INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
Campus Nova Cruz – Código INEP: 24083003
Av. José Rodrigues de Aquino Filho, Nº 640, RN 120, Alto de Santa Luzia, CEP 59215-000, Nova Cruz (RN)
CNPJ: 10.877.412/0015-633 – Telefone: (84) 4005-4107

Trabalho de Implementação:

Problema do Caminho Mais Curto (SPP – *Shortest Path Problem*)

(PROG – TADS1V)

1. Contextualização do Problema

Uma empresa de logística do Rio Grande do Norte deseja otimizar suas operações de transporte entre municípios da mesorregião do Agreste Potiguar. A malha rodoviária potiguar nessa região possui diversos municípios interligados por estradas, e a distância ou o tempo de viagem entre eles pode variar significativamente devido a fatores como a qualidade da estrada, volume de tráfego, características naturais, entre outros. Algumas rotas podem ser mais longas, outras mais curtas, e nem todos os municípios estão diretamente interligados. Além disso, devido a estradas de mão única em centros urbanos ou condições específicas de trechos, o tempo/custo de ir de uma cidade A para uma cidade B pode não ser o mesmo de ir de B para A. A empresa pretende determinar a rotas mais eficiente para cada par de municípios que ela atende, minimizando o tempo de viagem ou o custo de combustível, para aprimorar seus serviços e reduzir despesas operacionais.

2. Objetivos do Trabalho

- **Modelagem da Rede Intermunicipal**: representar alguns dos principais municípios do Agreste Potiguar como um grafo ponderado, em que os nós/vértices são os municípios e as arestas representam as rodovias, com pesos indicando a distância (em km) ou o tempo de viagem (em horas/minutos);
- **Implementação do Algoritmo**: implementar o tradicional algoritmo de Floyd-Warshall para determinar o caminho mais curto entre todos os pares de municípios do Agreste Potiguar que são atendidos pela empresa de logística;
- **Apresentação de Resultados**: apresentar o custo (distância ou tempo) da melhor rota entre dois municípios quaisquer atendidos pela empresa de logística. Ademais, disponibilizar a matriz de distâncias mínimas e matriz de predecessores após a execução do algoritmo de Floyd-Warshall;
- **Salvar e Carregar Dados de Arquivo**: permitir que dados importantes do problema sejam salvos e carregados de arquivo de texto, tais como o número de municípios, nomes dos municípios, matriz de adjacências, matriz de distâncias mínimas e matriz de predecessores.

3. Etapas do Trabalho

3.1 Definição do Grafo:

- Selecione um conjunto de 15 municípios da mesorregião do Agreste Potiguar. Entre eles, inclua os municípios de Nova Cruz, Santo Antônio, Monte Alegre, Passa e Fica e Brejinho;
- Defina as conexões (arestas) e seus pesos (distância em Km ou tempo de viagem em minutos entre os municípios). Pesquise dados reais de distâncias entre os municípios selecionados. Uma sugestão é utilizar o *Google Maps*, onde será possível identificar a conectividade entre os municípios escolhidos;
- Represente o grafo utilizando uma matriz de adjacências. Utilize um valor "infinito" para cidades que não possuem conexão direta.

3.2 Implementação do Floyd-Warshall:

- Implemente o algoritmo de Floyd-Warshall na linguagem de programação C;
- A função deve receber a matriz de adjacências como entrada e retornar a matriz de distâncias mínimas entre todos os pares de nós/cidades, além da matriz de predecessores, que é utilizada para formar as rotas/caminhos de menor custo entre pares de municípios.

3.3 Otimização de Rotas de Transporte:

- Escolha alguns pares de municípios da mesorregião do Agreste Potiguar e utilize as matrizes geradas pelo algoritmo de Floyd-Warshall para determinar o caminho mais curto entre eles, considerando o peso escolhido: distância ou tempo;
- Apresente o caminho detalhado para algumas rotas otimizadas, indicando a sequência de cidades e o custo/tempo total do caminho.

4. Algoritmo de Floyd-Warshall

O algoritmo de Floyd-Warshall tem sua origem nos estudos em teoria dos grafos e otimização realizados durante o século XX. A história desse algoritmo traz uma curiosidade: ele foi desenvolvido de forma independente e quase simultânea por diferentes pesquisadores (Bernard Roy, Stephen Warshall e Robert Floyd) no final da década de 1950 e início da década de 1960. Por essa razão, o algoritmo também é chamado de Roy-Floyd, Roy-Warshall ou até mesmo Roy-Floyd-Warshall.

