Instituição: Universidade Federal de Minas Gerais
Disciplina: Algoritmos e Estruturas de Dados III
Professora: Vinícius
Aluna: Lorena Mendes Peixoto
Matrícula: 2017015002

Atualizações Problemáticas - Documentação

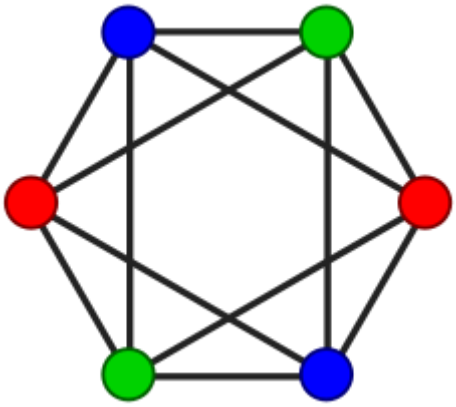
1. Introdução

Na computação e também no cotidiano, é possível deparar-se problemas que envolvem a rotulação ou segregação de itens que compõem um sistema. Por exemplo, alguns remédios, quando em contato, podem reagir negativamente, de modo a prejudicar a saúde de uma pessoa se os tomar ao mesmo tempo. Pacientes em tratamento, que precisam tomar doses diárias de tais remédios, devem dividir a dosagem em horários diferentes, de modo a evitar tais conflitos. Um outro exemplo é o transporte, a um supermercado, de alimentos perecíveis e materiais de limpeza. Este não pode ser feito ao mesmo tempo no mesmo caminhão, tendo em vista uma possível contaminação dos alimentos.

O presente trabalho lida com a seguinte situação: a empresa fictícia BH Software desenvolveu um aplicativo de comunicação vias áudio e vídeo. Este é suportado por uma rede de servidores que são sujeitos a atualizações ocasionais. Por questões de viabilidade (tempo, custos), os servidores não podem ser atualizados um a um e nem todos ao mesmo tempo. Uma solução é alocar “rodadas” nas quais grupos de servidores são atualizados enquanto outros se mantêm em pleno funcionamento. Uma regra importante é que servidores diretamente conectados não podem ser atualizados ao mesmo tempo, uma vez que o tráfego de um servidor offline é transferido para seu vizinho.

A resolução deste problema, bem como os anteriormente apresentados, pode ser dada por meio de algoritmos de coloração de grafos, que visam a rotulação de vértices de modo que adjacentes tenham cores distintas. A figura 1 mostra um exemplo de grafo k-colorado, sendo k = 3, que é o número de cores necessárias para que cada vértice seja adjacente a outro de cor diferente, e de modo que seja utilizado o menor número possível de cores.

Figura 1 - Grafo 3-colorado



Fonte: IME-USP (2016)

2. Objetivos

O objetivo do presente trabalho é implementar um sistema de atualização de servidores, seguindo os princípios previamente explicados: servidores vizinhos não podem ser atualizados ao mesmo tempo e a atualização geral do sistema deve ser feita no menor número possível de etapas. Deve-se fazer tal tarefa usando um método de força bruta e outro heurístico, visando comparar e compreender as diferenças em relação a resultados e tempo de execução de cada um.

3. Desenvolvimento

O desenvolvimento do trabalho se deu em duas etapas. Na primeira, foi implementado um algoritmo heurístico que calcula o menor número possível de cores dadas a um grafo, respeitado as regras do problema. Tais cores representam as rodadas de atualização dos servidores. Inicialmente, é lido o arquivo de entrada que contém, respectivamente, as seguintes informações:

- Número de servidores
- Número de conexões entre os servidores
- Primeira conexão
- Segunda conexão
- ...
- n - ésima conexão

A primeira informação diz respeito à quantidade de servidores existentes na rede em questão. Esses servidores são os vértices do grafo trabalhado.

