Resumo: Árvore Geradora Mínima, Algoritmo de Kruskal e de Prim

Árvore (Tree)

- Definição:
  - Uma **árvore** é um grafo **acíclico e conexo**.
- Teorema 1:
  - Uma árvore tem **um caminho entre qualquer par de vértices**.
  - *Formalmente*: $\forall\, u, v \in V, \exists\, \text{path}(u, v)$.
- Conceitos:
  - Um **vértice de grau 1** é chamado de **folha**.
  - Em alguns contextos, **vértices de grau 0** também podem ser considerados **folhas**.
  - Um **vértice com o grau maior que 1** é chamado de **vértice interno**.
- Teorema 2:
  - Toda árvore com pelo menos dois vértices possui **pelo menos duas folhas**.
- Teorema 3:
  - Toda árvore com $|V| \geq 1$ tem exatamente $|V| - 1$ arestas, ou seja: $|E| = |V| - 1$.

Árvore Geradora (Spanning Tree)

- Definição:
  - Uma árvore geradora é um subgrafo que é uma árvore e contém todos os vértices do grafo original e um subconjunto de arestas que preserva a conectividade.
  - *Formalmente*: $T = (V, E'), E' \subset E$.
- Algoritmo construtivo:
  1. Comece com um grafo conexo.
  2. Remova arestas de ciclos até não haver mais ciclos.
  3. O subgrafo restante será uma árvore geradora.
- Teorema:
  - Um grafo conexo com $|V|$ vértices e $|V| - 1$ arestas é uma árvore.
- Conceitos:
  - Branch: aresta presente na árvore geradora.
  - Chord: aresta que não faz parte da árvore geradora, mas está no grafo original.

Floresta (Forest)

- Definição:

    - Uma **floresta** é um grafo **acíclico**, em que **cada componente conexo é uma árvore**.

      > Uma floresta é uma **coleção de árvores**.

- Teorema:
    - Uma floresta com n vértices e k componentes conexos (ou árvores) possui exatamente $n - k$ arestas.
    - *Formalmente*: $|E| = |V| - |CC|$, onde $|CC| = k$ é o número de componentes conexos.

Floresta Geradora (Spanning Forest)

- Definição:
    - Uma **floresta geradora** é uma **coleção de árvores geradoras**, uma para cada componente conexo de um grafo desconexo.

Árvore Geradora Mínima (Minimum Spanning Tree - MST)

- Objetivo:
    - Encontrar uma árvore geradora com o menor custo total de arestas.
- Instância:
    - Grafo não direcionado $G = (V, E)$ com função de custo $w : E \to \mathbb{R}^+$.
- Solução:
    - Subconjunto de arestas $T \subseteq E$ tal que (V, T) é árvore e minimiza o custo total: $\sum_{e \in T} w(e)$.

Algoritmos para Árvore Geradora Mínima

Algoritmo de Kruskal

- Ideia:
    - Constrói a MST adicionando as arestas de menor peso uma a uma, desde que não formem ciclos.
- Descrição matemática:
    - Dado um grafo conexo não direcionado $G = (V, E)$ com pesos $w : E \to \mathbb{R}^+$, o algoritmo busca um subconjunto $T \subseteq E$ tal que:
        - (V, T) é uma árvore,
        - $|T| = |V| - 1$,
        - e $\sum_{e \in T} w(e)$ é minimizado.
- Algoritmo:
    1. Inicialize $T = \emptyset$ (subconjunto de arestas da MST).

2. Ordene as arestas E em:
    1. Ordem monotonicamente crescente de peso, se todos os pesos forem diferentes.
    2. Ordem não decrescente de peso, se houver pesos iguais.
3. Para cada aresta $e_i = (u, v)$ na ordem:
    - Se u e v estão em componentes distintos (não conectados em T):
    - Adicione e_i a T.
    - Caso contrário, ignore e_i (pois formaria ciclo).
    1. Pare quando $|T| = |V| - 1$.

> **Observação**
>
> Para verificar componentes disjuntos, utiliza-se a estrutura **Union-Find (Disjoint Set Union - DSU)**.

