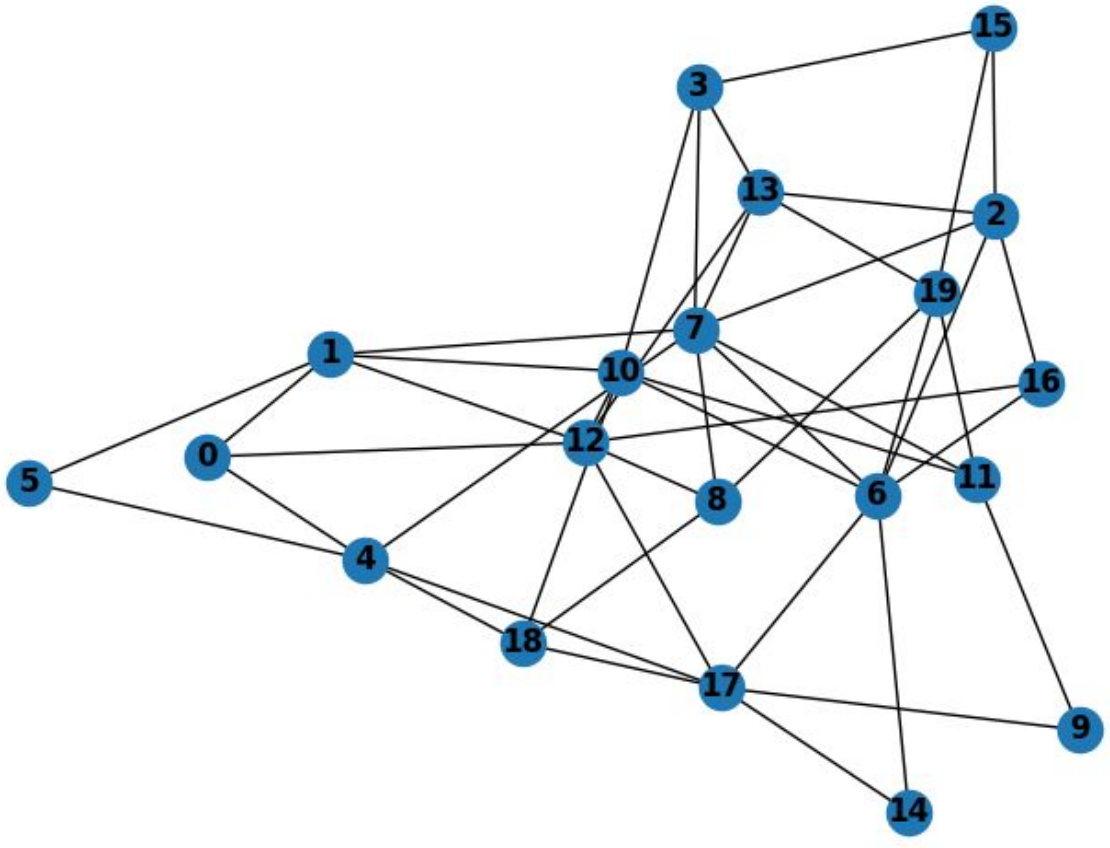
Universidade Federal de São João del-Rei
Grafos
Aula Prática - AGM
Felipe Francisco Rios de Melo

Este relatório está dividido em quatro tópicos: visualização de um grafo gerado aleatoriamente, visualização do grafo após a execução dos dois algoritmos omitindo arestas ausentes, visualização do grafo executado colorindo as arestas ausentes e por fim, uma rápida análise e discussão do tempo de execução dos algoritmos de Kruskal e Prim.

