Trabalho Prático 1

O Plano de Campanha

João Vitor Ferreira

Universidade Federal de Minas Gerais (UFMG)

Belo Horizonte — MG — Brasil

joaovitorferreira@ufmg.br

1. Introdução

O problema proposto foi de implementar um código utilizando-se de grafos e estruturas de dados com o intuito de solucionar o problema de satisfazibilidade de grafos. Para isso foi realizado o uso de um algoritmo de solução conhecido como K_sat, e para este trabalho foi utilizado uma de suas variantes, o 2-sat. Mais especificamente o, Algoritmos de Kosaraju.

2. Método

O programa foi desenvolvido na linguagem C++/C, compilada pelo compilador G++ da GNU Compiler Collection. E para sua execução usa-se o Makefile que está junto do arquivo-fonte que compila e executa o programa através do comando make, bastando apenas adicionar com o nome do arquivo contendo as entradas.

Ex: make < in.txt

2.1. Estrutura de Dados e Algoritmos.

Como estruturas de dados e algoritmos foram utilizados.

a) TAD Grado

Uma classe que contém os atributos e funções necessárias para o preenchimento do grafo, tais como as estruturas padrão para o preenchimento do grafo, como os vetores para as arestas padrão e inversas, um operador booleano para demarcar as arestas visitadas, e os elementos e as variáveis globais S (# de seguidores) e P (# de propostas). Já as funções implementadas foram para realizar a adição da aresta comum e inversa, DFS nas arestas comum e inversas, preencher as arestas, verificar o tipo da aresta, a presença de ciclo, e a inserção da aresta pelo tipo de entrada. Outras funções eram a de iniciar e retornar os valores de S e P, verificar o tipo de proposta, limpar os dados das estruturas antes de um grafo novo e por fim o método K_sat que performa o algoritmo de Kosaraju.

b) Algoritmo 2-sat

Para poder realizar a verificação da satisfazibilidade do grafo, o algoritmo K_sat foi implementado e alterado para poder atender aos requisitos proposto inicialmente com este trabalho.
O algoritmo inicialmente preenche as arestas e verifica o tipo inserindo conforme o tipo encontrado, após isso ele realiza a primeira DFS que insere os elementos numa lista até que todos os elementos sejam visitados. Após isso é realizado uma DFS nas arestas inversas pela lista de elementos e realiza a contabilização das componentes conexas. Por fim ele realiza a verificação da presença ou não de ciclo, ou seja, verifica se na mesma componente conexa está presente o elemento e seu complemento, logo, se na mesma componente conexa há o elemento x e ~x. Realizando esse mesmo algoritmo para todos os grafos na entrada.

Pseudocódigo

K_sat(Propostas_1, Propostas_2)

Preenche e insere as propostas nas arestas do grafo.

Para elemento não visitado até 2*P
    Realiza a DFS

Para cada elemento da lista de elementos inversa se ele não foi visitado
    Realiza a DFSi

Verifica se há um ciclo no grafo.