Trabalho Prático 1

Gabriel Franco Jallais – Matrícula: 2021031890

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

Belo Horizonte – MG – Brasil

gfjallais@ufmg.br

Entrega: 11/10/2022

1. Introdução

O problema proposto para este trabalho foi ajudar o deputado federal Alan Turing a melhorar seu plano de campanha e agradar seus apoiadores. Após uma votação no seu Instagram o deputado gostaria de obter o melhor conjunto de propostas que agradasse todos os seus eleitores, o critério foi o seguinte: cada eleitor votaria em duas propostas para serem mantidas no plano e duas para serem removidas, o eleitor seria satisfeito caso uma das duas propostas que ele votou para ser mantida, realmente fosse mantida, e uma das que ele votou para ser removida fosse removida.

Para isso ele requisitou a nossa ajuda, para elaborar um algoritmo que fosse capaz de dizer se é possível satisfazer todos os eleitores com base no resultado da votação.

2. Modelagem

2.1 Abordagem do problema

Uma possível solução para o problema seria modelá-lo como sendo um problema de satisfatibilidade de uma expressão booleana, a conjunção de diversas clausulas, onde cada uma fosse formada pela disjunção de dois literais. Foi a abordagem que tomei.

2.2 Modelagem do grafo

Como exigido na especificação do trabalho, foi utilizado um grafo para modelar o problema, e, como dito acima, foi utilizada a abordagem do problema de 2-SAT, com isso, o grafo foi modelado da seguinte forma:

Foi utilizada uma lista de adjacência para representar o grafo no programa, ela foi aplicada por meio de um vetor de vectors, da biblioteca <vector> da linguagem C++, visto que é uma forma eficiente e segura de alocar dinamicamente os vértices e arestas do grafo, visto que é uma estrutura já otimizada para isso e que não há o risco de mal manejo de ponteiros e memória por minha parte.

Foi reservado na memória duas vezes o número de propostas para o vetor de vetores, para representar os literais da expressão booleana e suas negações. Os n primeiros índices do vetor de vetores, onde n é o número de propostas passado na entrada, representam as propostas, e os n seguintes os valores negados.

Cada vetor de cada índice i armazena o índice das arestas que saem de i e incidem em outro vértice v.

Para cada cláusula (p + q), há uma aresta de ~p para q e de ~q para p, pois, caso um dos laterais seja falso, para que o problema seja satisfeito, o outro deve ser verdadeiro.

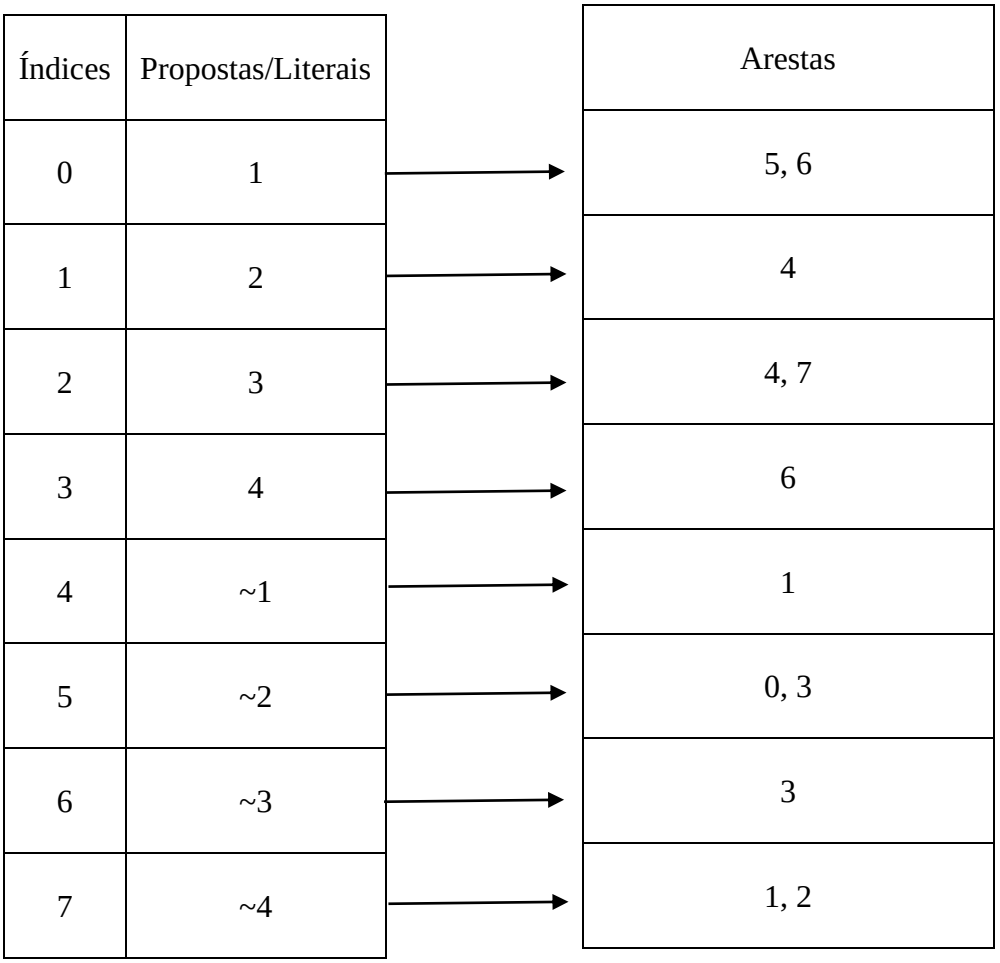
Exemplo:

