



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

RELATÓRIO TÉCNICO

DESCOBRIR ROTAS EM UM MAPA

Beatriz Abreu

Luiza Kuze

RESUMO

O trabalho consiste em desenvolver um algoritmo eficiente que possa determinar a melhor rota entre duas cidades distintas escolhidas pelo usuário, bem como a distância percorrida no trajeto completo. Para isso, o programa recebe um arquivo CSV, que contém dados sobre a distância entre pares de cidades.

Para a implementar o código, a equipe utilizou como base o algoritmo de Dijkstra, um algoritmo de busca em grafos que encontra o caminho mais curto entre um nó de partida e todos os outros nós em um grafo.

São José, 08 de maio de 2023



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

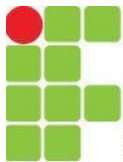
INTRODUÇÃO

Encontrar a melhor rota entre duas cidades distintas pode ser útil em diversas áreas, tais como o turismo e a logística. No turismo, com a rota definida, é possível planejar melhor as paradas, pontos turísticos a serem visitados, e estimar o tempo total de viagem. Na logística e transporte, com as rotas mais eficientes traçadas, a empresa pode economizar em combustível e tempo, além de garantir entregas mais rápidas e eficazes.

Com essas aplicações em mente, foi proposta a elaboração de um programa capaz de mostrar, dado um arquivo CSV em que cada linha contém um par de cidades e a distância entre elas, a melhor rota e a melhor distância entre duas cidades presentes nessa base de dados fornecida.

No desenvolvimento do código, foi utilizado o algoritmo de Dijkstra que é essencial para a lógica do programa. Ele é especialmente útil para encontrar a melhor rota entre duas cidades, já que considera o peso de cada aresta (nesse caso, a distância entre as cidades) para determinar o caminho mais curto.

Além disso, estruturas de dados tanto lineares quanto associativas foram utilizadas no projeto. Essas estruturas ajudaram a otimizar o programa e torná-lo mais eficiente, especialmente quando se trata de lidar com grandes quantidades de dados.



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

DESENVOLVIMENTO

O programa começa da seguinte forma: Na função *main*, cria uma tabela hash chamada “mapa” que tem como chave o nome de uma cidade e como valor uma lista de struct “adjacente” que armazena as cidades adjacentes a cidade chave e suas respectivas distâncias.

A tabela hash, estrutura de dados associativa, é uma excelente alternativa para armazenar esses dados, justamente por essa ideia de um dado que é acessado pela sua chave, assim fica fácil acessar quais são as cidades adjacentes de uma respectiva cidade. Em outras palavras, essa estrutura é uma ótima opção aqui por conta da sua capacidade de procurar, armazenar e obter elementos dentro de um grande conjunto de dados, o que faz todo sentido nessa parte inicial do projeto.

```
unordered_map <string, list<adjacente>> mapa;
```

Figura 1 - Criando tabela hash para armazenar dados

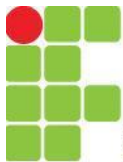
```
struct adjacente {  
    int distancia;  
    string cidade;  
};
```

Figura 2 - Struct “adjacente” utilizada na tabela hash “mapa”

Após criar a tabela hash anterior, é necessário colocar os dados dentro dela. Assim, foi desenvolvida uma função chamada **le_arquivo** para realizar a leitura de um arquivo CSV contendo dados sobre a distância entre pares de cidades. Esse arquivo CSV é colocado como primeiro argumento do programa e é o primeiro parâmetro desta função.

```
le_arquivo( nome_arquivo: argv[1], & mapa);
```

Figura 3 - Chamando a função de leitura de dados



Em seguida, já tendo um “mapa” da região que vai ser analisada a melhor rota, é preciso ter uma entrada de dados, pois é o usuário quem decide as cidades de partida e de destino do trajeto. Com isso, é chamada a função **entrada_dados**, que verifica imediatamente se esses dados coletados estão corretos e só retorna valores às variáveis “partida” e “destino” ao receber uma entrada válida do usuário, ou seja, uma cidade escolhida que exista no arquivo CSV fornecido pelo usuário.

```
string partida, destino;
```

Figura 4 - Criando string para armazenar as cidades de partida e destino

```
partida = entrada_dados( trajeto: "partida", tabela: mapa);  
destino = entrada_dados( trajeto: "destino", tabela: mapa);
```

Figura 5 - Chamando função para entrada de dados do usuário

Nessa parte da execução já temos: Um mapa da região de análise da melhor rota e cidades de partida e destino selecionadas pelo usuário. Assim, entra a parte da lógica do algoritmo de Dijkstra, que é a base da implementação da função **tabela_dijkstra**.