Algoritmo de Prim

- Ideia:
    - Começa com um único vértice e expande a árvore, a cada passo, com a aresta de menor custo que conecta um novo vértice.
- Descrição matemática:
    - A cada iteração, o algoritmo mantém um subconjunto $S \subseteq V$ de vértices já conectados, e adiciona a aresta de menor peso entre S e $V \setminus S$.
- Algoritmo:
    1. Escolha um vértice arbitrário $s \in V$ para iniciar.
    2. Inicialize $S = \{s\}$ e $T = \emptyset$.
    3. Enquanto $S \neq V$:
        - Encontre a aresta de menor peso (u, v) tal que $u \in S$ e $v \notin S$:
        - $(u, v) = \arg \min\limits_{(x,y) \in \mathrm{cut}(S)} w(x, y)$.
        - Adicione v a S e a aresta (u, v) a T.
    4. Ao final, (V, T) será a MST.

> **Observação**
>
> Usualmente implementado com uma **fila de prioridade (heap)** para otimizar a seleção da aresta de menor peso.

Algoritmo Reverse-Delete

- Ideia:

    - Começa com todas as arestas do grafo e remove as de maior peso, desde que a conectividade não seja perdida.
- Descrição matemática:
    - Arestas são removidas de E em ordem decrescente de peso, mantendo sempre o grafo conexo. O subconjunto final $T \subseteq E$ forma a MST.
- Algoritmo:
    1. Inicialize $T = E$.
    2. Ordene as arestas em:
        1. Ordem monotonicamente decrescente de peso, se todos os pesos forem diferentes.
        2. Ordem não crescente de peso, se houver pesos iguais.
    3. Para cada aresta $e_i = (u, v)$ na ordem:
        - Remova temporariamente e_i de T.
        - Se $(V, T \setminus \{e_i\})$ continua conectado:
        - Remova e_i permanentemente.
        - Caso contrário, mantenha e_i.
    4. Ao final, T será uma árvore geradora mínima.

> **Observação**
>
> Esse algoritmo é menos eficiente que Kruskal e Prim, pois exige verificar conectividade após cada remoção (pode exigir busca em profundidade ou largura repetidamente).

Observações Gerais

- Para tratar o caso de **arestas com pesos repetidos**, é possível aplicar uma pequena perturbação aleatória nos pesos.
- Qualquer algoritmo que **inclui arestas com base na propriedade do corte** e **exclui arestas com base na propriedade do ciclo** produzirá uma MST correta.
    - **Propriedade do Corte**:
        - Dado um corte qualquer $S \subset V$, a **menor aresta** que cruza o corte (ou seja, que conecta S a $V \setminus S$) está presente em **toda MST**.
    - **Propriedade do Ciclo**:
        - Em qualquer ciclo do grafo, a **aresta de maior peso não pertence a nenhuma MST**.
- Se todas as arestas possuem **pesos distintos**, a MST gerada é **única**, e todos os algoritmos corretos retornarão a mesma solução.
    - Porém, se houver **arestas com pesos iguais**, é possível que diferentes algoritmos (ou diferentes execuções de um mesmo algoritmo) resultem em **MSTs distintas**, embora todas tenham o **mesmo custo total mínimo**.

        - A diferença ocorre porque empates nas escolhas podem levar a diferentes estruturas, especialmente em presença de ciclos.

Árvore de Stainer (Stainer Tree)

- Definição
    - Dado:
        - Um grafo **não direcionado e conexo** $G = (V, E)$ com pesos associados às arestas: $w : E \to \mathbb{R}^+$.
        - Um subconjunto de vértices terminais $T \subseteq V$.
    - Objetivo do problema da árvore de Steiner:
        - É encontrar um **subgrafo em forma de árvore** $H = (V', E')$ de G, tal que:
            - $T \subseteq V' \subseteq V$,
            - H é conexo,
            - O **custo total** das arestas de H (isto é, $\sum_{e \in E'} w(e)$) seja **mínimo**.
        - Ou, encontrar um **subgrafo com algum critério de otimização**.
    - Os vértices em $V' \setminus T$ são chamados de **pontos de Steiner**, são vértices **adicionados** à solução para conectar os terminais com menor custo.
- Comparação com MST:

Árvore Geradora Mínima (MST)	Árvore de Steiner
Conecta **todos os vértices** do grafo	Conecta apenas um subconjunto $T \subset V$
Não adiciona vértices extras	Pode **adicionar vértices auxiliares** (Steiner)
Algoritmos eficientes como Kruskal e Prim	Problema NP-difícil, exige aproximação ou busca exata
Custo total mínimo sobre todos os vértices	Custo mínimo para conectar apenas os terminais

- Complexidade:
    - O problema da árvore de Steiner em grafos gerais é **NP-difícil**.
    - Nenhuma solução é conhecida para o problema da árvore de Steiner **em grafos**.

FIM