Entrada: 3 4

1 2 3 4

3 4 1 2

2 3 1 4



Índices	Propostas/Literais	Arestas
0	1	5, 6
1	2	4
2	3	4, 7
3	4	6
4	~1	1
5	~2	0, 3
6	~3	3
7	~4	1, 2

Caso dos 0s:

Assim como especificado no enunciado do trabalho se algum eleitor fornecer o valor 0 na entrada, significa que esse voto não foi utilizado, o que implica que, caso o outro voto, para manter ou remover, seja não nulo, todos os demais vértices devem apontar para o vértice correspondente ao valor deste segundo voto, visto que ele necessariamente será verdadeiro, para que todos os seguidores sejam satisfeitos.

Caso ambos sejam nulos, não é necessário adicionar nenhuma aresta, visto que não será necessário agradar esse seguidor em relação aos votos para manter ou remover propostas, dependendo de qual tenha sido a categoria em que os dois votos foram 0.

2.3 Algoritmo utilizado

O algoritmo utilizado consiste em encontrar os componentes fortemente conexos do grafo e caso um literal x esteja no mesmo componente fortemente conexo que sua negação, ~x, então o problema não será satisfeito, visto que x não pode ser verdadeiro e falso ao mesmo tempo. O problema será satisfeito caso contrário.

Para encontrar tais componentes foi utilizado o algoritmo de Kosaraju, que consiste em realizar, primeiramente, uma busca por profundidade no grafo formado, empilhando os vértices que forem finalizados, em uma pilha auxiliar, que também foi implementada com um vector.

Em seguida, são feitas outras buscas por profundidade, agora, porém, num grafo com os mesmos vértice e arestas do grafo original, mas com as arestas na direção oposta, as DFS são feitas a partir dos vértices no topo da pilha, e efetuada apenas nos vértices ainda não visitados.

Nessa segunda etapa, de buscas por profundidade, em cada busca é atribuído um valor aos vértices visitados, esse valor é armazenado num vector auxiliar. Assim basta checar, nesse vetor auxiliar, que segue o mesmo princípio de indexação que o vetor utilizado para representar o grafo - os n primeiros índices, onde n é o número de propostas passado na entrada, representam as propostas, e os n seguintes os valores negados - se x possui o mesmo valor que ~x.

Por meio deste método é possível dizer se o problema é satisfazível ou não.

Em geral foi utilizado vectors para representar as estruturas e foram criadas duas funções para realizar as DFSs no grafo original e outra no grafo com as arestas na direção oposta.