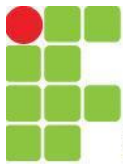
Essa nova função retorna uma tabela hash semelhante ao que se encontra no original algoritmo de Dijkstra, contendo como chave da tabela uma cidade e como valor uma struct “nodo” que armazena o custo (nesse caso, a distância) que essa cidade chave tem até o destino e o precedente que é a próxima cidade que o usuário deve seguir a partir desse nodo atual que está para encontrar a menor rota até o seu destino.

```
unordered_map<string, nodo> tabela_dijkstra = dijkstra( & mapa, destino);
```

Figura 6 - Chamando função que retorna a tabela dijkstra feita com dados do usuário

```
struct nodo {  
    int custo;  
    string precedente;  
};
```

Figura 7 - Struct “nodo” utilizada na tabela dijkstra



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

Novamente entra o uso da estrutura da tabela hash, pois ela permite que se acesse elementos em tempo constante, o que é importante para o desempenho do algoritmo. Além disso, vale lembrar que a tabela hash permite que se armazene pares chave-valor, em que outra vez será uma função extremamente útil dessa estrutura. Nesse caso, já relacionando com a nomenclatura utilizada no algoritmo de Dijkstra, a chave dessa tabela é uma string que representa um nó do grafo e o valor é uma struct “nodo” que contém informações sobre a distância até o nó de origem e o nó anterior no caminho mais curto.

A ideia da implementação utilizando o algoritmo de Dijkstra foi disponibilizada logo na descrição inicial do projeto, juntamente com um pseudo-código que facilitou bastante o trabalho.

Passo 1: Cria a tabela D, de forma que, para cada nodo do mapa:

$D[\text{nodo}] = (\text{infinito}, \text{próximo_nodo})$

OBS: $D[\text{nodoA}] = (0, \text{nodoA})$, sendo nodoA o destino

Passo 2: Cria Q, que é uma lista contendo todos os nodos (incluindo nodoA)

Passo 3: Enquanto Q não estiver vazio, faça o seguinte:

Extraia de Q o nodo u, que é o nodo com menor distância até nodoA

Para cada nodo v vizinho de u, faça o seguinte:

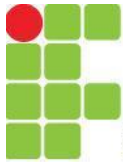
Calcule a distância do nodo v: $\text{dist_v} = \text{dist_u} + \text{distancia}(u, v)$

Se dist_v for menor do que a distância contida em $D[v]$, faça isto:

Atualize $D[v] = (\text{dist_v}, u)$

Figura 8 - Algoritmo inicial do projeto

Um detalhe é que nesse pseudo-código, há outra estrutura de dados ainda não mencionada no desenvolvimento. É a lista, uma estrutura de dados linear que já foi vista na unidade 1 da disciplina, a função dela aqui é a representação dos nodos que ainda não foram processados pelo algoritmo e, portanto, ainda podem ter suas distâncias atualizadas. Diferente das ocasiões anteriores, não temos um par de informações para analisar, ou seja, não há a necessidade de outra tabela hash. Pode-se entender que a escolha da lista nesse momento vem da facilidade de acessar, adicionar ou remover elementos individuais em um conjunto de dados.



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

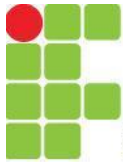
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

Com uma tabela dijkstra repleta de informações importantes de distâncias, basta somente acessar os dados dessa tabela e fazer a saída de dados para o usuário obter a resposta sobre a sua melhor rota e distância. Isso é feito em suas funções diferentes para facilitar o entendimento, essas funções recebem os nomes **melhor_rota** e **melhor_distancia**.

```
cout << endl << "-> A melhor rota é: " << endl << melhor_rota( &tabela_dijkstra, partida, destino) << endl << endl;  
cout << "-> A melhor distância é: " << endl << melhor_distancia( &tabela_dijkstra, partida) << " km" << endl;
```

Figura 9 - Saída de dados



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

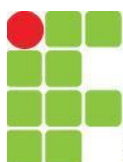
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

RESULTADOS

Os resultados obtidos pelo programa são bastante positivos e demonstram que ele é capaz de cumprir com sucesso a tarefa proposta de encontrar a menor rota entre duas cidades, considerando os dados do arquivo CSV fornecido.

