UNIVERSIDADE FEDERAL DE OURO PRETO
IVES HENRIQUE SENIBALDE DE OLIVEIRA
YANN EDUARDO DE SOUZA SILVA

Professor: Pedro Henrique Lopes Silva
Disciplina: BCC 202

Problema do Caixeiro Viajante com implementação de Listas Encadeadas

Ouro Preto
2023

1. Introdução

Considere um contexto onde um caixeiro viajante tem que passar por um número *n* de cidades diferentes, iniciando e finalizando sua jornada na primeira cidade. Com um conjunto de cidades e as distâncias entre todas as combinações possíveis de cidades disponíveis, a sequência de visitas não tem impacto. O desafio do caixeiro viajante é descobrir a rota que garanta a menor distância total da viagem. Na Figura 1, são exibidas as cidades e as distâncias entre elas.

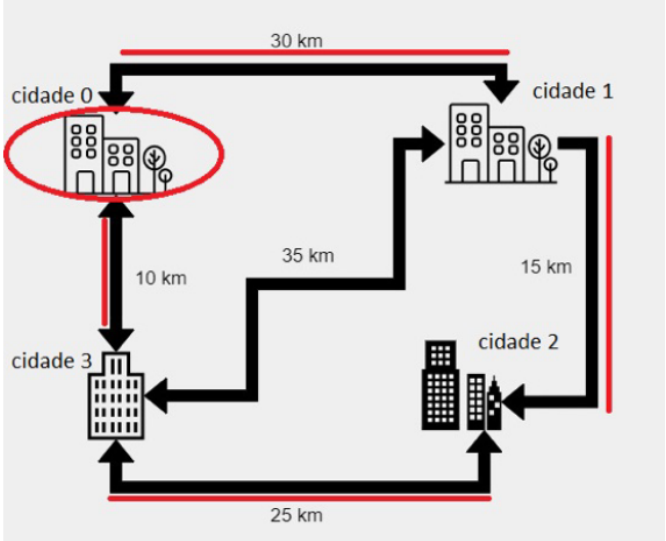


Figura 1: Cidades e a distância entre elas.

A Figura 2 apresenta a menor rota, partindo da cidade 0, que passa por todas as cidades da Figura 1.

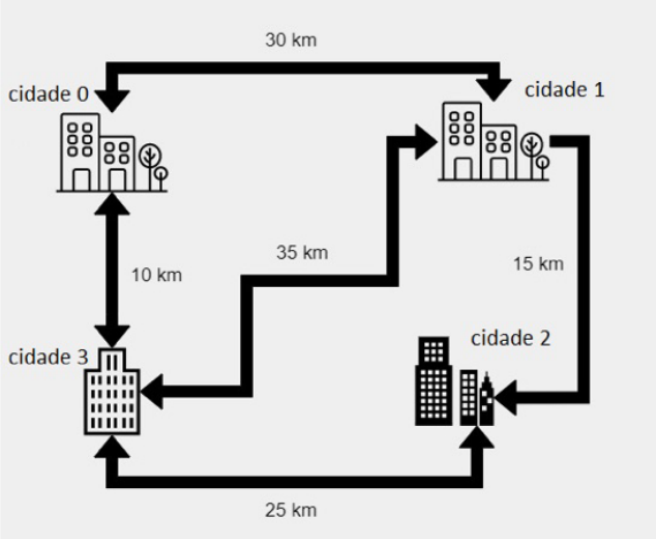


Figura 2: Menos rota entre as cidades.

Consequentemente, o propósito deste trabalho é desenvolver um algoritmo na linguagem de programação C capaz de calcular, com base no número de cidades fornecido e suas distâncias entre si, a rota e a distância total que o caixeiro viajante deve percorrer.

2. Implementação

O código desenvolvido contém um total de 12 funções, tendo que duas delas são as funções principais “encontraCaminho” e “bubbleSort”, e o restante são funções auxiliares.

A implementação do algoritmo adotou a estratégia de força bruta, a qual envolve o cálculo de todas as rotas viáveis possíveis e a medição da distância total percorrida em cada uma delas. Dessa forma, a análise das diferentes rotas conduz à solução do problema, identificando o caminho mais curto.

