**Formal argument using the redundancy:**

Suppose D-1-WL distinguishes two graphs $G$ and $H$, but ordinary 1-WL does not. This means:

- Ordinary 1-WL: The stable color histograms of $G$ and $H$ are identical, i.e., $\{\{C^1_\infty(v) \mid v \in V(G)\}\} = \{\{C^1_\infty(w) \mid w \in V(H)\}\}$.
- D-1-WL: The stable color histograms differ, or vertices with matching ordinary 1-WL colors get different D-1-WL colors.

But if ordinary 1-WL assigns the same color histograms, then:

$$M_{\text{total}}(G, t) = M_{\text{total}}(H, t) \text{ for all } t \geq 0.$$

Now, for any pair of vertices $v \in V(G)$ and $w \in V(H)$ with $C^1_{t-1}(v) = C^1_{t-1}(w)$, we also have:

- $\{\{C^1_{t-1}(u) \mid u \in N(v)\}\} = \{\{C^1_{t-1}(u') \mid u' \in N(w)\}\}$ (by definition of color refinement).
- Since $M_{\text{total}}(G, t-1) = M_{\text{total}}(H, t-1)$ and the neighbor multisets are equal, we must also have:

$$\{\{C^1_{t-1}(u) \mid u \notin N(v)\}\} = \{\{C^1_{t-1}(u') \mid u' \notin N(w)\}\}.$$

Therefore, the non-neighbor multisets must also be equal, which means $D_t^1(v) = D_t^1(w)$.

This contradicts the assumption that D-1-WL distinguishes $G$ and $H$ while ordinary 1-WL does not.

**Conclusion.** We have shown both directions:

- Ordinary 1-WL $\preceq$ D-1-WL (trivial, since D-1-WL uses more information)
- D-1-WL $\preceq$ Ordinary 1-WL (non-trivial, using the redundancy argument)

Hence,

$$\text{Ordinary 1-WL} \equiv \text{D-1-WL}.$$

8. **Problem 8:** Let $X$ be a set, and let $k_1 : X \times X \to \mathbb{R}$ and $k_2 : X \times X \to \mathbb{R}$ be two kernels with feature maps $\phi_1 : X \to \mathbb{R}^d$ and $\phi_2 : X \to \mathbb{R}^e$, respectively. Show that $k_+(x, y) = k_1(x, y)^2 + k_2(x, y)^3$ for $x, y \in X$ is a valid positive semidefinite kernel.

**Ans:** We have to prove that $k_+$ is a valid kernel by proving it is positive semidefinite, i.e., for any $n \geq 1$ and any $x_1, \ldots, x_n \in X$, the Gram matrix $K_+$ with entries $(K_+)_{ij} = k_+(x_i, x_j)$ is positive semidefinite.

We use the fact that kernels are closed under certain operations:

- Addition: If $k$ and $k'$ are kernels, then $k + k'$ is a kernel.
- Multiplication: If $k$ and $k'$ are kernels, then $k \cdot k'$ is a kernel (pointwise product).
- Powers: If $k$ is a kernel, then $k^n$ (product of $k$ with itself $n$ times) is a kernel.

**Step 1: Show $k_1^2$ is a kernel.**

Since $k_1$ is a kernel, we can write $k_1(x, y) = \langle \phi_1(x), \phi_1(y) \rangle$.

The pointwise product of two kernels is a kernel. Therefore:

$$k_1^2(x, y) = k_1(x, y) \cdot k_1(x, y)$$

is a kernel.

*Explicit feature map*: We can construct a feature map for $k_1^2$ using the tensor product:

$$\psi_1(x) = \phi_1(x) \otimes \phi_1(x) \in \mathbb{R}^{d \times d}.$$

Then:

$$k_1^2(x,y) = \langle \phi_1(x), \phi_1(y) \rangle^2 = \langle \phi_1(x) \otimes \phi_1(x), \phi_1(y) \otimes \phi_1(y) \rangle = \langle \psi_1(x), \psi_1(y) \rangle.$$

**Step 2: Show $k_2^3$ is a kernel.**

Similarly, since $k_2$ is a kernel with $k_2(x,y) = \langle \phi_2(x), \phi_2(y) \rangle$:

$$k_2^3(x,y) = k_2(x,y) \cdot k_2(x,y) \cdot k_2(x,y)$$

is a kernel (product of three kernels).

*Explicit feature map*: Using the tensor product:

$$\psi_2(x) = \phi_2(x) \otimes \phi_2(x) \otimes \phi_2(x) \in \mathbb{R}^{e \times e \times e}.$$

Then:

$$k_2^3(x,y) = \langle \phi_2(x), \phi_2(y) \rangle^3 = \langle \psi_2(x), \psi_2(y) \rangle.$$

**Step 3: Show $k_+ = k_1^2 + k_2^3$ is a kernel.**

Since kernels are closed under addition, and we have shown that $k_1^2$ and $k_2^3$ are both kernels:

$$k_+(x,y) = k_1^2(x,y) + k_2^3(x,y)$$

is a kernel.

*Explicit feature map*: We can construct a feature map for $k_+$ by concatenating:

$$\phi_+(x) = [\psi_1(x), \psi_2(x)] \in \mathbb{R}^{d^2+e^3}.$$

Then:

$$k_+(x,y) = \langle \psi_1(x), \psi_1(y) \rangle + \langle \psi_2(x), \psi_2(y) \rangle = \langle \phi_+(x), \phi_+(y) \rangle.$$

**Verification of positive semidefiniteness:**

For any $n$ points $x_1, \ldots, x_n \in X$ and any coefficients $c_1, \ldots, c_n \in \mathbb{R}$:

$$\begin{aligned}
\sum_{i,j=1}^{n} c_i c_j k_+(x_i, x_j) &= \sum_{i,j=1}^{n} c_i c_j \left[k_1(x_i, x_j)^2 + k_2(x_i, x_j)^3\right] \\
&= \sum_{i,j=1}^{n} c_i c_j k_1(x_i, x_j)^2 + \sum_{i,j=1}^{n} c_i c_j k_2(x_i, x_j)^3 \\
&= \sum_{i,j=1}^{n} c_i c_j \langle \psi_1(x_i), \psi_1(x_j) \rangle + \sum_{i,j=1}^{n} c_i c_j \langle \psi_2(x_i), \psi_2(x_j) \rangle \\
&= \left\langle \sum_{i=1}^{n} c_i \psi_1(x_i), \sum_{j=1}^{n} c_j \psi_1(x_j) \right\rangle + \left\langle \sum_{i=1}^{n} c_i \psi_2(x_i), \sum_{j=1}^{n} c_j \psi_2(x_j) \right\rangle \\
&= \left\| \sum_{i=1}^{n} c_i \psi_1(x_i) \right\|^2 + \left\| \sum_{i=1}^{n} c_i \psi_2(x_i) \right\|^2 \\
&\geq 0.
\end{aligned}$$

Therefore, the Gram matrix is positive semidefinite, confirming that $k_+$ is a valid kernel.

**Conclusion.** The function $k_+(x, y) = k_1(x, y)^2 + k_2(x, y)^3$ is a valid positive semidefinite kernel, with feature map $\phi_+(x) = [\phi_1(x) \otimes \phi_1(x), \phi_2(x) \otimes \phi_2(x) \otimes \phi_2(x)]$.