Color-Coding for k-Cycle Detection In Graphs

1 Problem and Model

- Task: detect a simple k-cycle ($k \geq 3$) in an undirected simple graph $G = (V, E)$ under the RAM model with unit-cost $\Theta(\log n)$-bit operations.
- Output: vertices $(v_0, \ldots, v_{k-1})$ with $(v_i, v_{i+1}) \in E$ for all i (indices modulo k), all vertices distinct.
- If none exists, the algorithm may report failure; color-coding is Monte Carlo with tunable failure probability, DFS is exact but exponential.

2 Algorithms

We compare a randomized fixed-parameter color-coding algorithm against a deterministic DFS baseline.

2.1 Color-coding with subset DP

Each iteration colors every vertex independently and uniformly with one of k colors. A k-cycle is *colorful* if its vertices receive pairwise distinct colors.

For a coloring, define DP states $\mathsf{DP}[S][v] = 1$ when there exists a simple path ending at v that uses exactly the color set $S \subseteq [k]$ (one occurrence per color). Initialization sets $\mathsf{DP}[\{c(v)\}][v] = 1$ for all v. Transitions over increasing $|S|$ apply

$$\mathsf{DP}[S \cup \{c(u)\}][u] \leftarrow 1 \quad \text{if } \mathsf{DP}[S][v] = 1,\ (v, u) \in E,\ c(u) \notin S.$$

A colorful k-cycle exists if some state with $|S| = k$ closes an edge to its start vertex; parent pointers reconstruct the vertex sequence.

Lemma 1 (Correctness of one coloring)**.** *If a coloring makes some simple k-cycle colorful, the DP discovers it and returns a valid k-cycle.*

Proof. Proceed by induction on $|S|$. Initialization establishes all length-1 colorful paths. The transition preserves simplicity because a new color implies a previously unseen vertex; adjacency ensures a valid extension. Thus any colorful path of color set S ending at v sets $\mathsf{DP}[S][v] = 1$. For a colorful k-cycle $(v_0, \ldots, v_{k-1})$, the DP marks $\mathsf{DP}[\,[k]\,][v_{k-1}] = 1$. Since $(v_{k-1}, v_0) \in E$, the closing check reconstructs exactly the original cycle. □

Lemma 2 (Single-coloring success probability)**.** *For any fixed simple k-cycle, a random coloring makes it colorful with probability $p(k) = k!/k^k \geq e^{-k}$.*

Proof. All k^k colorings are equally likely. The cycle is colorful under $k!$ injective colorings, hence probability $k!/k^k$. Stirling's bound $k! \geq (k/e)^k$ yields $k!/k^k \geq e^{-k}$. □

To reduce failure probability below δ, we perform

$$R(k) = \left\lceil \frac{\ln(1/\delta)}{p(k)} \right\rceil$$

independent colorings.

Theorem 1 (Amplified success)**.** *With $R(k)$ colorings, the probability that a k-cycle present in G is missed is at most δ.*

Proof. Lemma 2 gives per-trial success probability at least $p(k)$. Independence yields failure probability $(1 - p(k))^{R(k)} \leq e^{-p(k)R(k)} \leq e^{-\ln(1/\delta)} = \delta$. □

Complexity. The DP has $O(2^k n)$ states and examines each incident edge per state, giving $O(2^k m)$ time and $O(2^k n)$ space per coloring. Total expected time is $O(R(k)\, 2^k m)$.

2.2 Depth-limited DFS baseline

DFS enumerates simple paths of length $k-1$ from each start vertex, accepting when the final edge closes a k-cycle. Let d be the average branching factor; worst-case time is $\Theta(nd^{k-1})$ and exponential in k.

Theorem 2 (DFS blow-up on the ladder)**.** *For the ladder distribution with width $b \geq 2$, DFS starting at s must enumerate b^{k-1} distinct length-$(k-1)$ paths before encountering the unique closing edge, yielding time $\Omega(b^{k-1})$ even for a single start vertex.*

Proof. Every path from s to L_{k-1} chooses one vertex per layer, giving exactly b^{k-1} simple paths of length $k-1$. Only one vertex in L_{k-1} is adjacent to s. In adversarial adjacency ordering, that vertex is explored last, forcing DFS to traverse all other $b^{k-1} - 1$ paths before closing the cycle. Each traversal costs $\Theta(k)$, so time is $\Omega(b^{k-1})$. □

3 Adversarial Ladder Distribution

All experiments use a single synthetic family that maximally amplifies branching for DFS while guaranteeing a k-cycle:

- Fix $k \geq 3$ and total vertex count n. Set the layer width $b = \max\{1, \lfloor (n-1)/(k-1) \rfloor\}$.
- Create layers $L_0, \ldots, L_{k-1}$ with $|L_0| = 1$ (start vertex s) and $|L_i| = b$ for $i \geq 1$ (truncated to fit n vertices).
- Add all edges between L_i and L_{i+1}, giving a complete bipartite join per consecutive layer.
- Add a single closing edge from one vertex in L_{k-1} back to s. This yields exactly one simple k-cycle but b^{k-1} distinct simple paths of length $k-1$ from s to L_{k-1}.