Para calcular o número de rotas usando a função *R(n)*, na qual *R* representa as rotas possíveis e *n* é o número de cidades, aplicam-se princípios da análise combinatória. Desse modo, para *n* cidades, o número total de escolhas é expresso pela permutação de *n-1* elementos, o que gera a função: ***R(n) = (n - 1)!***

Segue uma lista dos protótipos das funções utilizadas e os objetivos das mesmas:

1. alocarGrafo - aloca a estrutura GrafoPonderado e a lista.
2. alocarLista - aloca um vetor de lista.
3. listaEhVazia - verifica se a lista é vazia.
4. desalocarLista - desaloca a lista.
5. desalocaGrafo - desaloca o grafo.
6. insereLista - cria uma nova célula que vai conter as informações.
7. leGrafo - faz a leitura dos dados do arquivo.
8. bubbleSort - ordena a lista.
9. ordenaEImprimeLista - imprime a lista ordenada.
10. encontraCaminho - retorna o menor caminho.
11. imprimeCaminho - imprime o menor caminho e a distância percorrida.
12. copiaCaminho - faz a cópia do caminho.

3. Testes

Cinco testes que estão armazenados na pasta "tests" foram executados. Cada teste começa com uma linha que indica a quantidade de cidades a serem visitadas. Em seguida, são fornecidas, em cada linha, informações sobre cada cidade: suas conexões com outras cidades e as distâncias entre elas. Para cada teste, o resultado esperado é a lista ordenada de adjacências de cada vértice, organizada de forma crescente conforme as distâncias. Além disso, são apresentados o melhor trajeto e a menor distância a percorrer.

Assim sendo, abaixo estará listado as entradas dos cinco testes feitos e suas

respectivas saídas esperadas:

Teste 1 (entrada):

4
0 0 0
0 1 10
0 2 15
0 3 20
1 0 10
1 1 0
1 2 35
1 3 25
2 0 15
2 1 35
2 2 0
2 3 30
3 0 20
3 1 25
3 2 30
3 3 0

Saída:

Adjacencias do vertice 0: (0, 0) -> (1, 10) -> (2, 15) -> (3, 20) -> NULL
Adjacencias do vertice 1: (1, 0) -> (0, 10) -> (3, 25) -> (2, 35) -> NULL
Adjacencias do vertice 2: (2, 0) -> (0, 15) -> (3, 30) -> (1, 35) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 20) -> (1, 25) -> (2, 30) -> NULL
Melhor caminho: 0 1 3 2 0
Melhor distancia: 80

Teste 2 (entrada):

6
0 0 0
0 1 1
0 2 2
0 3 1
0 4 1
0 5 2
1 0 1
1 1 0
1 2 7
1 3 1
1 4 4
1 5 3
2 0 2
2 1 7
2 2 0
2 3 3
2 4 1
2 5 1
3 0 1
3 1 1
3 2 3
3 3 0
3 4 8
3 5 1
4 0 1
4 1 2
4 2 1
4 3 8
4 4 0
4 5 1
5 0 2
5 1 3
5 2 1
5 3 1
5 4 1
5 5 0

Saída:

Adjacencias do vertice 0: (0, 0) -> (1, 1) -> (3, 1) -> (4, 1) -> (2, 2) -> (5, 2) -> NULL
Adjacencias do vertice 1: (1, 0) -> (0, 1) -> (3, 1) -> (5, 3) -> (4, 4) -> (2, 7) -> NULL
Adjacencias do vertice 2: (2, 0) -> (4, 1) -> (5, 1) -> (0, 2) -> (3, 3) -> (1, 7) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 1) -> (1, 1) -> (5, 1) -> (2, 3) -> (4, 8) -> NULL
Adjacencias do vertice 4: (4, 0) -> (0, 1) -> (2, 1) -> (5, 1) -> (1, 2) -> (3, 8) -> NULL
Adjacencias do vertice 5: (5, 0) -> (2, 1) -> (3, 1) -> (4, 1) -> (0, 2) -> (1, 3) -> NULL
Melhor caminho: 0 1 3 5 2 4 0
Melhor distancia: 6
Teste 3 (entrada):

