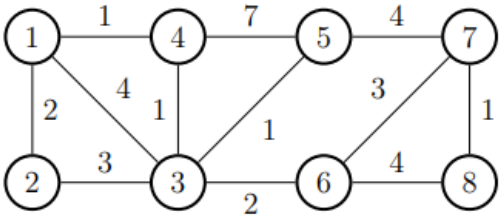
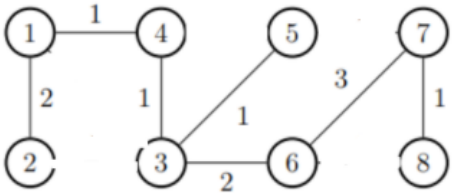
Exercício 1. *Considere o grafo abaixo, apresente uma árvore geradora de custo mínimo.*



ALGORITMO DE KRUSKAL: Complexidade *O(E log E)*
Passo #1) Ordenar arestas pelo peso:
Arestas de peso 1: (1,4),(3,4),(3,5),(7,8)
Arestas de peso 2: (1,2),(3,6)
Arestas de peso 3: (2,3),(6,7)
Arestas de peso 4: (1,3),(5,7),(6,8)
Arestas de peso 7: (4,5)

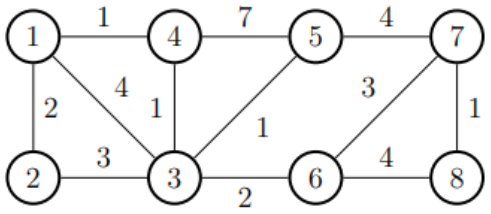
Passo #2) Colocar a menor aresta à árvore, observando se não cria ciclos
Posso adicionar (1,4),(3,4),(3,5),(7,8) (Peso 1 cada aresta, totalizando custo 4);
Posso adicionar (1,2) e (3,6) (Peso 2 cada, totalizando custo total 4+4=8);
Não posso adicionar **(2,3)** porque fecha um ciclo. Posso adicionar (6,7), custo total igual a 8+3=11;
Não posso adicionar **(1,3)** porque também fecha um ciclo com (1,2) e (2,3). Também **não** posso fechar ciclo entre **(6,8)** porque fecha ciclo com (7,8) e (6,7). Idem para **(5,7)**, que fecha ciclo com (3,5),(3,6),(6,7).
Não posso adicionar **(4,5)** por que fecharia um ciclo com (3,4) e (3,5).

Passo #3) Fazer o passo 2 até que TODOS os vértices estejam conectados



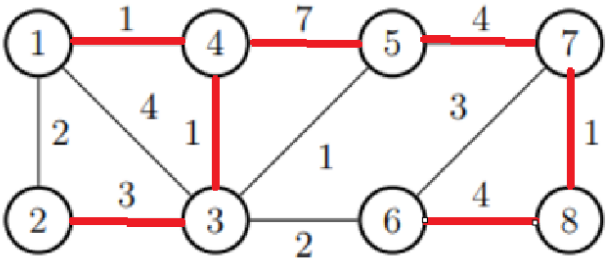
Custo mínimo = 11

Exercício 2. *Considere o grafo abaixo, apresente uma árvore geradora de custo mínimo obtida pelo algoritmo de Prim.*



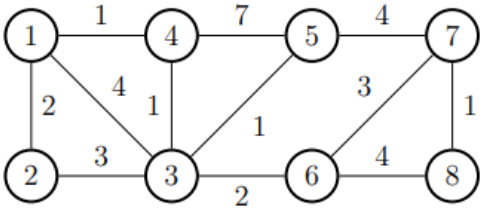
REVISAAAAR!
ALGORITMO DE PRIM: Complexidade *O(E + V log V)*
Passo #1) Escolha um vértice inicial: **1**
Passo #2) Usar uma fila de prioridade para selecionar a menor aresta que conecte o conjunto de vértices já visitados ao restante do grafo
Vértice 1: fila de prioridade = {4,2,3}
Vértice 2 = {1,3}
Vértice 3 = {4,**5**,6,2,1}
Vértice 4 = {1,3,5}
Vértice 5 = {3,7,4}
Vértice 6 = {3,7,8}
Vértice 7 = {8,6,5}
Vértice 8 = {7,6}

Vértice 1: fila de prioridade = {4,2,3} - Escolho o 4
Vértice 4 = {1,3,5} - escolho o 3, porque o 1 já foi visitado
Vértice 3 = {4,2,1,} - escolho o 2, porque o 1 já foi visitado
Vértice 2 = {1,3} - Todos os vértices que conectam ao 2 já foram visitados, então eu volto pro Vértice 3 sem fechar uma aresta.
Vértice 3 = {4,2,1,} - Volto pro vértice 4 porque todos já foram escolhidos
Vértice 4 = {1,3,5} - Escolho o 5, porque ele não foi visitado
Vértice 5 = {7,4} - Escolho o 7, porque ele não foi visitado
Vértice 7 = {8,6,5} - Escolho o 8
Vértice 8 = {7,6} - Escolho o 6, porque o 7 já foi
Vértice 6 = {3,7,8} - não escolho nenhum
RESUMO:
Pares utilizados: (1,4),(4,3),(3,2),(4,5),(5,7),(7,8)(8,6)
Passo #3) Repita até que todos os vértices estejam incluídos.

