Autores do projeto: Paulo Ricardo Cunha (11080312); Wesley Lima de Araujo (11201721514)
Link da apresentação: https://youtu.be/QgR0o5YqCxs

Algoritmos Gulosos para Coloração em Grafos - Uma implementação funcional

1. Introdução e objetivos

O problema de otimização conhecido como Coloração de Grafos consiste em dar um rótulo para cada vértice de um grafo, com a condição de que vértices vizinhos (que possuem uma aresta entre si) não possuam o mesmo rótulo. É convenção falar que esses rótulos são “cores”. A otimização almejada nesse problema é usar a menor quantidade possível de cores. Esse problema é NP-difícil, então para resolvê-lo existem diversos algoritmos que usam abordagens distintas, como Meta-Heurísticas e Aproximações. Uma das abordagens Heurísticas mais comuns é a Gulosa. Essa abordagem consiste em construir uma solução para o problema usando decisões que são ótimas naquela etapa da construção onde a decisão está sendo tomada, sem levar em conta seus possíveis efeitos.

Esse documento é o relatório final do projeto da disciplina Paradigmas da Programação. O objetivo do projeto é implementar em Haskell dois Algoritmos Gulosos para o problema de otimização conhecido como Coloração de Grafos. A principal motivação para esse tema é que o Projeto de Graduação de um dos membros do grupo é desenvolver uma biblioteca de algoritmos para coloração de grafos usando Python. Então a vontade de comparar as implementações de alguns algoritmos é o principal motivador, especialmente porque todo referencial teórico lido só mostrava implementações em linguagens imperativas.

2. Sobre os algoritmos e suas implementações

Aqui falaremos muito brevemente sobre os algoritmos implementados e sobre as nossas implementações em Haskell deles. No nosso trabalho implementamos os algoritmos Guloso e DSatur. Ambos são heurísticas gulosas que devolvem uma solução viável para o problema. A referência usada na implementação de ambos foi a Seção 2 do livro de Lewis[1] que apresenta pseudocódigos imperativos de ambos os algoritmos. Para mais informações sobre esses algoritmos, recomendamos consultar essa mesma referência.

Vale destacar que além da implementação dos algoritmos, também fizemos uma implementação própria da estrutura Grafo como um tipo de dado, e também de algumas funções utilitárias para grafos. A nossa implementação considera que grafos são representados pela lista de suas arestas, onde Aresta também é um tipo de dado. Um exemplo de um grafo g usando nossa implementação seria `g = Grafo [Aresta ("a","b"), Aresta ("b","d"), Aresta ("b","e"), Aresta ("c","e"), Aresta ("d","e")]`. Os elementos da tupla da aresta são os vértices. Para obter uma lista de vértices desse grafo basta usar uma outra função implementada por nós, a `vertices g`. Mais informações sobre as implementações se encontram em comentários do próprio código.

[1] R. R. Lewis. A Guide to Graph Colouring: Algorithms and Applications.

O algoritmo Guloso recebe como entrada um grafo e uma sequência dos vértices desse grafo. O Guloso vai colorindo os vértices desse grafo seguindo a sequência passada na entrada. Para chamar a implementação desse algoritmo use a função `guloso` passando como parâmetros um grafo e uma lista dos vértices desse grafo, sendo que essa lista é a sequência em que os vértices serão coloridos. Já o DSatur (Degree of Saturation) funciona de uma maneira diferente. Pois a ordem em que os vértices do Guloso são coloridos é algo pré-definido na chamada do algoritmo, já no DSatur o próximo vértice a ser colorido depende da métrica grau de saturação que é a quantidade de vértices vizinhos coloridos naquele momento da construção. Para chamar o DSatur use a função `dsatur` com os parâmetros sendo o grafo e o vértice inicial a ser colorido, que é necessário para começar a calcular a métrica.

3. Uso do projeto

Segue um exemplo de uso dos algoritmos implementados.

1. Dentro do diretório onde está o arquivo Main.hs inicie o ghci: `stack ghci`
2. Construa um grafo usando a estrutura implementada: `g = Grafo [Aresta ("a","b"), Aresta ("b","d"), Aresta ("b","e"), Aresta ("c","e"), Aresta ("d","e")]`
3. Para usar o Guloso passe o grafo e uma sequência dos vértices: `guloso g ["a","b","c","d","e"]`
4. A função retornará uma lista de inteiros, sendo que cada inteiro é a cor do vértice na respectiva posição na sequência passada na chamada `[0,1,0,0,2]`
5. Para usar o DSatur passe o grafo e o vértice inicial da coloração: `dsatur g "a"`
6. O DSatur também devolve uma lista de inteiros com mesmo significado que o guloso: `[0,1,0,2,0]`
7. Para saber qual a ordem de vértices associada à coloração devolvida pelo dsatur use `vertices g` que nesse exemplo devolve `["a","b","d","e","c"]`

4. Considerações finais

Implementar um algoritmo sobre grafos usando programação funcional exigiu o uso intenso de muitas funções recursivas sobre estruturas de dados. O próprio ato de colorir era recursivo, o que envolvia a decisão de qual cor atribuir aos vértices. Porém, mesmo sendo uma experiência nova para os membros do grupo, o fato das estruturas serem persistentes facilitou as implementações, mas isso exigiu persistir as informações em cada nova chamada como parâmetros das funções. Em geral, o propósito do trabalho foi atendido, pois o grupo implementou os algoritmos usando o paradigma funcional. A opinião final do grupo é que a implementação funcional feita foi interessante, mas ela é pouco compreensível em comparação com a implementação imperativa já realizada.