4
0 0 0
0 1 1
0 2 1
0 3 3
1 0 1
1 1 0
1 2 4
1 3 5
2 0 1
2 1 4
2 2 0
2 3 6
3 0 3
3 1 5
3 2 6
3 3 0

Saída:

Adjacencias do vertice 0: (0, 0) -> (1, 1) -> (2, 1) -> (3, 3) -> NULL
Adjacencias do vertice 1: (1, 0) -> (0, 1) -> (2, 4) -> (3, 5) -> NULL
Adjacencias do vertice 2: (2, 0) -> (0, 1) -> (1, 4) -> (3, 6) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 3) -> (1, 5) -> (2, 6) -> NULL
Melhor caminho: 0 1 3 2 0
Melhor distancia: 13

Teste 4 (entrada):

5
0 0 0
0 1 2
0 2 0
0 3 3
0 4 6
1 0 2
1 1 0
1 2 4
1 3 3
1 4 0
2 0 0
2 1 4
2 2 0
2 3 7
2 4 3
3 0 3
3 1 3
3 2 7
3 3 0
3 4 3
4 0 6
4 1 0
4 2 3
4 3 3
4 4 0

Saída:

Adjacencias do vertice 0: (0, 0) -> (2, 0) -> (1, 2) -> (3, 3) -> (4, 6) -> NULL
Adjacencias do vertice 1: (1, 0) -> (4, 0) -> (0, 2) -> (3, 3) -> (2, 4) -> NULL
Adjacencias do vertice 2: (0, 0) -> (2, 0) -> (4, 3) -> (1, 4) -> (3, 7) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 3) -> (1, 3) -> (4, 3) -> (2, 7) -> NULL
Adjacencias do vertice 4: (1, 0) -> (4, 0) -> (2, 3) -> (3, 3) -> (0, 6) -> NULL
Melhor caminho: 0 1 2 4 3 0
Melhor distancia: 15

Teste 5 (entrada):

```
5
0 0 0
0 1 5
0 2 10
0 3 0
0 4 1
1 0 5
1 1 0
1 2 0
1 3 10
1 4 1
2 0 10
2 1 0
2 2 0
2 3 2
2 4 1
3 0 0
3 1 10
3 2 2
3 3 0
3 4 1
4 0 1
4 1 1
4 2 1
4 3 1
4 4 0
```

Saída:

```
Adjacencias do vertice 0: (0, 0) -> (3, 0) -> (4, 1) -> (1, 5) -> (2, 10) -> NULL
Adjacencias do vertice 1: (1, 0) -> (2, 0) -> (4, 1) -> (0, 5) -> (3, 10) -> NULL
Adjacencias do vertice 2: (1, 0) -> (2, 0) -> (4, 1) -> (3, 2) -> (0, 10) -> NULL
Adjacencias do vertice 3: (0, 0) -> (3, 0) -> (4, 1) -> (2, 2) -> (1, 10) -> NULL
Adjacencias do vertice 4: (4, 0) -> (0, 1) -> (1, 1) -> (2, 1) -> (3, 1) -> NULL
Melhor caminho: 0 1 3 2 4 0
Melhor distancia: 19
```

4. Análise

O programa foi analisado usando a ferramenta valgrind, a fim de identificar e verificar possíveis problemas de vazamento de memória. As imagens abaixo mostram os resultados das análises feitas pela ferramenta, apontam que não foram identificados vazamentos de memória nas alocações, conforme indicado nos textos gerados. A última imagem apresenta o resultado dos testes feitos pelo corretor de testes.