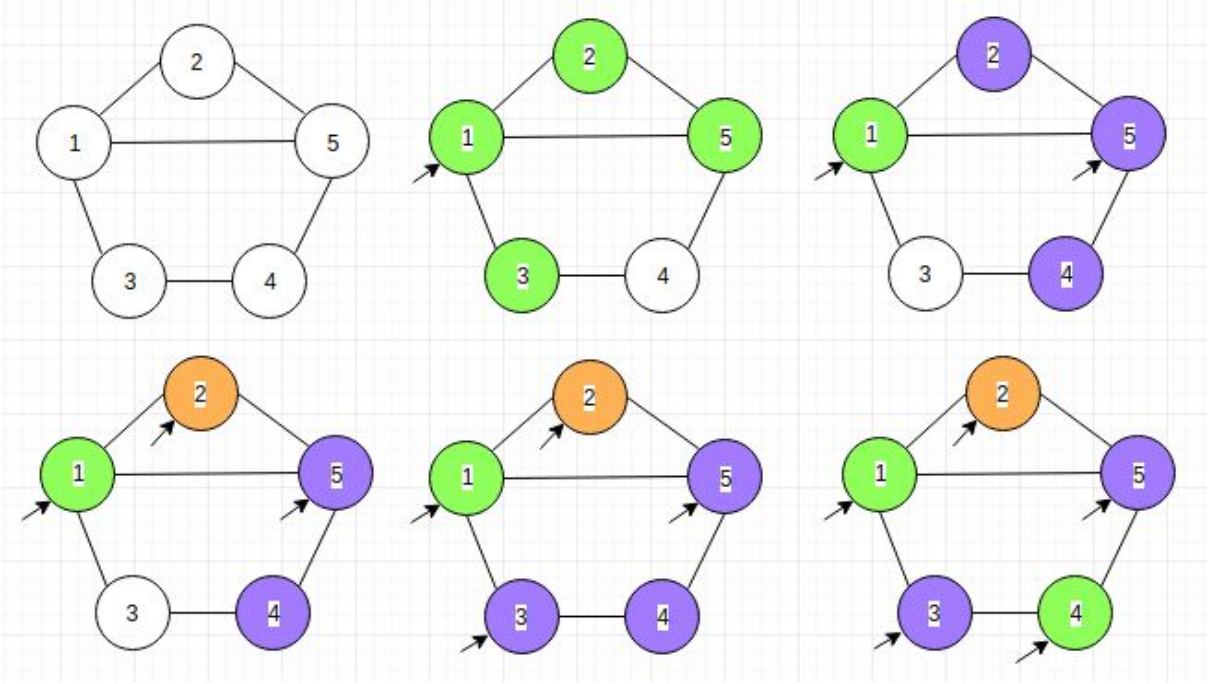
O segundo parâmetro é o número **n** de conexões do grafo, ou seja, o número de arestas, que representam a vizinhança entre dois servidores quaisquer.

Os n-ésimos parâmetros seguintes são as relações entre os servidores, representadas por um par que contém a identificação de dois vértices quaisquer.

Para a implementação, foi utilizada uma estrutura de dados chamada *Server*. Ela é composta pelos atributos *id, color, updated, pos, degree* e *next*. O primeiro é um inteiro de 1 a n que contém a identificação de um servidor; o segundo é um inteiro que representa a coloração ou não de um vértice; o terceiro, da mesma forma, mostra se um servidor já foi atualizado ou não; o quarto armazena a posição do vértice quando ele é armazenado em um vetor; *degree* representa o grau de um vértice, ou seja, o número de arestas às quais ele é ligado; a última representa um vértice vizinho.

A resolução segue o seguinte algoritmo, ilustrado pela figura 2:

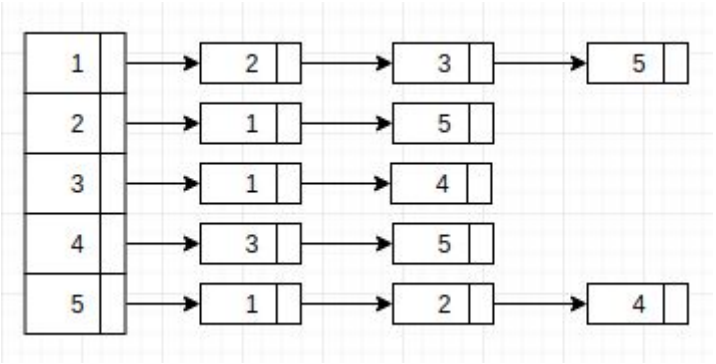
Figura 2 - Um algoritmo heurístico para coloração de grafos



Fonte: criado pela autora

Inicialmente, tem-se um grafo cujos vértices são os servidores enumerados em ordem crescente a partir de 1. Todos os seus atributos inteiros são zerados, com exceção do *id*, que é o próprio número. É montada uma lista de adjacências, conforme figura 3, que associa cada vértice aos seus vizinhos por encadeamento.

Figura 3 - Lista de adjacências



Fonte: criado pela autora

Montada a lista, a análise começa pelo vértice de maior grau em ordem crescente, ou seja, o 1. Se ele ainda não for atualizado (*updated*), tenta-se colocar a menor cor possível nele (as cores são inteiros começados em 1). Para isto, observa-se cada vizinho visando descobrir se ele já é colorido ou não com a cor 1. Se algum for, tenta-se a cor 2, e assim por diante. Definida a menor cor possível, colore-se o vértice em questão e todos os seus vizinhos não atualizados e seta-se sua flag *updated*. O processo é repetido até que todos os vértices tenham tal flag setada e as cores definidas.

4. Considerações

Foram testados todos os casos-base 3 vezes e observado, através da ferramenta "time", o tempo real gasto para cada um, conforme mostrado no gráfico 1.

Gráfico 1 - Tempo médio de execução por quantidade de vértices e arestas

Fonte: criado pela autora

No geral, a complexidade de tempo e de espaço da implementação foi O(|V| * |A|) , onde |V| é o número de vértices e |A| é o número de arestas. Essa complexidade é influenciada tanto pelo algoritmo de coloração quanto pela estruturação em lista de adjacências.

Devido à grande quantidade de trabalhos e provas neste fim de semestre, não consegui fazer, a tempo, a versão de força bruta. Compreendi o funcionamento, planejei o algoritmo, mas não tive tempo para implementá-lo.

5. Referências Bibliográficas

USP, IME. Coloração de vértices de grafos não-dirigidos. Disponível em: <https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/vertex-coloring.html>. Acesso em: 27 jun. 2018.