Algoritmos 1 – Prof. Vinicius Fernandes

Trabalho Prático 1

1. **Apresentação**

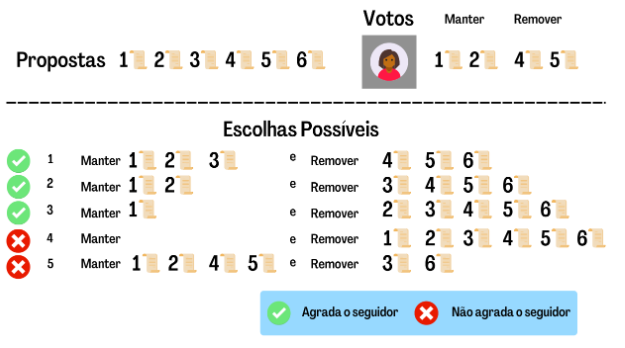
**Nome**: Mateus Henrique Souza Silva

**Matrícula**: 2020006841

1. **Introdução**

Durante uma eleição, um candidato elaborou algumas propostas para seu plano de governo e decidiu apresenta-las ao público. Afim de obter maior aceite popular, o candidato abriu uma votação para que seus eleitores pudessem votar nas propostas as quais eles gostariam que se mantivessem e as quais eles gostariam que fossem removidas do plano de governo. Como apenas 4 propostas puderam ser votadas, foi decidido que os eleitores votariam em 2 propostas que fossem a favor da permanência e 2 que fossem contra.

Após a votação, o candidato gostaria de saber se existe um conjunto de propostas, tal que agradasse a todos os seus eleitores. Para agradar a um eleitor, é necessário que ao menos uma das propostas votadas para permanecer, fiquem no plano de governo e ao menos uma votada contra, seja retirada do mesmo.



Como exemplo, podemos ver na imagem acima, possíveis soluções para o eleitor em questão. Ao manter as propostas 1, 2 e 3 e remover as propostas 4, 5 e 6, o eleitor é satisfeito, pois as propostas 1 e 2 foram mantidas e as propostas 4 e 5 foram retiradas, assim como havia votado o eleitor.

1. **Modelagem**

A primeira parte do problema acima, consiste em mapear os votos de forma que possamos utilizar algum algoritmo computacional para resolvê-lo. Na implementação adotada nesse trabalho, os votos foram mapeados com álgebra booleana, ou seja, os votos para manter uma proposta são uma entrada T (True) e os votos para a remoção de uma proposta são uma entrada F (False). Portanto, um grupo de seguidores, pode ser expressado como uma fórmula booleana.

**Exemplo**:

Seguidor 1:

Manter: 1 e 5

Remover: 2 e 4

Expressão booleana resultante: (1 v 5) ∧ (¬2 v ¬4)

Tendo em vista a modelagem acima, é possível definir se um conjunto de propostas satisfaz ou não todos os eleitores através do uso de componentes fortemente conexas em grafos.

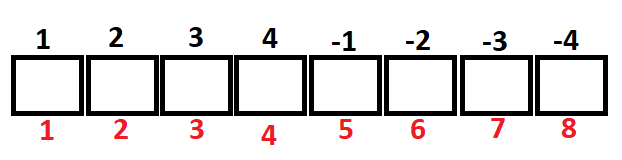
Na implementação usada neste trabalho, foi utilizado um grafo definido com listas de adjacências para representar as arestas.

Inicialmente, precisamos inserir os vértices, que representam os votos, no grafo de forma que possamos mapear os votos a favor e contra dos eleitores. Para isso, foi definido que os votos contrários às propostas fossem representados por números negativos e os votos favoráveis por números positivos.

Já para inserção na estrutura de dados escolhida, não podemos utilizar número negativos, pois somente conseguimos acessar posições **n**, tal que **n >= 0**. Para solucionar esse problema, os votos foram mapeados como **x = x** e **-x = m-(-x)**, onde m representa o número de propostas passíveis de votação. Dessa forma, os vértices do grafo vão de **1** a **2m**, representando assim, todas os possíveis votos para todas as propostas:

**Exemplo:**

**m = 4**



Após a inserção de todos os votos no grafo, é rodada uma busca em profundidade (**BFS**) em cada um dos vértices do grafo, afim de visitar todos os vértices e inseri-los na pilha de ordenação. Feita a primeira BFS, é utilizado o algoritmo de Kosaraju para encontrar as componentes fortemente conexas do grafo.

O algoritmo de **Kosaraju** consiste em, dado um grafo original **G**, inverte a direção das arestas de **G**, dando origem a um novo grafo **Gr**, onde **Gr** é o grafo reverso de **G**. Posteriormente, faz uma BFS em **Gr** e calcula a permutação de pós ordem dos vértices de **Gr.**

Feito isso, o algoritmo faz outra **BFS** em **G**, partindo da ordem gerada pela permutação de pós ordem de **Gr**, e a cada etapa da **BFS**, os vértices descobertos nela, fazem parte da mesma componente fortemente conexa.

Por fim, para verificar se uma expressão booleana é satisfazível ou não, basta verificar se os vértices **x** e ¬**x** pertencem a mesma componente. Isso se deve ao fato de que, se um vértice **x** deve ser True em uma expressão, é impossível que ele possa ser False ao mesmo tempo, o que nos leva a uma contradição.