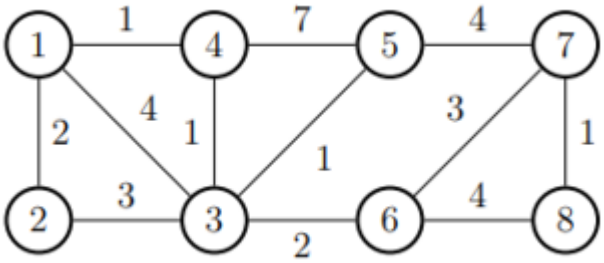


Custo total: 22

Exercício 3. *Considere o grafo abaixo, apresente uma árvore geradora de custo mínimo obtida pelo algoritmo de Kruskal.*



Exercício 4. *Mostre um grafo simples ponderado que possui duas árvores geradoras de custo mínimo distintas.*

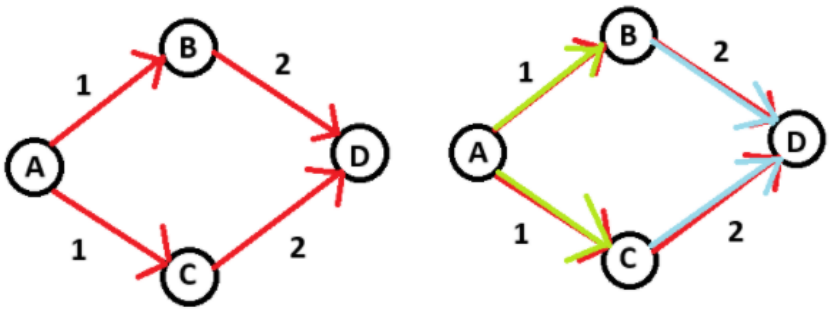


Bem, ao aplicar o algoritmo de Kruskal ou de Prim, o critério de desempate sobre arestas de mesmo peso pode determinar qual árvore geradora de custo mínimo vai ser gerada. Um grafo ponderado pode ter dois caminhos com o mesmo custo, com vértices diferentes, de modo que visitamos todos os vértices pelo menos 1 vez.

Exercício 5. *Os algoritmos de Prim e Kruskal, também sofrem problemas se os pesos das arestas forem negativos?*

Não há problema em haver pesos negativos em um grafo ponderado em nenhum dos algoritmos, visto que ele evita loops. Se houver loops, ele não utilizará a(s) aresta(s) com maior peso, ele conectará todos os nós pertencentes ao ciclo, mas sem fechar o ciclo.

Exercício 6. *O algoritmo de Kruskal pode dar árvores diferentes dependendo de como os empates são resolvidos. Dê um exemplo de grafo onde o algoritmo encontra duas árvores geradoras mínimas diferentes para dois critérios de desempate diferentes.*



O algoritmo de Kruskal irá ordenar as arestas de acordo com seus pesos, e adicionar à árvore geradora mínima de modo a evitar ciclos. As arestas A-B e A-C são escolhidas (em verde), pois chegam a vértices não descobertos ainda. Mas as arestas B-D e C-D são arestas que chegam em D, o último vértice não descoberto, e também possuem o mesmo peso. Logo, deve-se utilizar de alguma estratégia para desempatar as arestas B e C (que estão em azul)

Exercício 7. *Seja uv uma aresta de peso máximo de um ciclo de G. Mostre que G e $G-uv$ (o grafo obtido de G pela remoção de uv) possui uma mesma árvore geradora mínima.*

Se a aresta *uv* é a de peso máximo do grafo *G*, na hora de realizar o algoritmo para encontrar a árvore geradora mínima, ele não vai ser escolhido, não conectando os vértices *uv*, o que é a mesma coisa de *G-uv* (remover a aresta *uv* do grafo *G*). Como uma árvore geradora mínima não admite ciclos, basta remover a aresta de retorno que torna o grafo um ciclo de maior peso, que é a mesma coisa de *G-uv*

Exercício 8. *Assuma que os pesos das arestas do grafo de entrada são todos inteiros e limitados por um valor W fixo e conhecido. É possível melhorar a complexidade do algoritmo de Kruskal?*

Sim, utilizando um algoritmo de ordenação chamado counting sort, que tem complexidade O(E+W), ao invés de algoritmos envolvendo comparação como o *quicksort* ou o *merge sort*, que tem complexidade O(E log E).

Exercício 9. *Assuma que os pesos das arestas do grafo de entrada são todos inteiros e limitados por um valor W fixo e conhecido. É possível melhorar a complexidade do algoritmo de Prim?*

também não sei

Exercício 10. *Escreva um algoritmo para decidir se um dado grafo é conexo. Que tipo de certificado o seu algoritmo pode devolver para provar uma resposta afirmativa? E para provar uma resposta negativa?*

Um grafo é conexo se existir pelo menos um caminho entre cada par de vértices do grafo. Eu posso aplicar um BFS (Busca em largura) e a cada nó encontrado, ele entra para uma lista. Posso comparar esta lista com a lista de vértices do grafo.