Dessa forma, esses resultados são bastante promissores e demonstram a eficácia do programa. Em resumo, o programa desenvolvido é capaz de auxiliar os usuários a escolherem a rota mais conveniente e eficiente para o seu deslocamento. Tendo como parâmetro que a melhor rota é a rota com a menor distância possível a ser percorrida entre uma cidade de partida até o seu destino.



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

MANUAL

Atenção, Usuário!!

Os manuais de compilação e execução tanto para Linux quanto para Clion foram preparados utilizando um exemplo de base de dados que é o arquivo “*distancias.csv*”. Caso a sua base de dados não seja essa base teste incluída no programa, não esqueça de incluir o seu arquivo.csv no diretório do projeto.

Esse arquivo deve ter o formato a seguir em cada uma das suas linhas:

cidade1,cidade2,distancia

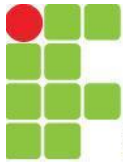
Sendo que, “cidade1” e “cidade2” devem ser um par de cidades (cidades diferentes) e “distancia” a distância entre esse par de cidades. A imagem abaixo deixa explícito como deve ser o formato para que o programa seja executado corretamente.

```
Agrolândia,Ituporanga,27
Agrolândia,Rio do Sul,33
Alfredo Wagner,Bom Retiro,24
Alfredo Wagner,Ituporanga,56
Alfredo Wagner,Leoberto Leal,31
Alfredo Wagner,Rancho Queimado,47
Angelina,Major Gercino,27
```

Figura 10 - Exemplo do formato da base de dados do arquivo “distancias.csv”

Além disso, vale prestar atenção que em cada uma das três colunas (“cidade1”, “cidade2” e “distancia”) há uma vírgula separando as informações. O que é uma característica do arquivo csv, portanto esse separador não pode ser esquecido.

Quando a sua base de dados cumprir os requisitos propostos, somente será necessário colocar o nome do seu arquivo no lugar de “*distancias.csv*” ao longo dos passos do manual. Para facilitar, uma dica é já deixar o nome do seu arquivo como “*distancias.csv*”, assim dificilmente você terá complicações no programa.



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

Compilação e execução no sistema operacional Linux:

- 1) Se você baixou um arquivo compactado contendo o projeto, descompacte-o para um diretório de sua escolha. Certifique-se de lembrar o local onde descompactou o arquivo.
- 2) Abra uma janela de terminal e acesse o diretório onde os arquivos do projeto estão localizados, utilizando o comando `"cd [nome_diretorio]"`. Onde `"nome_diretorio"` é o nome do diretório em que está o projeto.
- 3) Recomendação: Digite o comando `"ls"` para listar todos os arquivos no diretório atual. Certifique-se de que os arquivos `"main.cpp"` e `"distancias.csv"` estão presentes.
- 4) Para compilar o projeto, digite o seguinte comando no terminal: `"g++ main.cpp funcoes.cpp -o projeto"`. Este comando irá compilar os arquivos fonte `"main.cpp"` e `"funcoes.cpp"` e gerar o executável `"projeto"`.
- 5) Para executar o programa, digite o seguinte comando no terminal: `"./projeto distancias.csv"`. O processo de execução será retomado mais adiante nesse manual.

Revisão dos comandos

1) `cd [nome_diretorio]`

2) `ls`

3) `g++ main.cpp funcoes.cpp -o projeto`

4) `./projeto distancias.csv`

Figura 11 - Revisão dos comando para compilação e execução via Linux



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

Compilação e execução no Clion:

- 1) Se você baixou um arquivo compactado contendo o projeto, descompacte-o para um diretório de sua escolha. Certifique-se de lembrar o local onde descompactou o arquivo.
- 2) Abra o CLion em seu computador e acesse o projeto.
- 3) Com o arquivo aberto no CLion, localize na barra de ferramentas superior o painel de configuração *"Edit Run/Debug Configurations"* (figura 12) e clique em *"Edit Configurations..."*.