O algoritmo de Floyd-Warshall é um dos algoritmos mais conhecidos e essenciais da Teoria dos Grafos, sendo utilizado para determinar os caminhos de menor custo entre todos os pares de vértices de um grafo ponderado. Isso implica que, se há uma rede de pontos (cidades, roteadores, entre outros) e sabe-se o custo de ir de um ponto a outro diretamente, o Floyd-Warshall indicará qual é o caminho de menor custo (ou mais rápido, ou de menor distância) entre qualquer par de pontos nessa rede. Ele pode ser usado em grafos direcionados, em que ir de A para B pode não ser o mesmo que ir de B para A, ou não direcionados. Uma vantagem do Floyd-Warshall é que ele sabe lidar com arestas de pesos negativos, desde que o grafo não possua ciclos negativos.

Devido a essas características, o algoritmo é classificado como um resolvedor de problemas de Caminho Mais Curto para Todos os Pares de Vértices (*All-Pairs Shortest Path*). Apesar de sua

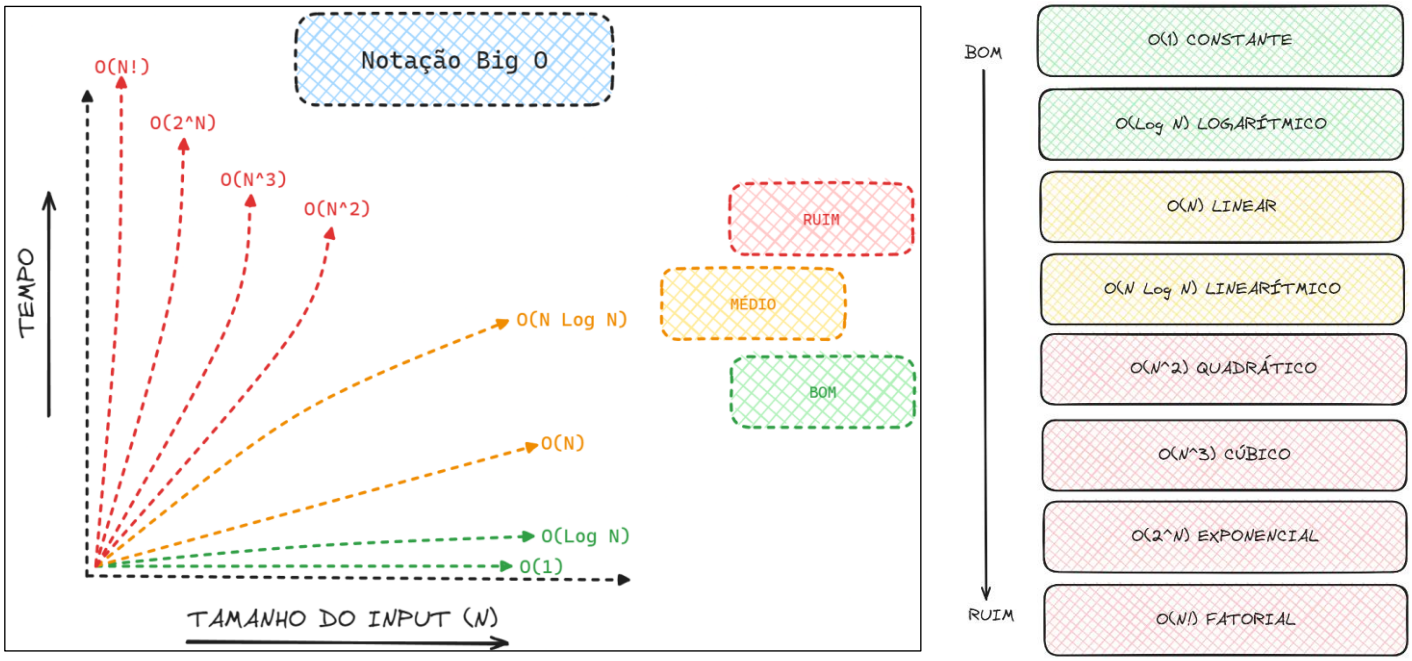
complexidade cúbica, $O(n^3)$, o algoritmo de Floyd-Warshall é utilizado na prática para grafos pequenos ou médios, em situações em que é necessário conhecer o custo mínimo ou distância mínima entre todos os pontos de interesse, como em problemas de logística e transporte (otimização de rotas, planejamento de entregas), roteamento de redes, fluxos de trabalho, entre outros.

