**Parallelized Graph Coloring Algorithm Documentation**

1. Algorithms

1. Introduction

The GraphColoring class provides a parallelized graph coloring algorithm using threads.

The algorithm aims to color the nodes of a graph such that no adjacent nodes share the same color.

Implemented using Java with multi-threading support.

1.1 getColoredGraphRecursive

The recursive function responsible for exploring the search space and finding a valid coloring. It operates on a single node at a time, utilizing parallel threads for efficient exploration.

1.2 Recursive Coloring

Check if a solution is already found (early exit if codes is not empty).

Base case: If the last node is reached, check if the color assignment is valid.

For each color, check validity, and either spawn a new thread or add the color to the list of valid colors.

2. Synchronization

2.1 ReentrantLock

A ReentrantLock is used for synchronization purposes. It ensures that critical sections of code are executed atomically, preventing data corruption or race conditions in multi-threaded environments.

3. Performance Measurements

3.1 Execution Time

The execution time of the parallelized graph coloring algorithm is measured using System.nanoTime() before and after the algorithm execution. This provides insights into the efficiency of the parallelization.

O imagine care conține text, captură de ecran, Font

Descriere generată automat

3.2 Thread Management

The algorithm employs a dynamic thread management strategy. The number of threads is controlled by the threadsNumber parameter. Threads are created to explore different coloring possibilities concurrently, leading to potential performance improvements.

3.3 Scalability

Performance measurements are taken for various graph sizes and complexities to assess the scalability of the algorithm. This includes analyzing how the algorithm's execution time scales with an increasing number of threads and graph size.

4. Conclusion

This documentation provides an overview of the parallelized graph coloring algorithm, detailing the algorithms used, synchronization mechanisms, and performance measurement strategies. It serves as a guide for understanding, implementing, and optimizing the provided code.