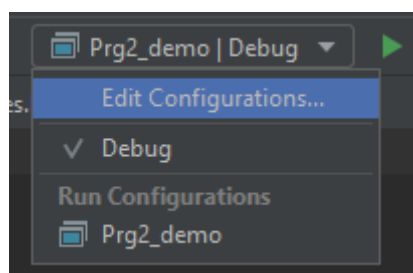


Figura 12 - Painel *"Edit Run/Debug Configurations"*

- 4) No painel de configuração, localize a seção *"Program arguments"* e adicione o arquivo *"distancias.csv"*. Certifique-se de também colocar o diretório em que está esse arquivo no seu computador em *"Working directory"*.

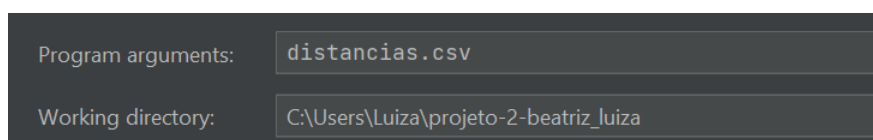
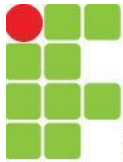


Figura 13 - Seções *"Program arguments"* e *"Working directory"* no Clion

- 5) Após adicionar o argumento do programa, clique em *"Apply"* para confirmar as mudanças.



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

- 6) Para compilar o programa, localize na barra de ferramentas superior o ícone “Build” da figura 14 ou pressione Ctrl+F9.



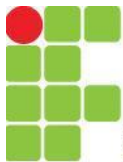
Figura 14 - Ícone para compilação do programa

- 7) Na mesma barra de ferramentas, encontre o ícone “Run” da figura 15. Clique neste ícone ou pressione Shift+F10 para executar o programa. O programa deve ser executado com o argumento que você especificou anteriormente.



Figura 15 - Ícone para execução do programa

- 6) O processo de execução será retomado adiante nesse manual.



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

Processo de execução:

“Esse tópico vale para ambas as opções apresentadas de execução (Linux e Clion), por mais que as figuras a seguir sejam da execução por Clion. A saída de dados é idêntica em relação ao conteúdo dos dados apresentados, porém a forma de apresentar eles pode ser um pouco diferente”.

Você será solicitado a fornecer informações de entrada. O primeiro pedido será para a cidade de partida, na qual você deve digitar o nome da cidade e pressionar a tecla "enter". Em seguida, o mesmo procedimento será repetido para a cidade de destino. Certifique-se de seguir essas etapas para que o programa possa executar corretamente e fornecer os resultados esperados.

```
Entre com a cidade de partida
Angelina
Entre com a cidade de destino
Alfredo Wagner
```

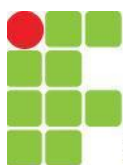
Figura 16 - Entrada de dados via Clion

Após o último procedimento, o programa apresenta os resultados esperados por você: a melhor rota entre as duas cidades escolhidas e a distância para percorrer todo esse trajeto.

```
-> A melhor rota |®:
Angelina    Rancho Queimado    Alfredo Wagner

-> A melhor distância |®:
62 km
```

Figura 17 - Saída de dados via Clion



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

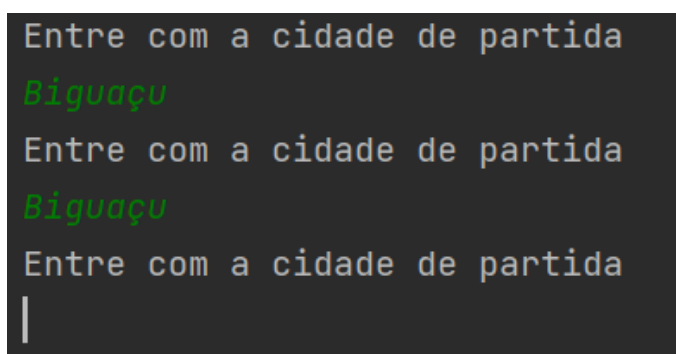
CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

Possíveis situações ao executar o programa:

“Aqui, são possíveis situações que você pode encontrar ao tentar executar o seu programa. Caso uma delas seja o seu caso, atente se a resolução proposta (quando houver)”.