Observações:

A complexidade de algoritmos é uma medida que indica o quão eficiente um algoritmo é, tanto em termos de **tempo de execução** quanto do uso de **espaço** de memória que ele consome, à medida que o tamanho da entrada de dados aumenta. Um algoritmo de complexidade cúbica, denotado como $O(n^3)$, significa que o tempo de execução (ou o consumo de recursos) cresce proporcionalmente ao cubo do tamanho da entrada n. Isso indica um crescimento muito rápido, tornando esse algoritmo impraticável para grandes volumes de dados. Um problema que leva um segundo para ser solucionado com $n = 100$ levaria aproximadamente 1000 segundos (cerca de 16 minutos) para $n = 1000$.

A complexidade computacional do Floyd-Warshall é de $O(n^3)$, o que significa que o tempo que ele permanece em execução aumenta cubicamente com o número de vértices. Para grafos pequenos e médios, isso é perfeitamente aceitável. Para grafos muito grandes, outros algoritmos seriam mais indicados, mas o Floyd-Warshall se destaca pela sua simplicidade e pela capacidade de encontrar todos os caminhos de uma só vez.

Figura 1. Comparativo entre complexidades de algoritmos – Notação *Big* O.



4.1 Matrizes do Algoritmo de Floyd-Warshall

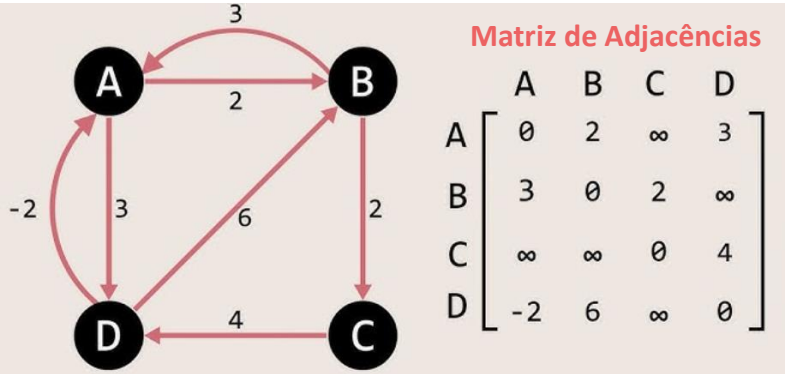
O algoritmo de Floyd-Warshall funciona manipulando três matrizes principais: matriz de adjacências, matriz de distâncias mínimas e matriz de predecessores. Em relação as suas dimensões, todas elas são matrizes quadradas de ordem n (matriz $n \times n$), em que n corresponde ao número de vértices (pontos)

do grafo. No contexto do problema deste trabalho, o número de vértices é dado pelo número de municípios selecionados.

Matriz de Adjacências

A matriz de adjacências A é o ponto de partida do algoritmo de Floyd-Warshall. Cada elemento $A[i][j]$ indica os vértices que estão diretamente conectados e o custo (distância, tempo, latência, etc.) do caminho direto do vértice i até o vértice j. Dessa forma:

- Se existe uma conexão direta do vértice i para o vértice j, existência da aresta (i, j), com um peso (custo, tempo, distância, *etc.*), esse valor é $A[i][j]$;
- Se não há uma conexão direta do vértice i para o vértice j, o valor de $A[i][j]$ é considerado "infinito" (ou muito grande), indicando que não há um caminho direto conhecido ou que ele é extremamente custoso;
- A distância de um vértice para ele mesmo é sempre igual a zero, ou seja, $A[i][i] = 0$.



Matriz de Distâncias Mínimas

A matriz de distâncias mínimas D é o principal resultado da execução do algoritmo de Floyd-Warshall. Ela armazena os menores custos (distâncias, tempo, *etc.*) para ir de qualquer vértice de origem a qualquer outro vértice de destino em um grafo ponderado.