This construction ensures every instance contains a k-cycle while forcing DFS to enumerate exponentially many candidate paths before encountering the unique closing edge.

4 Experimental Setup

- **Algorithms evaluated:** color-coding with $R(k) = \lceil \ln(10)/p(k) \rceil$ (target failure $\delta = 0.1$) and the DFS baseline above.
- **Instances:** adversarial ladder only; parameters $k \in \{5, 6, 7\}$ and $n \in \{20, 40, 80, 200, 300, 500\}$ with width $b = \lfloor (n-1)/(k-1) \rfloor$; exactly one k-cycle per graph.
- **Metrics and timeouts:** 5 repetitions of 20 graphs each per configuration; time per graph derived from totals. Runs exceeding 20 seconds are terminated and marked as timeouts (observed for DFS on larger ladders).

5 Results

The figures below present mean time per graph versus n for $k \in \{5, 6, 7\}$. Timeout markers (red crosses) indicate DFS exceeded the 20 second cap for the largest ladders.

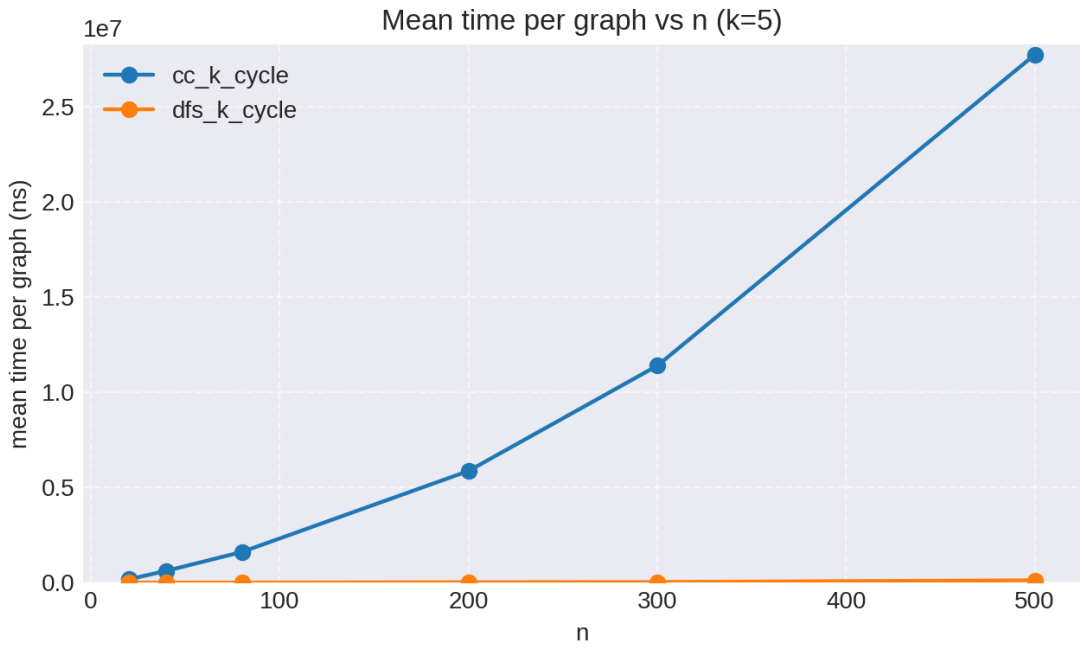

Figure 1: Runtime on ladder graphs for $k = 5$.

