Projeto I
O Problema do Caixeiro Viajante

SCC0202 – Algoritmos e Estruturas de Dados I

Profs. Marcelo G. Manzato e Rudinei Goularte
PAEs: Fernanda T. Marana e Wan Song Rocha

Data de Entrega: 13/10/2022

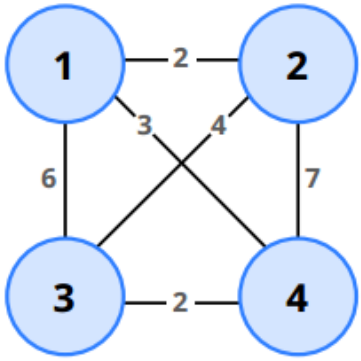
Este projeto deverá ser feito em grupos de 3 alunos. A entrega deve ser realizada via Run.Codes e via Tidia – veja detalhes sobre entrega abaixo.

Descrição do Problema

O **Problema do Caixeiro Viajante** (PCV) é um dos problemas clássicos na área de **Computação** e pode ser descrito da seguinte maneira. Suponha que você seja um representante comercial e precisa visitar *n* cidades diferentes para vender seu produto. Nesta viagem, **cada cidade deve ser visitada uma única vez** e, ao final do trajeto, você deve **retornar à cidade de origem**. Existem diferentes possibilidades de realizar este trajeto, mas o interesse está no trajeto de menor custo.

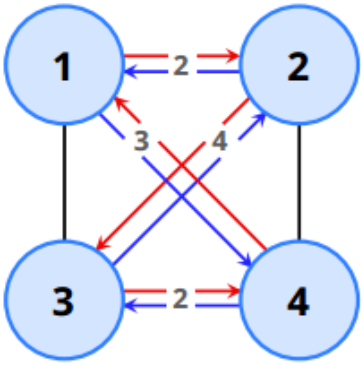
A definição de custo está diretamente relacionada ao propósito da aplicação. Em alguns casos, por exemplo, deseja-se identificar o trajeto com a menor quantidade possível de pedágios. Em outros, a quantidade é indiferente e não interfere na escolha, contanto que as rodovias sejam de melhor qualidade. Neste trabalho, o critério de **custo** que deverá ser considerado é o de **menor distância**. Além disso, por simplicidade, considere que entre quaisquer duas cidades *A* e *B pode haver, no máximo* um, caminho direto (i.e., pode não haver caminho direto ou, se houver, será 1 e somente 1).

Figura 1. O Problema do Caixeiro Viajante para 4 cidades



A Figura 1 ilustra o Problema do Caixeiro Viajante para 4 cidades, que foram esquematizadas em círculos. As conexões entre as duas cidades indicam a existência de um caminho direto, e em cada ligação existe um valor que representa a distância entre tais cidades. Dentre os possíveis trajetos que saem da cidade número 1, o de menor custo (custo 11) está destacado em vermelho na Figura 2 (1 − 2 − 3 − 4 − 1). Vale ressaltar que o caminho inverso (1 − 4 − 3 − 2 − 1), destacado em azul, é equivalente, e consequentemente, possui o mesmo custo.

Figura 2. Trajeto de Menor Distância para a cidade de número 1



Além do cenário aqui apresentado, diversos outros problemas reais possuem características e desafios similares ao PCV, tais como: roteamento de telecomunicações, definição de rotas de ônibus escolares e de entrega de refeições, problemas de logística em geral, entre outros.

Objetivo do Trabalho

O trabalho tem como principal objetivo desenvolver uma solução para o PCV, ou seja, dada uma cidade de origem, encontrar o trajeto de menor custo que passe por todas as cidades e por cada uma delas apenas uma vez, retornando à cidade de

origem.

O desenvolvimento da solução envolve as seguintes atividades principais:

1. Modelar uma solução para o PCV utilizando as Estruturas de Dados estudadas (até o momento da entrega) em aula.

2. Desenvolver um algoritmo que encontre o trajeto de menor custo usando uma abordagem de "força bruta" com solução exata. Nesta abordagem, deve-se avaliar todos os trajetos possíveis para encontrar o de menor custo → problema de análise combinatória.

3. Implementar a solução em Linguagem C, usando (obrigatoriamente) o conceito de TAD, modularização de código e Makefile para compilação e geração da aplicação executável.

4. Analisar a complexidade computacional da solução proposta usando notação *big O* e apresentar um gráfico de crescimento de tempo de execução para diferentes tamanhos de entrada (máximo de $n = 12$).

Essas atividades compõem as 3 partes do trabalho, descritas abaixo.

Parte I - Modelagem da Solução (25 % da nota final)

Serão avaliadas a coerência e relevância da solução e respectiva justificativa para os seguintes aspectos:

- Que Estrutura(s) de Dados foi(ram) escolhida(s) para modelar o problema? Justifique.
- Quais vantagens e limitações a(s) estrutura(s) escolhida(s) apresenta(m)?
- Explique claramente a lógica utilizada para resolver o problema.

Parte II - Implementação (45 % da nota final)

Esta parte avalia a estratégia de implementação utilizada. Por isso, serão considerados aspectos como organização e corretude do código, e documentação interna.

A manipulação das Estruturas de Dados deve ser baseada em TAD e o código deve ser escrito em Linguagem C (C99, flags: -std=c99 -Wall), compilado com gcc. Além disso, sua implementação deve atender os requisitos descritos a seguir.