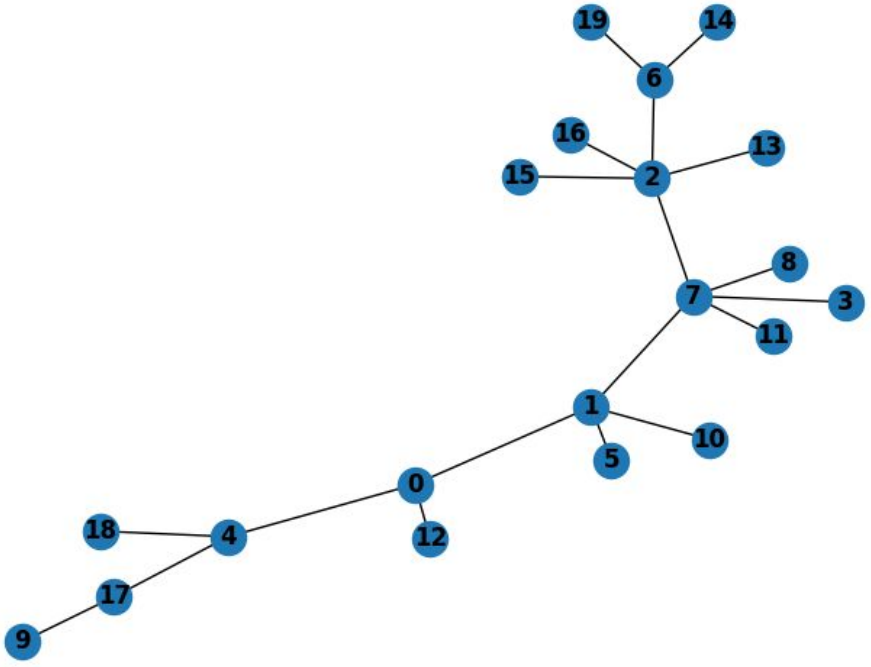
1. Visualização de um grafo gerado aleatoriamente com as bibliotecas *networkx* e *matplotlib:*



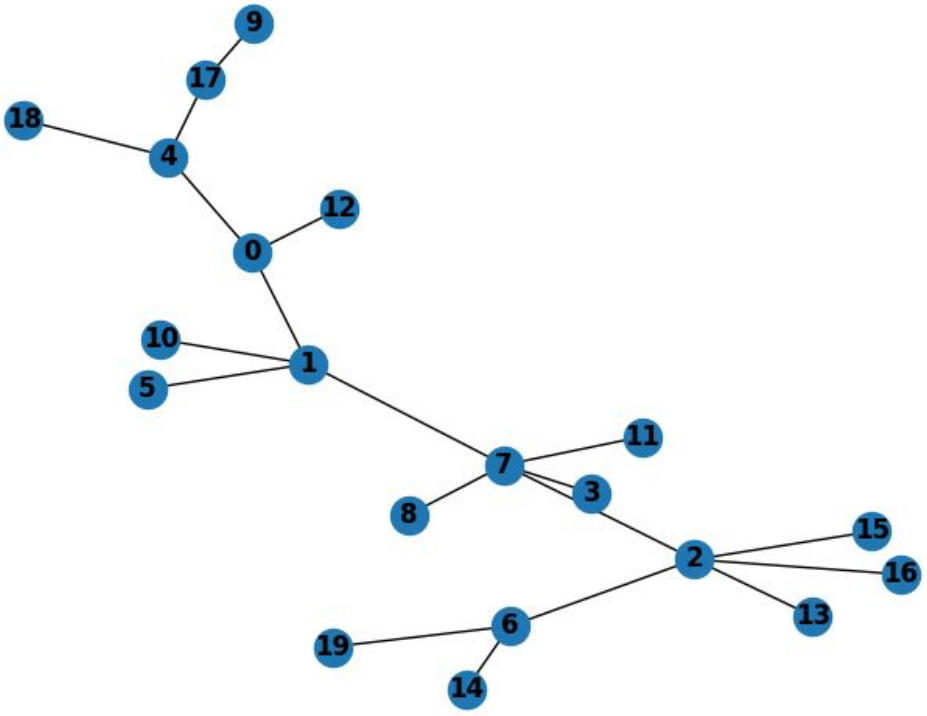
OBS: A visualização dos grafos alteram a disposição de vértices e arestas a cada plotagem, mas mantém suas propriedades de conexão.