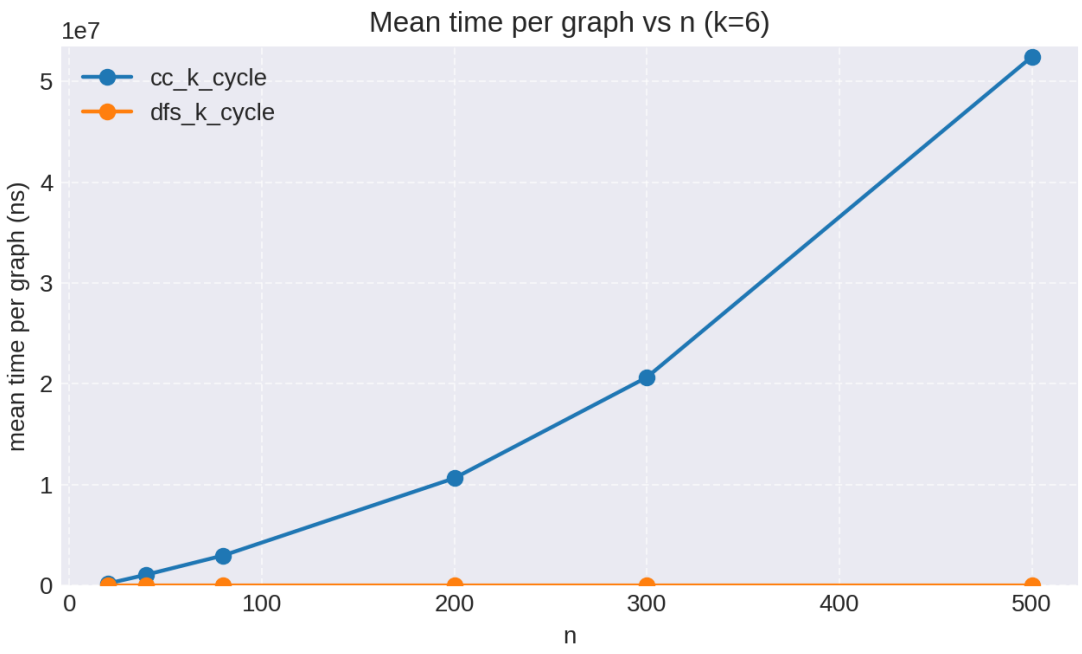

Figure 2: Runtime on ladder graphs for $k = 6$.

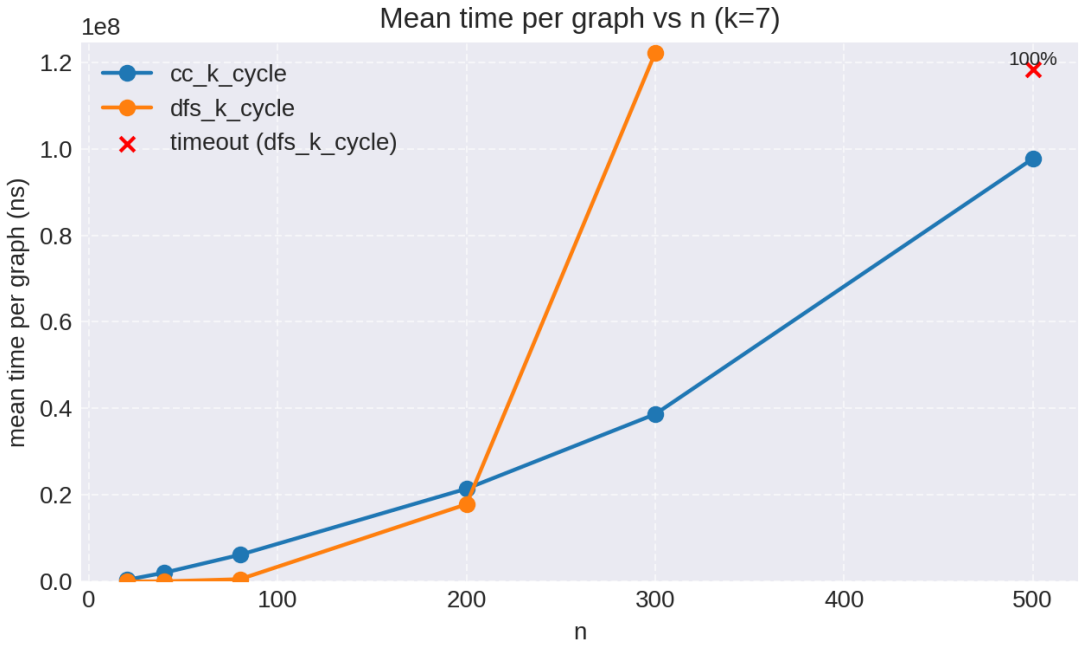

Figure 3: Runtime on ladder graphs for $k = 7$.

5.1 Observations

- **Color-coding scaling:** mean time per graph grows roughly linearly with n at fixed $k \in \{5, 6, 7\}$, consistent with the $O(2^k m)$ dependence when k is fixed.
- **DFS divergence:** DFS matches color-coding for small n but growth accelerates with b^{k-1}; at $k = 7$ and $n = 500$ it times out, consistent with Theorem 4.
- **Stability and error risk:** no empirical errors occurred in simulations; in real deployments, errors remain possible due to adversarial ordering, implementation defects, or hardware nondeterminism.

6 Use Cases and Risks

- **Motif discovery in biology:** short-cycle motifs in protein or gene interaction networks where k is small and sparsity holds.
- **Fraud-ring detection:** small cycles linking accounts in financial transaction graphs; color-coding tolerates moderate noise while keeping failure probability tunable.
- **Routing sanity checks:** fixed-length loops in communication networks that indicate policy violations or misconfigurations.
- **Algorithm suitability:** color-coding is preferable when k is small and graphs are sparse-to-moderate; DFS is viable only when both k and branching are tiny.
- **Operational risk:** the adversarial ladder shows naive DFS can fail catastrophically in graphs with many simple paths even when a single cycle exists; color-coding still has Monte Carlo error that must be controlled via repetitions.

7 Implementation Summary

We implemented both algorithms in C++17 (`kcycle_bench`). The ladder generator, color-coding DP, DFS baseline, and validation are contained in `color_coding/src/main.cpp`. The benchmark driver accepts an `algo_id` and configuration tuple, generates ladder instances deterministically from seeds, times only the algorithm body, and enforces the 20 second timeout. Plots are produced by `scripts/plot_results.py` from `results/raw_results.csv`.