Entrada do Programa: Serão disponibilizados 3 arquivos de teste que mostram como deverá ser a entrada do programa. Tais arquivos contêm conjuntos de entrada com 5, 10 e 12 cidades. A primeira linha dos arquivos indica a quantidade de cidades que o mesmo possui (5, 10 ou 12) e as linhas subsequentes incluem pares de cidades (*A, B*) com caminho direto entre as duas cidades, e a distância entre elas. A tabela

abaixo ilustra parcialmente os pares de cidades do arquivo de entrada para o exemplo apresentado na Figura 1, lembrando que pode não haver um caminho entre um par de cidades.

Cidade *A*	Cidade *B*	Distância
1	2	2
1	3	6
...	...	...
4	3	2

Exemplo do Arquivo de Entrada com 4 cidades e os pares indicados na tabela acima:
4
1
1 2 2
1 3 6
...
4 3 2

O programa deve ler um conjunto de entradas a partir do arquivo correspondente e armazenar os dados na(s) estrutura(s) implementada(s). **A cidade de origem corresponde à 2a linha do arquivo de entrada (1 no exemplo acima).**

As possibilidades de trajeto, dada a cidade de origem, deverão ser geradas por meio da abordagem de "força bruta" com solução exata, isto é, realizando todas combinações de trajeto entre cidades do conjunto de entrada. Dentre as possibilidades, o programa deve calcular e informar a de **menor** distância. Vale ressaltar que **todas as cidades devem ser visitadas**, **cada cidade só pode ser visitada uma única vez** e o **trajeto deve terminar na cidade de origem**.

Saída do Programa: O programa deve apresentar como saída, para a cidade de origem informada, o menor trajeto e o respectivo custo (ou seja, a menor distância encontrada). Seguindo o exemplo da Figura 1, a tabela abaixo ilustra o conteúdo da saída esperada:

Cidade Origem	Rota	Menor Distância
1	1 − 2 − 3 − 4 − 1	11

Vale ressaltar que os arquivos de teste fornecidos têm como objetivo apenas apresentar o formato esperado para a entrada de dados e direcionar testes iniciais. Na correção, seu programa será testado nos conjuntos disponibilizados e em

conjuntos diferentes. Portanto, **teste seu programa em conjuntos elaborados pelo próprio grupo**.

Parte III - Análise de Complexidade e Relatório (30 % da nota final)

A principal atividade desta parte é analisar a complexidade, em termos de tempo de execução, da implementação realizada. Para tanto:

1. Faça análise assintótica (usando a notação ***big O***) da complexidade computacional da solução proposta, avaliando a complexidade das operações básicas na(s) estrutura(s) de dados escolhida(s) e do algoritmo para encontrar o menor trajeto para o PCV.

2. Meça o tempo de execução para processamento de conjuntos de entrada de diferentes tamanhos (máximo de $n = 12$). Contabilize apenas o tempo para encontrar o menor trajeto a partir da cidade de origem informada, ou seja, desconsidere tempos de leitura/escrita em arquivos.

3. Elabore um gráfico que apresenta as medidas de tempo de execução em função do tamanho do conjunto de entrada e explique o resultado (ou seja, relacione o resultado apresentado no gráfico à complexidade computacional obtida no item 1).

EXTRA DE 2.0 PONTOS - Não Obrigatório: Implemente uma solução mais eficiente para o PCV, que reduza significativamente a complexidade computacional do algoritmo baseado em força bruta. Explique a sua solução e apresente o cálculo de complexidade (*big O* e gráficos de tempo de execução), mostrando que a abordagem adotada trouxe melhorias. **Importante**: a implementação do algoritmo de **força bruta é obrigatória**, mesmo que o grupo implemente uma segunda solução mais eficiente.

Entrega do Trabalho

Faça um relatório de no máximo 5 páginas com todas as informações e requisitos apresentados nas Partes I, II e III. Sobre a Parte II, inclua no relatório apenas a informação necessária para compilação e execução de seu programa, indicando quais são os arquivos do(s) TAD(s), do programa principal, e demais arquivos criados. Não inclua código no relatório.

Crie um arquivo Nome_Grupo_Relatorio.zip (Nome_Grupo deve ser substituído pelo nome do grupo) contendo: o relatório (em pdf). Esse arquivo deve ser entregue no **Escaninho** do Tidia até **13/10/2022**, 23:59h. Apenas um integrante do grupo deve postar o arquivo em seu escaninho Tidia.

Crie um segundo arquivo zip chamado Nome_Grupo_Codigo.zip (Nome_Gupo deve ser substituído pelo nome do grupo) contendo: o makefile e **todo código fonte** necessário para compilar e executar o programa, incluindo arquivos*.c* (*main* e outros) e arquivos*.h*. Não adicionar arquivos executáveis ou arquivos objeto. Esse arquivo deve ser submetido no RunCodes. Apenas um aluno do grupo deverá fazer a submissão. O trabalho deve ser entregue (ambos arquivos zip) no **RunCodes** também até **13/10/2022**, 23:59h.

Atenção: Trabalhos submetidos em datas posteriores à exigida terão 1,0 ponto de desconto por dia de atraso. Apenas um integrante do grupo precisa submeter o trabalho no Tidia.

Bom trabalho a todos!