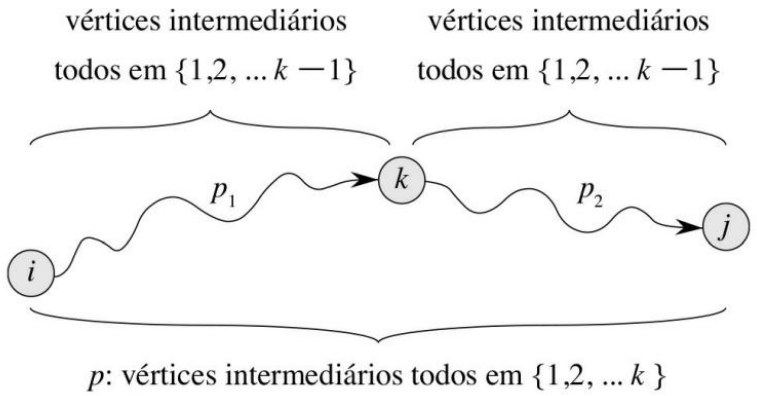
- O elemento $D[i][j]$ representa o custo mínimo para ir do vértice i até o vértice j, considerando caminhos diretos e indiretos;
- Inicialmente, $D[i][j]$ é o peso da aresta direta entre os vértices i e j, se existir, ou “infinito” (valor muito grande) se não houver ligação direta entre estes vértices;
- A diagonal principal $D[i][i]$ é inicializada com zero, pois a distância de um vértice para ele mesmo é zero.

No início da execução do algoritmo de Floyd-Warshall, a matriz D é idêntica à matriz de adjacências do grafo que modela o problema a ser solucionado. O algoritmo então atualiza a matriz D iterativamente, considerando todos os caminhos intermediários, até que ela contenha os menores caminhos possíveis entre todos os pares de vértices e se torne, definitivamente, a matriz de distâncias mínimas.

Durante a execução do algoritmo de Floyd-Warshall, ele considera cada vértice k como um intermediário possível entre todos os pares de vértice i e j. Em cada passo, ele tenta melhorar a estimativa atual de $D[i][j]$, verificando se passar por k resulta em um caminho mais curto:

$$D[i][j] = min(D[i][j], D[i][k] + D[k][j])$$

Isso significa que, para cada par de vértices i e j, o algoritmo verifica se o custo do caminho atual parar ir do vértice i até o vértice j ($D[i][j]$) é menor do que ir de i para k ($D[i][k]$) e depois de k para j ($D[k][j]$). Se a segunda opção for menor, ele atualiza $D[i][j]$ com esse novo valor.



p: vértices intermediários todos em {1,2, ... k }

Matriz de Predecessores

A matriz de predecessores P permite que, depois que a matriz de distâncias mínimas estiver pronta, seja possível reconstruir o caminho mínimo completo entre dois vértices i e j e não apenas o seu custo. Isso é feito percorrendo os predecessores de trás para frente, ou seja, de j para i. Assim, seja P a matriz de predecessores, com $P[i][j]$ representando o vértice imediatamente anterior a j no caminho mínimo de i até j, então:

- $P[i][j] = k$ indica que o caminho mínimo do vértice i até o vértice j passa pelo vértice k antes de chegar ao vértice j;
- Se não houver caminho do vértice i até o vértice j, então $P[i][j] = -1$ (ou $P[i][j] = null$, dependendo da implementação);
- Se existe uma aresta direta (caminho direto) do vértice i para o vértice j, então $P[i][j] = i$.

Sempre que o algoritmo de Floyd-Warshall encontra um caminho mais curto do vértice i até o vértice j passando por um vértice intermediário k, ele atualiza a matriz de distâncias mínimas e a matriz de predecessores:

$$Se\ D[i][k] + D[k][k] < D[i][j]\ \ então\text{:}\ \begin{cases} D[i][j] = D[i][k] + D[k][j] \\ P[i][j] = P[k][j] \end{cases}$$

A partir da determinação da matriz de predecessores e da matriz de distâncias mínimas, torna-se fácil obter uma rota entre dois vértices do grafo. Para encontrar o melhor caminho entre o vértice de origem i e o vértice de destino j os seguintes passos podem ser adotados:

