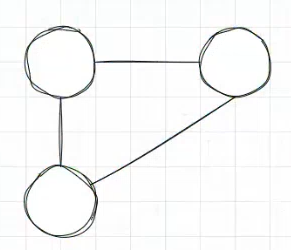
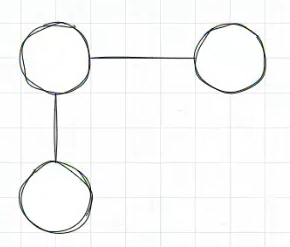
Algoritmo de **Kruskal**

* Voltamos a considerar Grafos Não Direcionados;
* Grafos Possuem Pesos nas Arestas;
* Queremos extrair a **Árvore Geradora Mínima** de um Grafo.

No contexto de grafos, o que é **Árvore**?

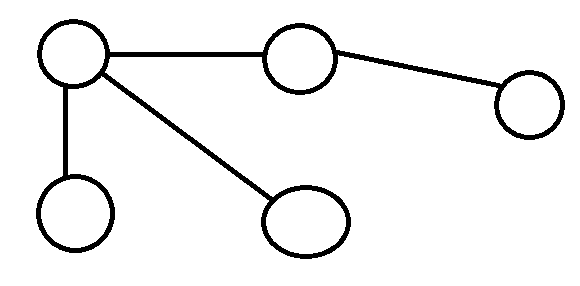
* **Árvore** é um Grafo **sem ciclo**:

Não é Árvore (Tem Ciclo) É Árvore (Não tem Ciclo)

No contexto de grafos, o que é **Árvore Geradora**?

* Conjunto de arestas que **conecta todos os vértices**, sem formar um ciclo.

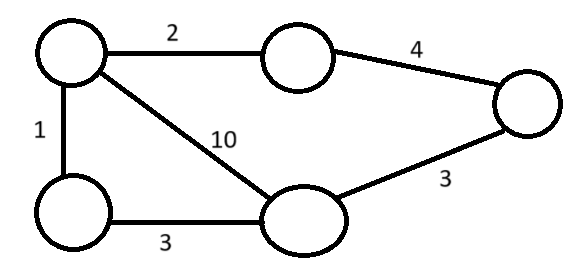


No contexto de grafos, o que é **Árvore Geradora Mínima**?

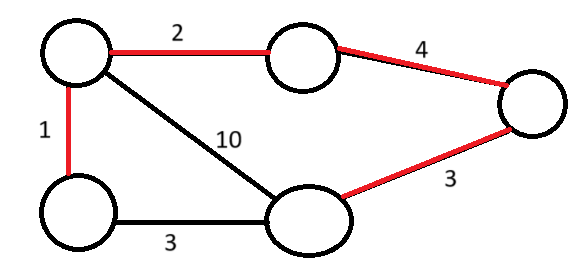
* O conjunto de arestas **de menor custo** que conecta todos os vértices, sem formar um ciclo.
* Um grafo pode ter duas árvores geradoras mínimas.

Algoritmo de **Kruskal**

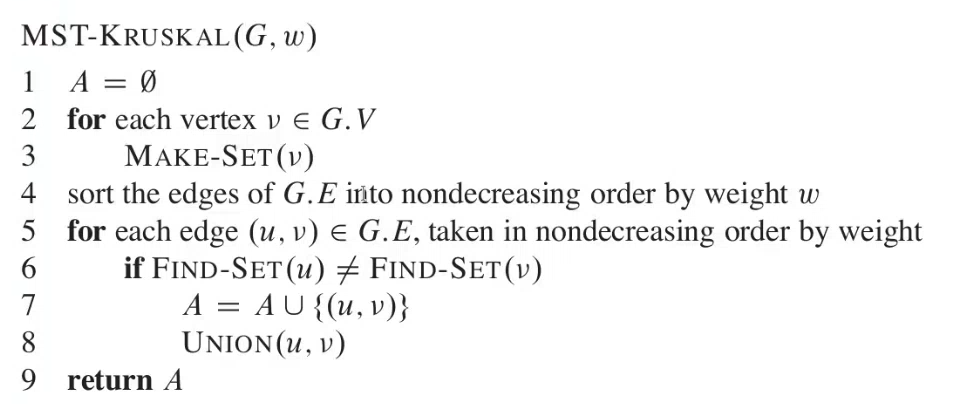
* Exemplo de **Entrada**:

****

* Exemplo de **Saída**: Encontra-se o **custo** da **Árvore Geradora Mínima**.
  + Neste caso, **custo** = 2 + 4 + 3 + 1 = 10



**Pseudocódigo** do Algoritmo de **Kruskal:**



* **Make-Set(v):** Cria um conjunto onde o elemento v é o único membro.
  + Inicialmente, cada vértice é o seu próprio conjunto.
  + Na prática, pensando na implementação em código, criaremos uma lista e cada vértice será uma das posições dessa lista.
* **Find-Set(v):** Quando estamos avaliando uma aresta (por exemplo, A–B):
  + Perguntamos: "A e B estão no **mesmo conjunto**?"
  + Se FIND-SET(A) ≠ FIND-SET(B), significa que: Eles estão **em conjuntos diferentes**, então **podemos adicionar essa aresta** (não vai formar ciclo).
* **Union(u, v):** Une dois conjuntos diferentes, o conjunto de u com o conjunto de v.
  + Na prática, pensando na implementação em código, dizemos que o índice u (de um vetor de representantes) "aponta" para o conjunto de v.