2. Visualização do grafo após execução do Algoritmo de Kruskal e Prim, omitindo as arestas ausentes:

2.1. Algoritmo de Kruskal

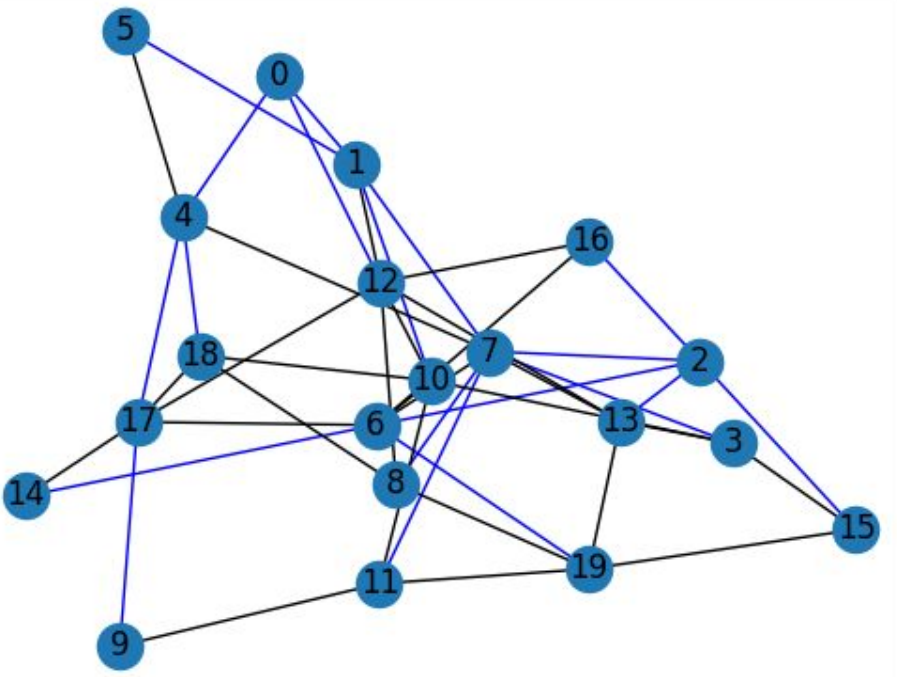


2.2. Algoritmo de Prim

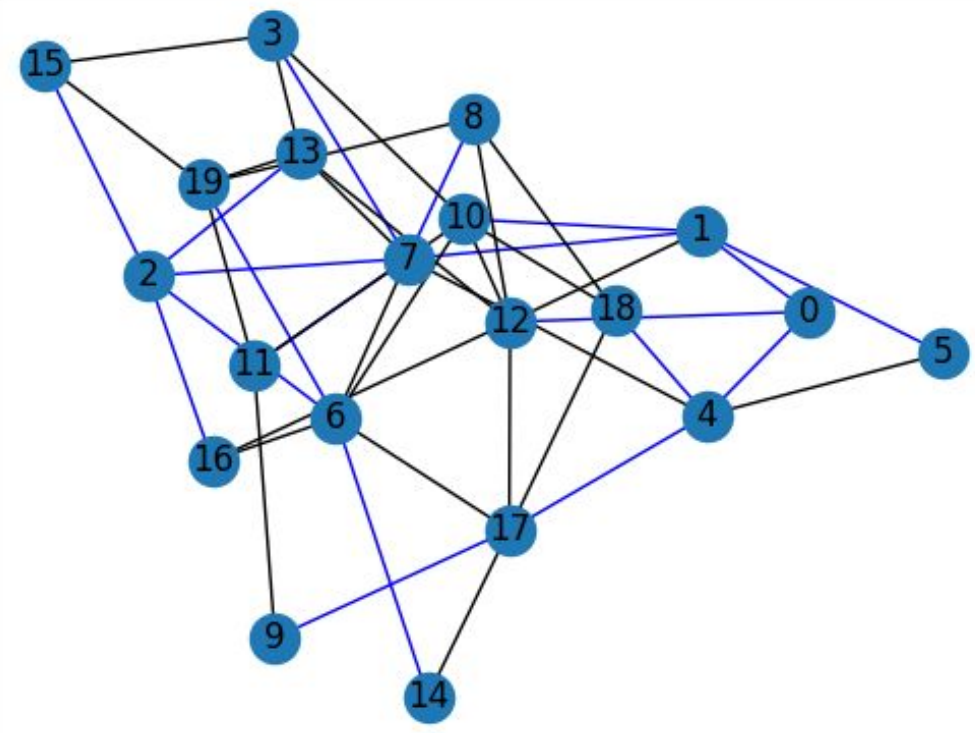


3. Visualização do grafo após execução do Algoritmo de Kruskal e Prim, colorindo as arestas ausentes de azul:

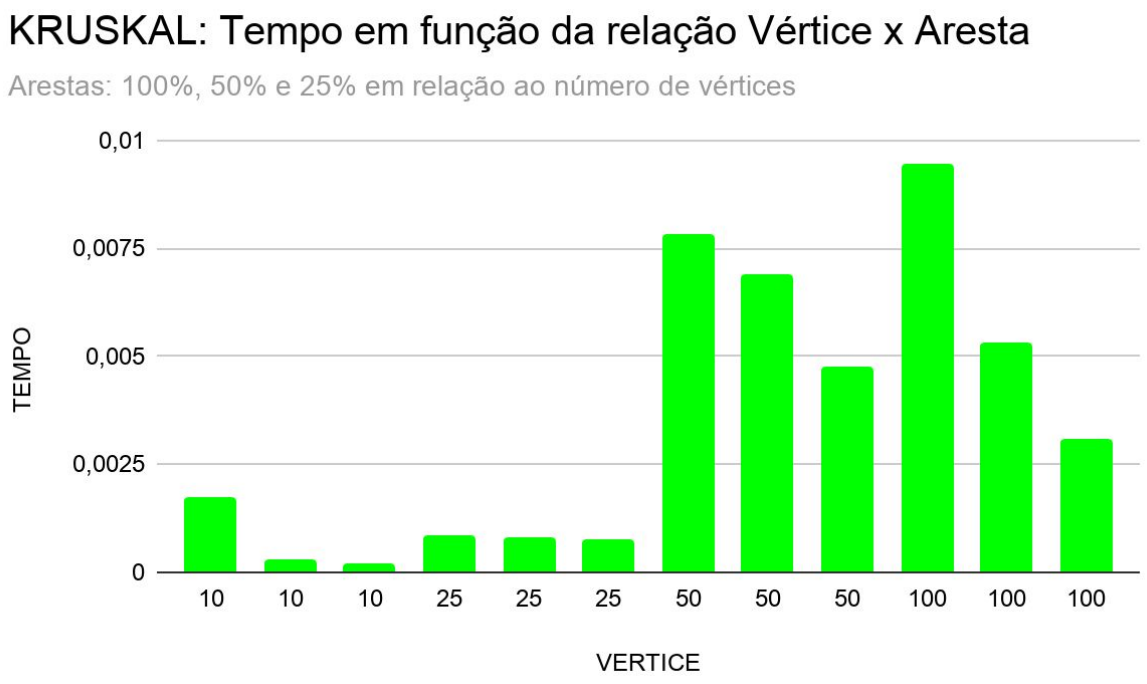
3.1. Algoritmo de Kruskal

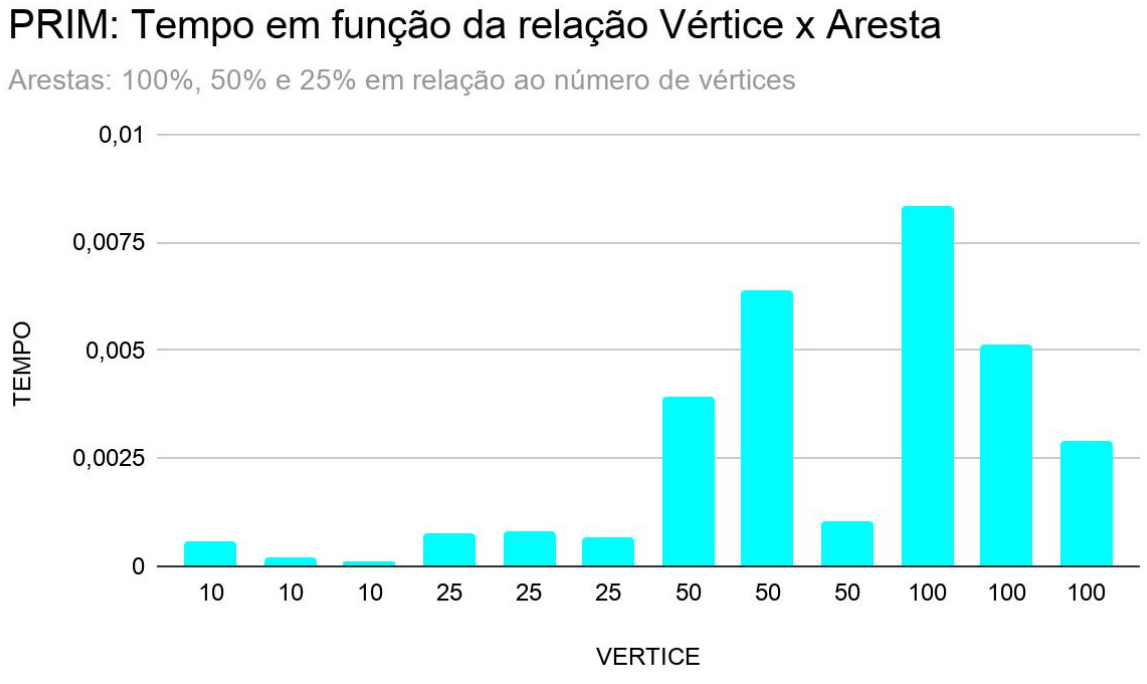


3.2. Algoritmo de Prim



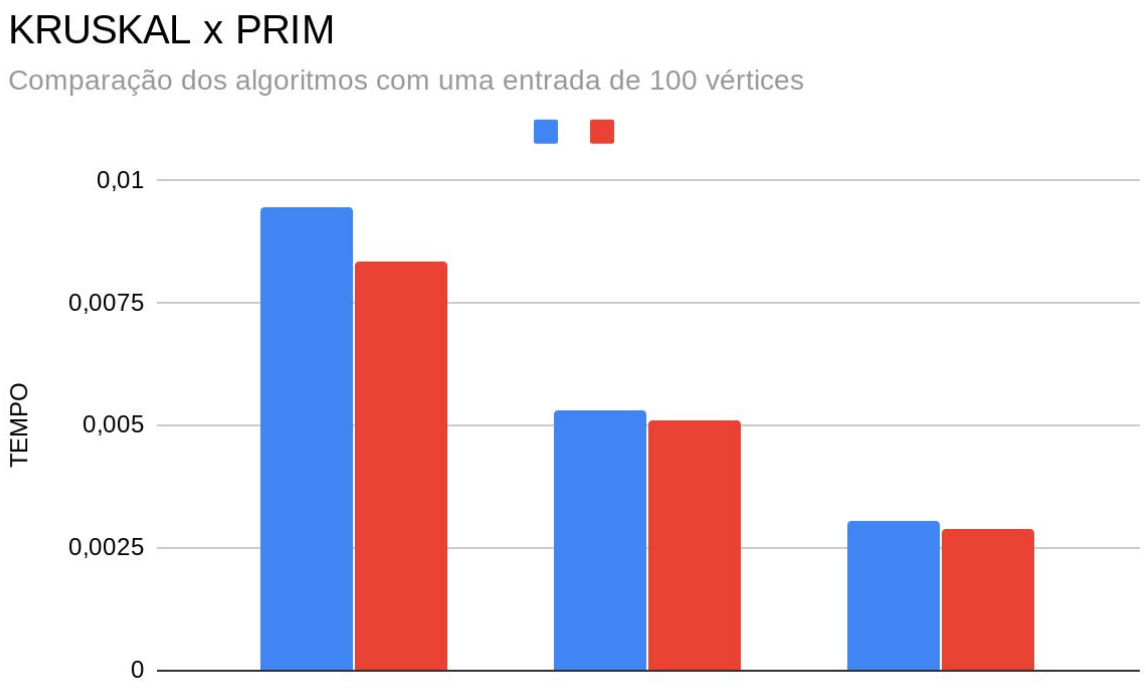
