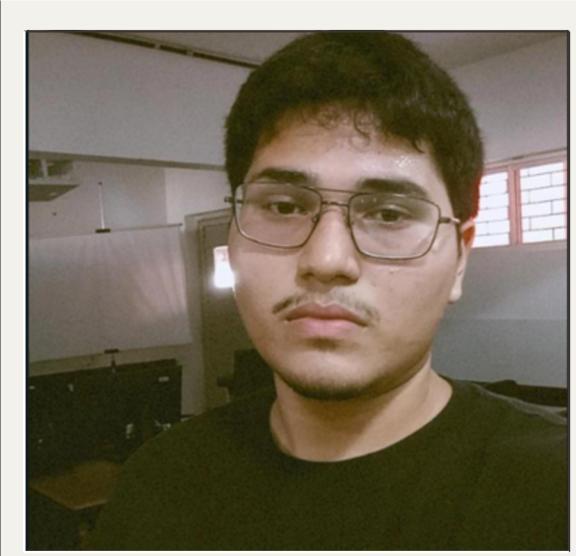


{ ED II }

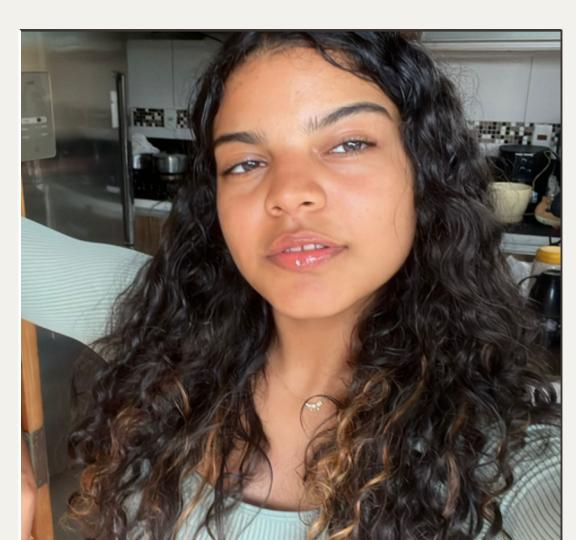
Grafos Coloridos



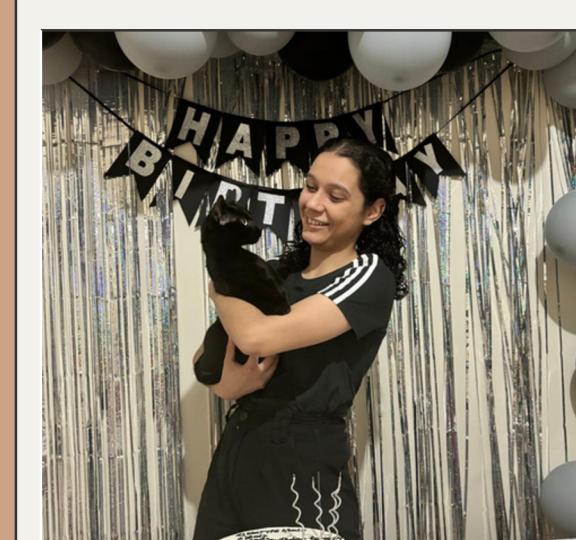
Integrantes



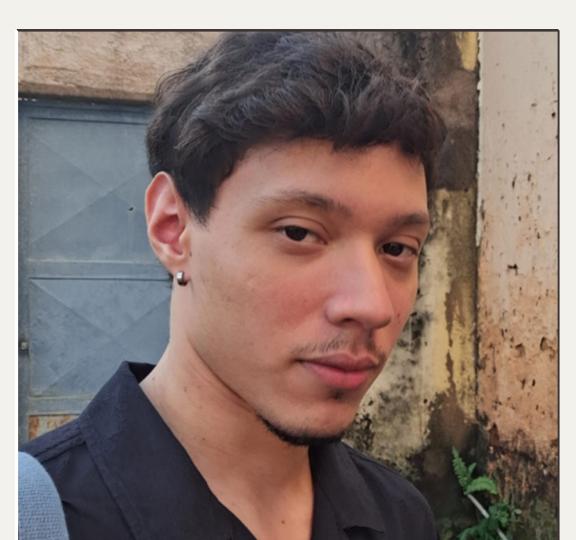
Marinaldo



Maria Vitória



Rafaela



Irving



Grafos

Leonhard Euler , em 1736, se questionava das pontes da cidade de Königsberg sera eram possivel atravessar cada uma das pontes, apenas uma vez e retornar ao ponto de partida. A partir desses estudos iniciou sobre teoria dos grafos , entre diversos conceitos esta o **grafos colorido**.

