Advanced Interactive Graph Optimization Challenge

Problem Statement

You are given a dynamic weighted graph with n vertices (numbered from 0 to n-1) that evolves through multiple phases. Your goal is to solve a multi-objective optimization problem involving graph coloring, domination, and resource allocation through strategic interactive queries.

Core Challenge: Multi-Phase Graph Evolution

The graph undergoes 3 distinct phases:
1. **Discovery Phase**: Learn the initial graph structure
2. **Adversarial Phase**: An adversary modifies the graph based on your queries
3. **Optimization Phase**: Solve the final multi-objective problem

Multi-Objective Goals

You must simultaneously optimize:
1. **k-Coloring**: Find a proper k-coloring with minimum k (k ≥ 3)
2. **Weighted Domination**: Find a minimum-weight dominating set
3. **Resource Allocation**: Distribute limited resources across vertices optimally

Interactive Protocol

You have access to 5 different query types with varying costs:
- `STRUCTURE x` (Cost: 1) - Returns neighbors and edge weights for vertex x
- `COLOR_CHECK k S` (Cost: 2) - Check if set S can be k-colored properly
- `DOMINATION_WEIGHT S` (Cost: 3) - Returns total weight of dominating set S
- `DISTANCE_MATRIX S` (Cost: 5) - Returns all-pairs shortest paths within set S
- `ADVERSARY_PREDICT` (Cost: 10) - Get hint about adversary's next move

Dynamic Graph Properties

- **Vertex Weights**: Each vertex has a weight $w[i] \in [1, 100]$
- **Edge Weights**: Each edge has weight $e[i,j] \in [1, 50]$
- **Capacity Constraints**: Each vertex has capacity $c[i] \in [1, 20]$
- **Resource Budget**: You have R total resources to allocate

Phase-Specific Rules

Phase 1: Discovery (Rounds 1-T_1)

- Graph is static
- All query types available
- Goal: Learn structure efficiently

Phase 2: Adversarial (Rounds T_1+1-T_2)

- Adversary can add/remove edges based on your query history
- ADVERSARY_PREDICT becomes crucial
- Graph structure changes after every 3 queries

Phase 3: Optimization (Rounds T_2+1-T_3)

- Graph becomes static again
- Must solve the multi-objective problem
- Limited query budget remaining

Constraints and Scoring

- **Total Query Budget**: 4*n + 20 queries across all phases
- **Graph Size**: $5 \le n \le 25$
- **Time Phases**: $T_1 = n$, $T_2 = 2n$, $T_3 = 3n$
- **Resource Budget**: R = 2*n

Scoring Function

```
Score = 1000 - 50*(k-χ) - 10*W_dom - 5*R_waste - 100*P_violations
```

Where:
- k = colors used, χ = chromatic number
- W_dom = weight of your dominating set
- R_waste = unused resources
- P_violations = constraint violations

Multi-Objective Constraints

1. **Coloring Constraint**: Adjacent vertices must have different colors
2. **Domination Constraint**: Every vertex must be dominated
3. **Capacity Constraint**: Resources allocated to vertex i ≤ c[i]
4. **Budget Constraint**: Total resources allocated ≤ R

Input/Output Format

Input

- First line: `n R T₁ T₂ T₃`
- Interactive queries until final answer

Queries

- `STRUCTURE x` → neighbors, weights, vertex_weight, capacity
- `COLOR_CHECK k S` → feasible (true/false)
- `DOMINATION_WEIGHT S` → total_weight, dominates_all
- `DISTANCE_MATRIX S` → matrix of shortest paths
- `ADVERSARY_PREDICT` → hint about next adversarial move

Final Answer

```
ANSWER k=[colors] dominating_set=[vertices] allocation=[resources]
```

Example Interaction

Visible Test Case: n=5, R=10

Phase 1 (Discovery):

```
> STRUCTURE 0
< neighbors=[1,2] weights=[(1,5),(2,3)] vertex_weight=10 capacity=4
> STRUCTURE 1
< neighbors=[0,3,4] weights=[(0,5),(3,2),(4,7)] vertex_weight=8 capacity=3
> COLOR_CHECK 3 [0,1,2,3,4]
< feasible=true
```

Phase 2 (Adversarial):

```
> ADVERSARY_PREDICT
< hint="Will add edge (2,4) if you query vertex 2 again"
> STRUCTURE 3
< neighbors=[1] weights=[(1,2)] vertex_weight=6 capacity=5
[Adversary adds edge (0,3) with weight 4]
```

Phase 3 (Optimization):

```
> DOMINATION_WEIGHT [1,2]
< total_weight=14 dominates_all=true
> ANSWER k=3 dominating_set=[1,2] allocation=[0,3,2,5,0]
< Score: 850/1000 (k-optimal, good domination, efficient allocation)
```

Advanced Algorithmic Challenges

This problem requires:
1. **Multi-Phase Strategy**: Adapt query strategy across phases
2. **Game Theory**: Predict and counter adversarial moves
3. **Multi-Objective Optimization**: Balance competing objectives
4. **Resource Management**: Optimize query budget allocation
5. **Dynamic Programming**: Handle changing graph structure
6. **Approximation Algorithms**: NP-hard subproblems require heuristics

Problem Complexity Analysis

The problem introduces several NP-hard subproblems:
- Graph k-coloring (NP-complete)
- Minimum dominating set (NP-hard)
- Multi-objective optimization under constraints

- Game-theoretic adversarial elements
- Dynamic graph structure handling

Query budget management across phases requires:
- Strategic resource allocation
- Adaptive algorithm selection
- Uncertainty handling and prediction
- Multi-phase optimization coordination