4. Em seguida foi gerado grafos aleatórios variando a quantidade de vértices e arestas. Por exemplo: para um grafo de 10 vértices, foi executado com 45 arestas (denso, 100%), 22 arestas (50%) e 11 (esparso, 25%). Fazendo esse processo para grafos de 10, 25, 50 e 100 vértices e marcando o tempo de execução, para os dois algoritmos, foi obtido os seguintes gráficos:





A complexidade do algoritmo de Prim em implementações mais comuns para um grafo são por listas de adjacência e por matrizes de adjacência e suas respectivas complexidades O(|A| log |V|) e O(V²) no pior caso, já em implementações mais complexas, utilizando uma heap de fibonacci, o algoritmo pode ser executado em O(A + |V| log |V|). E o algoritmo de Kruskal possui complexidade de tempo igual a O(m log n), onde m representa o número de arestas e n o número de vértices.

Fazendo uma comparação direta dos dois algoritmos (Kruskal x Prim), temos o seguinte gráfico:



Com o auxílio do gráfico, é possível notar que algoritmo de Prim é significativamente mais rápido no limite quando você tem um gráfico realmente denso com muitas mais arestas do que vértices. Já em situações típicas (grafos esparsos), Kruskal se iguala ao Prim, porque usa estruturas de dados mais simples.