1) O programa não apresentou a melhor rota entre cidades com acentuação em execução via Clion por sistema operacional Windows

Se as cidades selecionadas na entrada de dados pelo usuário tiverem caracteres de acentuação, como “~”, “^” ou até mesmo “ç”, o programa não realiza a entrada dos dados corretamente. A recomendação aqui, é tirar a acentuação das cidades de partida e destino diretamente no arquivo CSV contendo a relação de distâncias, assim o programa funciona.

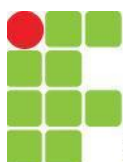


```
Entre com a cidade de partida
Biguaçu
Entre com a cidade de partida
Biguaçu
Entre com a cidade de partida
|
```

Figura 18 - Problema na entrada de dados ao escolher cidade com caracter especial via Clion em sistema operacional Windows

2) O programa apresentou uma saída de dados confusa via Clion por sistema operacional Windows

A saída de dados pode apresentar caracteres estranhos. Isso ocorre por conta da acentuação nas próprias palavras escolhidas na saída de dados, o Clion não imprime corretamente essa acentuação pelo sistema operacional Windows. Esse problema é



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

apresentado na figura 17, no próprio manual de execução Clion. Nessa situação, o programa continua funcionando e não há um procedimento de correção.

3) Mensagem de erro - Figura 19

Caso você deixe vazio o espaço dos argumentos em “*Program Arguments*” no Clion, será apresentada na tela a mensagem da figura 19. Para resolver, deve ser revisado se foi colocado corretamente o argumento (o arquivo CSV contendo as relações de distâncias entre cidades).

```
terminate called after throwing an instance of 'std::logic_error'  
what(): basic_string::_M_construct null not valid
```

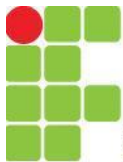
Figura 19 - Erro ao não inserir argumento no programa

4) Mensagem de erro - Figura 20

Caso você digite errado o nome do arquivo CSV (argumento do programa) em “*Program Arguments*” ou digite errado o endereço em que este arquivo está em “*Working Directory*”, será apresentada na tela a mensagem da figura 20. Para resolver, devem ser revisadas as escritas do nome e diretório desse arquivo CSV.

```
Erro : No such file or directory
```

Figura 20 - Erro ao digitar o diretório errado no programa



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

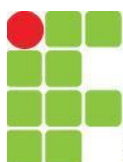
CONCLUSÃO

Portanto, como mencionado anteriormente, os resultados obtidos atenderam às expectativas dos objetivos propostos. O projeto foi excelente para colocar em prática os conhecimentos de estruturas de dados associativas, reforçar o estudo sobre as estruturas de dados lineares e exercitar a lógica de programação a partir do algoritmo de Dijkstra.

Embora o projeto tenha sido um sucesso, algumas melhorias e alterações podem ser feitas para torná-lo ainda melhor. Um exemplo é encontrar uma alternativa para permitir a entrada de dados com acentuação pelo Clion no Windows, uma vez que muitos usuários usam esse sistema operacional, que é projetado para ser fácil de usar.

Além disso, o programa pode ser aprimorado com uma análise mais abrangente da melhor rota, considerando pedágios e tráfego nas cidades presentes na base de dados. Essas informações adicionais podem ser extremamente úteis para ajudar as pessoas a se deslocarem com mais facilidade entre duas cidades diferentes.

Dessa forma, o projeto foi uma excelente oportunidade de aprender mais sobre o uso das estruturas de dados, colocando em prática o conhecimento obtido durante as aulas desde a unidade 1 até o momento atual em que a disciplina se encontra.



INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES - CÂMPUS SÃO JOSÉ

BIBLIOGRAFIA

Guia Github. Disponível em:

<<https://moodle.ifsc.edu.br/mod/page/view.php?id=713818>>. Acesso em: 29 abr. 2023.

Pseudo-Código Dijkstra. Disponível em:

<<https://moodle.ifsc.edu.br/mod/page/view.php?id=713871>>. Acesso em: 4 2023.

Algoritmo de Dijkstra. Disponível em:

<<https://www.youtube.com/watch?v=hsJBilAiZDY>>. Acesso em: 7 maio. 2023.

Simulador do Algoritmo Dijkstra. Disponível em:

<<https://cmps-people.ok.ubc.ca/ylucet/DS/Dijkstra.html>>. Acesso em: 7 maio. 2023.