1. **Inicialização:** inicie com uma lista vazia que representará o caminho;
2. **Verificação de Conexão:** verifique se existe um caminho entre i e j. Se o valor na matriz de distâncias mínimas gerada pelo algoritmo de Floyd-Warshall entre i e j for "infinito" ($D[i][j] = \infty$), significa que não há caminho entre eles;
3. **Construção Reversa do Caminho:** adicione o vértice de destino j à lista do caminho. Em seguida, retroceda a partir de j usando a matriz de predecessores. O predecessor de j no caminho de i para j é $P[i][j]$. Adicione $P[i][j]$ à lista que representa o caminho;
4. **Iteração:** o próximo vértice a ser adicionado à lista é $P[i][P[i][j]]$ e assim por diante. Este processo continua até que o vértice a ser adicionado na lista que representa o caminho seja o vértice de origem i;
5. **Inverter o Caminho:** uma vez que tenha sido percorrido todos os predecessores até chegar ao vértice de origem i, a lista que representa o caminho estará na ordem inversa (do destino para a origem). Simplesmente inverta essa lista para obter o caminho na ordem correta, da origem ao destino.

4.2 Pseudocódigo do Algoritmo de Floyd-Warshall

No quadro a seguir é apresentado o pseudocódigo do algoritmo de Floyd-Warshall. Ele recebe como entrada a matriz de adjacências A e o número de vértices do grafo. Os três laços de repetição aninhados conferem ao algoritmo uma complexidade $O(n^3)$. Deve-se destacar que é possível encontrar na literatura algumas adaptações no teste efetuado na linha 11 do pseudocódigo para garantir um pouco mais de eficiência do algoritmo.

De acordo com o pseudocódigo, uma matriz D é utilizada para representar a matriz de distâncias mínimas. Entretanto, é possível fazer as atualizações de custo na própria matriz de adjacências inicialmente fornecida. Dessa maneira, não seria necessário o consumo de espaço adicional de memória com uma segunda matriz, mas a informação inicialmente contida na matriz de adjacências seria perdida devido às atualizações.

Pseudocódigo: Algoritmo de Floyd-Warshall

```
1:  Inicialize a matriz de distâncias D com os valores da matriz de adjacências A.
2:  Inicialize a matriz de predecessores P:
3:      Para cada D[i][j]:
4:          Se D[i][j] é infinito então P[i][j] = −1
5:          Senão P[i][j] = i                  // o próprio i é o predecessor do caminho direto
6:
7:  Para cada vértice k:                       // considerado como intermediário
8:      Para cada vértice i:                   // considerado como origem
9:          Para cada vértice j:               // considerado como destino
10:             // verifica se o caminho i → j → k é menor do que o caminho i → j
11:             Se D[i][k] + D[k][j] < D[i][j] então
12:                 D[i][j] = D[i][k] + D[k][j]      // atualiza a distância mínima
13:                 P[i][j] = P[k][j]                // atualiza o predecessor para j
```

5. Funcionamento do Programa

O programa a ser desenvolvido neste trabalho deve ser estruturado com o uso de funções. Dessa forma, é importante que o programa seja particionado em pequenas etapas, as quais irão dar origem às funções e procedimentos do código. Na implementação, os seguintes pontos também devem ser levados em consideração:

- Permitir que o usuário informe os nomes dos municípios que serão utilizados pelo algoritmo de Floyd-Warshall;
- Permitir que o usuário realize a inserção dos elementos da matriz de adjacências, considerando os municípios escolhidos;
- Permitir que o usuário salve em um único arquivo de texto os dados de número de municípios, nomes dos municípios e matriz de adjacências em um formato adequado;
- Permitir a impressão de qualquer uma das matrizes de interesse do algoritmo de Floyd-Warshall em momentos oportunos;
- Permitir que o usuário carregue o arquivo de texto com os dados necessários para executar o algoritmo de Floyd-Warshall (número de municípios, nomes dos municípios e matriz de adjacências) para evitar que o usuário tenha que digitar as informações todas as vezes que for executar o programa;
- Permitir que o usuário salve em arquivo de texto, os dados das matrizes de distâncias mínimas e predecessores resultantes da execução do algoritmo de Floyd-Warshall, o que garantirá o armazenamento dos resultados;
- Permitir que, após a execução do algoritmo, o usuário possa selecionar um município de origem e outro de destino e o programa exiba o menor caminho entre estes municípios e o custo (distância ou tempo) para efetuar esse caminho.