Os **grafos** são objetos matemáticos formados por vértices e arestas que conectam esse vértices. A posição e o comprimento das arestas são irrelevantes, fazendo com que o mesmo grafo seja representado de varias maneiras distintas. **O que importa e as relações que existe entre os vértices.**





Grafos Coloridos

Guthrie ganhou notoriedade por ter proposto, em 1852 conhecido como **Teorema das quatro cores**.

A coloração de grafos consiste na atribuição de cores aos elementos de um grafo, **sujeita a certas restrições**. Na sua forma mais comum, trata-se da coloração dos vértices, em que nenhuma dupla de vértices adjacentes pode receber a mesma cor é **chamado de coloração de vértices**.

De forma similar, a coloração de arestas consiste em atribuir uma cor a cada aresta, garantindo que **arestas adjacentes não compartilhem a mesma cor**. Já na coloração de faces de um grafo planar, cada face (ou região) recebe uma cor de modo que duas faces que compartilham um limite não tenham a mesma cor.

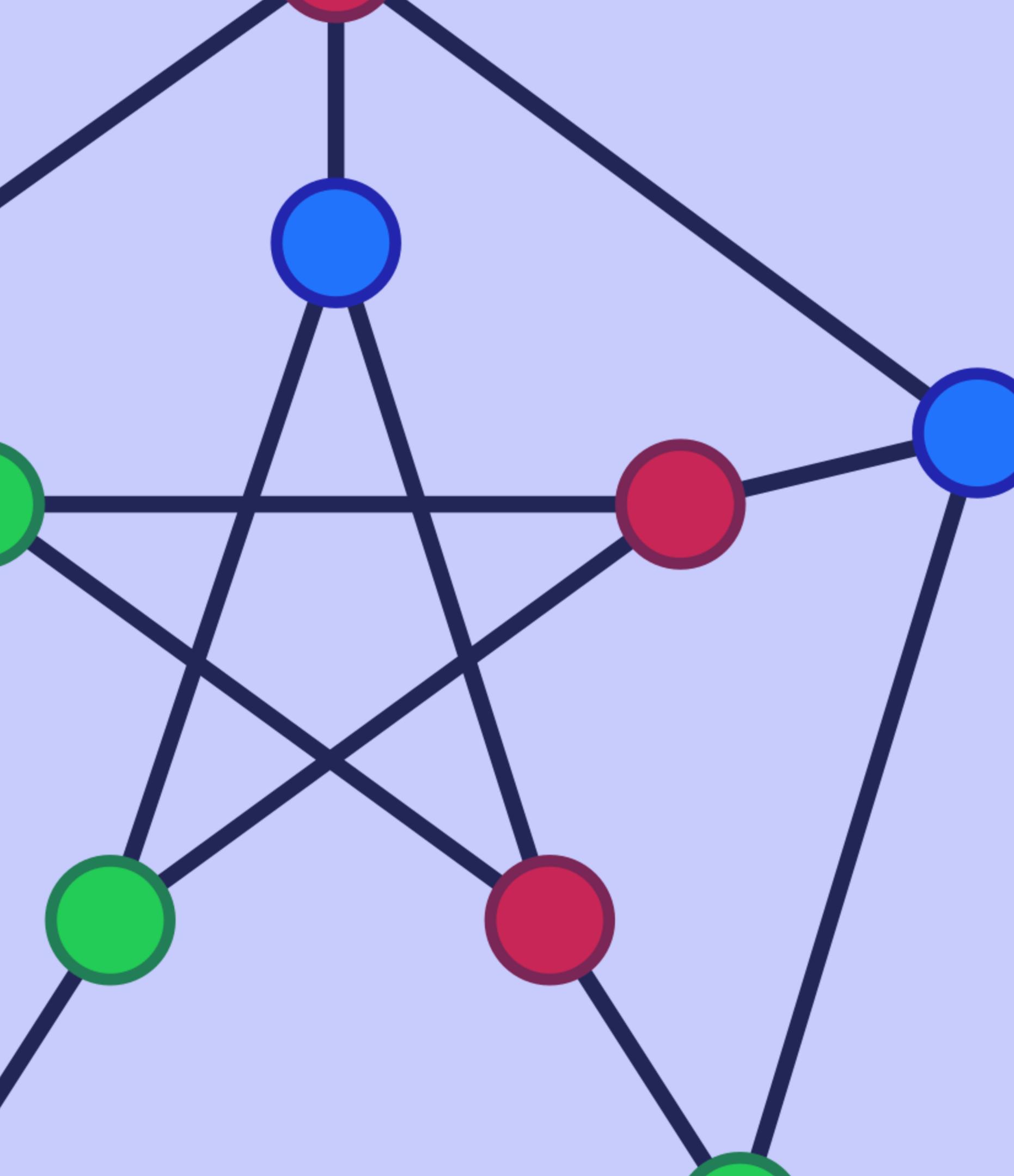


Objetivo

O trabalho tem por objetivo apresentar um sistema que possa ser eficiente em atribuir frequências de rádio a transmissores distribuídos geograficamente de forma que evite interferências entre transmissores próximos .

Os transmissores nesse problema são visualizados como vértices de um grafo, enquanto as possíveis interferências são enxergadas como arestas entre os vértices.



