Resumo: Problema do Menor Caminho, Algoritmo de Dijkstra e de Bellman-Ford

Problema do Caminho Mínimo

- Instância:
  - Grafo direcionado $G = (V, E)$ com função de pesos $w : E \to \mathbb{R}$.
  - Um vértice de origem $s \in V$.
- Objetivo:
  - Encontrar o **menor caminho** de s até cada vértice $u \in V$.
- Definição:
  - O comprimento de um caminho é a **soma dos pesos** das arestas do caminho.
  - Se não há caminho de x para y, então $L_x(y) = \infty$.

Algoritmo de Dijkstra

- Requisitos:
  - Todos os pesos das arestas devem ser **não negativos**.
- Objetivo:
  - Encontrar os menores caminhos a partir da origem s para todos os vértices do grafo.
- Algoritmo Formal:
  - entrada: Um grafo $G = (V, E)$, um mapa de pesos W e um vértice inicial s.
  - saída: A distância do vértice s para todos os outros.
  1. $\forall\, v \in V$, defina $d[v] \leftarrow \infty$
  2. $d[s] \leftarrow 0$
  3. $S \leftarrow V$ (conjunto de vértices não explorados)
  4. Enquanto $S \neq \emptyset$:
     1. $v^* = \text{argmin}(d[u] \mid u \in S)$
     2. $S \leftarrow S \setminus v^*$
     3. Para cada $u \in N(v^+) \cap S$:
        - $d[u] \leftarrow \min(d[u], d[v^*] + w(v^*, u))$
- Algoritmo Descritivo:
  1. Começamos atribuindo uma distância infinita a todos os vértices, exceto o vértice de origem s, que recebe distância 0.
  2. Criamos um conjunto S com todos os vértices do grafo, indicando que todos ainda estão por explorar.
  3. Enquanto houver vértices em S:

        - Selecionamos o vértice v com a menor distância estimada $d[v]$ entre os que ainda estão em S.
        - Removemos v do conjunto S, marcando-o como processado.
        - Para cada vizinho u de v que ainda estiver em S, verificamos se o caminho passando por v é melhor que o caminho atual para u:
            - Se for, atualizamos a distância de u com esse novo valor: $d[u] = d[v] + w(v, u)$, onde $w(v, u)$ é o peso da aresta entre v e u.

Algoritmo de Bellman-Ford

- Vantagem:
    - Funciona com pesos negativos.
- Limitação:
    - Não funciona se houver **ciclo negativo acessível a partir da origem**.
- Algoritmo (Formal):
    - entrada: Um grafo $G = (V, E)$, um mapa de pesos W e um vértice inicial s.
    - saída: A distância do vértice s para todos os outros.
    1. $\forall\, v \in V$, defina $d[0, v] \leftarrow \infty$
    2. $d[0, s] \leftarrow 0$
    3. Para $i = 1$ até $n - 1$:
        - Para cada $v \in V$:
            - $d[i, v] = d[i - v, v]$
        - Para cada $\text{aresta}(u, v) \in E$:
            - $d[i, v] = min(d[i, v], d[i - 1, u] + w(u, v))$
- Algoritmo Descritivo:
    1. Começamos atribuindo uma distância infinita a todos os vértices, exceto o vértice de origem s, que recebe distância 0.
    2. Repetimos o processo de atualização das distâncias **exatamente** $n - 1$ **vezes**, onde n é o número de vértices no grafo:
        - Para cada vértice $v \in V$, copiar a estimativa da iteração anterior: $d[i, v] \leftarrow d[i - 1, v]$ (Ou seja, inicialmente, nada muda nesta rodada.).
        - Para cada aresta $(w, v) \in E$, verificar se o caminho passando por w melhora a distância até v: $d[i, v] = min(d[i, v], d[i - 1, u] + w(u, v))$.
- Algoritmo para achar Ciclo Negativo:
    1. Os passos do anterior.
    2. Se houver alguma aresta (u, v) tal que $d[n - 1, v] > d[n - 1, u] + w(u, v)$, então **existe um ciclo negativo acessível**.

Considerações sobre os Algoritmos

- Dijkstra é **mais eficiente**, mas **não tolera pesos negativos**.

- Bellman-Ford é **mais geral**, porém **mais lento**.
- A saída dos algoritmos forma uma **árvore de caminhos mínimos** a partir da origem.