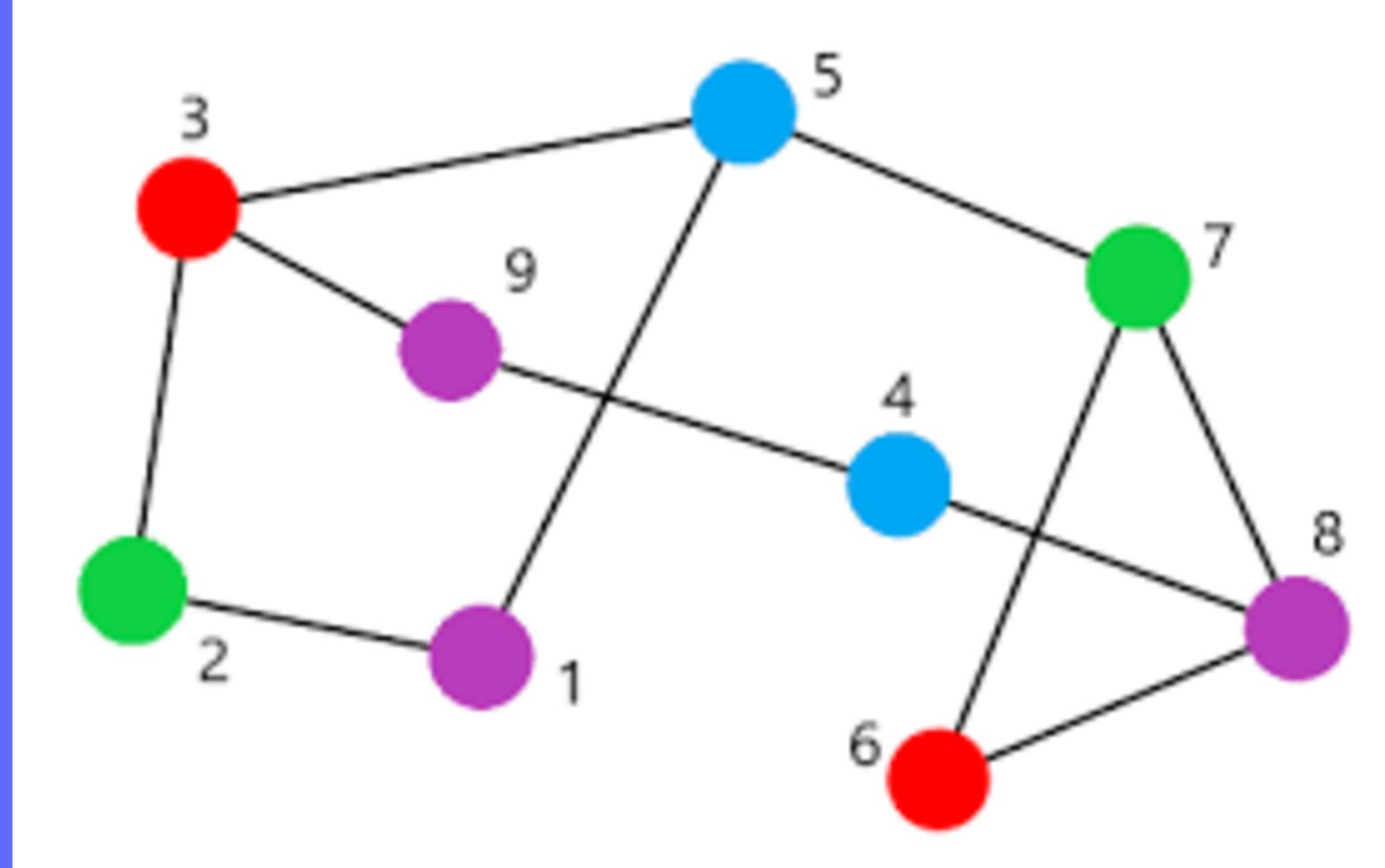


O problema

O problema consiste em colorir o grafo de forma que as frequências(cores) atribuídas a um determinado vértice, seja diferente de suas arestas adjacentes garantindo assim a coexistência harmônica.

Para a solução computacional, utilizou-se o algoritmo DSATUR(Degree of Saturation), que se mostra eficiente na resolução desse tipo de problemática.

Metodologia



"Vértices com mais vizinhos coloridos são priorizados, recebendo a menor cor disponível."

Neste trabalho, adotamos uma abordagem computacional baseada no algoritmo heurístico DSATUR, que utiliza o grau de saturação dos vértices como critério principal para coloração. O algoritmo foi implementado em linguagem C, com suporte à leitura de grafos no formato .col, fornecidos pela base DIMACS, padrão em pesquisas sobre coloração de grafos. A representação do grafo foi feita por meio de listas de adjacência, permitindo maior flexibilidade e economia de memória.

Durante a execução, o vértice com maior saturação é escolhido a cada etapa, garantindo que os vértices mais restritos sejam coloridos primeiro, o que contribui para uma coloração mais eficiente.

O critério de desempate considera o grau do vértice e, em último caso, o índice. A ordenação dos vértices foi feita com qsort.

Aplicação da Metodologia

Utilizamos instâncias da base **DIMACS em formato .col**, amplamente usada em testes com grafos. A instância analisada, **inithx.i.1.col**, **possui 864 vértices e 18.707 arestas.**

O sistema **foi implementado em C**, com grafos representados por listas de adjacência. A leitura dos arquivos constrói a estrutura do grafo, e o algoritmo DSATUR é aplicado, **atribuindo cores aos vértices com base no grau de saturação.**

Ao final, o sistema imprime no terminal a cor de cada vértice, o total de cores usadas e o tempo de execução.



```
int menorCor = 0;
while (ehColorido[menorCor])
    menorCor++;

cor[u] = menorCor;
```

```
    allsquare[x][y].currentPiece = willMove.piece;
    allsquare[x][y].pieceColor = willMove.color;
    allsquare[willMove.curX][willMove.curY].currentPiece = ...
    allsquare[willMove.curX][willMove.curY].pieceColor = ...
    willMove.ready = false;
    this->getCo...
```

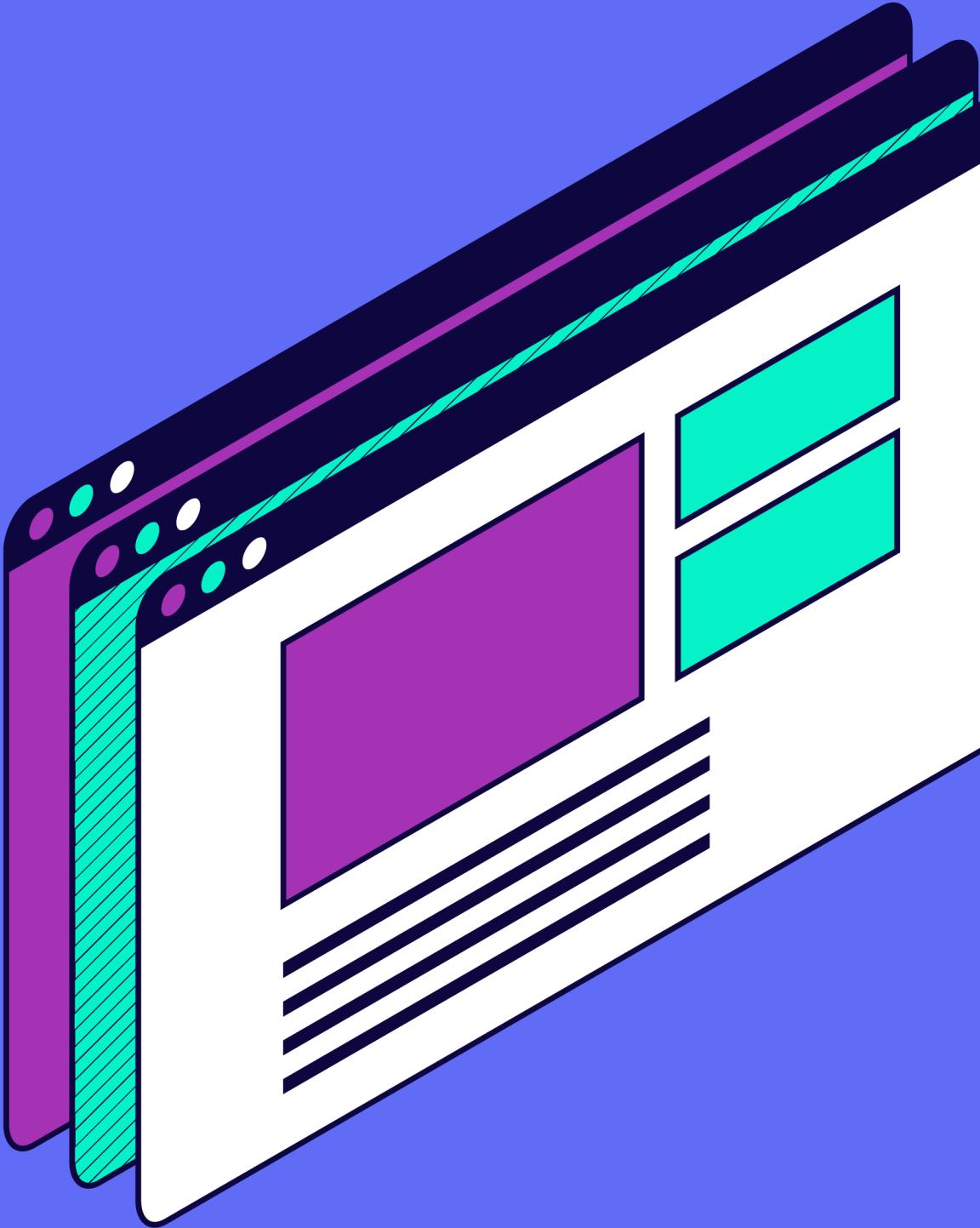
Avaliação Experimental

Vantagens

- Número de cores **próximo da coloração mínima**.
- Supera heurísticas simples como Welsh-Powell e Grau Máximo Primeiro.
- Usa o grau de saturação para **se adaptar dinamicamente** à configuração do grafo .
- Mais sofisticado que heurísticas ingênuas, mas ainda simples de programar, pois **não exige estruturas muito complexas**.
- Bom desempenho em diferentes tipos de grafos, sendo útil em problemas práticos como **alocação de frequências**.

Desvantagens

- Não assegura o uso do **número cromático mínimo**.
- Custo total da ordenação: **$O(n^2)$** com qsort e **$O(n \log n)$** com heap (mais eficiente, mas mais complexo na implementação).
- Sensibilidade a empates: empates em grau de saturação e grau **são resolvidos com base no índice**.
 - A coloração final pode depender da ordem inicial dos vértices.





Conclusão

- Conclui-se que o algoritmo DSATUR desempenhou sua funcionalidade de maneira precisa mas não perfeita, chegando próximo do número cromático obtido por outros algoritmos mais precisos como o backtracking que possui grau de complexidade $O(b^d)$, enquanto o DSATUR $O(N^*N)$

OBRIGADA PELA ATENÇÃO