Saída do Teste 1:

```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ valgrind ./exe < tests/teste1.in
==42900== Memcheck, a memory error detector
==42900== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==42900== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==42900== Command: ./exe
==42900==

Adjacencias do vertice 0: (0, 0) -> (1, 10) -> (2, 15) -> (3, 20) -> NULL
Adjacencias do vertice 1: (1, 0) -> (0, 10) -> (3, 25) -> (2, 35) -> NULL
Adjacencias do vertice 2: (2, 0) -> (0, 15) -> (3, 30) -> (1, 35) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 20) -> (1, 25) -> (2, 30) -> NULL

Melhor caminho: 0 1 3 2 0
Melhor distancia: 80
==42900==
==42900== HEAP SUMMARY:
==42900==     in use at exit: 0 bytes in 0 blocks
==42900==   total heap usage: 27 allocs, 27 frees, 5,568 bytes allocated
==42900==
==42900== All heap blocks were freed -- no leaks are possible
==42900==
==42900== For lists of detected and suppressed errors, rerun with: -s
==42900== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Tempo de execução do Teste 1:

```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ time ./exe < tests/teste1.in

Adjacencias do vertice 0: (0, 0) -> (1, 10) -> (2, 15) -> (3, 20) -> NULL
Adjacencias do vertice 1: (1, 0) -> (0, 10) -> (3, 25) -> (2, 35) -> NULL
Adjacencias do vertice 2: (2, 0) -> (0, 15) -> (3, 30) -> (1, 35) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 20) -> (1, 25) -> (2, 30) -> NULL

Melhor caminho: 0 1 3 2 0
Melhor distancia: 80

real    0m0,001s
user    0m0,001s
sys     0m0,000s
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Saída do Teste 2:

```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ valgrind ./exe < tests/teste2.in
==42959== Memcheck, a memory error detector
==42959== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==42959== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==42959== Command: ./exe
==42959==

Adjacencias do vertice 0: (0, 0) -> (1, 1) -> (3, 1) -> (4, 1) -> (2, 2) -> (5, 2) -> NULL
Adjacencias do vertice 1: (1, 0) -> (0, 1) -> (3, 1) -> (5, 3) -> (4, 4) -> (2, 7) -> NULL
Adjacencias do vertice 2: (2, 0) -> (4, 1) -> (5, 1) -> (0, 2) -> (3, 3) -> (1, 7) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 1) -> (1, 1) -> (5, 1) -> (2, 3) -> (4, 8) -> NULL
Adjacencias do vertice 4: (4, 0) -> (0, 1) -> (2, 1) -> (5, 1) -> (1, 2) -> (3, 8) -> NULL
Adjacencias do vertice 5: (5, 0) -> (2, 1) -> (3, 1) -> (4, 1) -> (0, 2) -> (1, 3) -> NULL

Melhor caminho: 0 1 3 5 2 4 0
Melhor distancia: 6
==42959==
==42959== HEAP SUMMARY:
==42959==     in use at exit: 0 bytes in 0 blocks
==42959==   total heap usage: 49 allocs, 49 frees, 5,976 bytes allocated
==42959==
==42959== All heap blocks were freed -- no leaks are possible
==42959==
==42959== For lists of detected and suppressed errors, rerun with: -s
==42959== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Tempo de execução do Teste 2:

```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ time ./exe < tests/teste2.in

Adjacencias do vertice 0: (0, 0) -> (1, 1) -> (3, 1) -> (4, 1) -> (2, 2) -> (5, 2) -> NULL
Adjacencias do vertice 1: (1, 0) -> (0, 1) -> (3, 1) -> (5, 3) -> (4, 4) -> (2, 7) -> NULL
Adjacencias do vertice 2: (2, 0) -> (4, 1) -> (5, 1) -> (0, 2) -> (3, 3) -> (1, 7) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 1) -> (1, 1) -> (5, 1) -> (2, 3) -> (4, 8) -> NULL
Adjacencias do vertice 4: (4, 0) -> (0, 1) -> (2, 1) -> (5, 1) -> (1, 2) -> (3, 8) -> NULL
Adjacencias do vertice 5: (5, 0) -> (2, 1) -> (3, 1) -> (4, 1) -> (0, 2) -> (1, 3) -> NULL

Melhor caminho: 0 1 3 5 2 4 0
Melhor distancia: 6

real    0m0,001s
user    0m0,001s
sys     0m0,000s
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Saída do Teste 3:

```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ valgrind ./exe < tests/teste3.in
==43008== Memcheck, a memory error detector
==43008== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==43008== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==43008== Command: ./exe
==43008==

Adjacencias do vertice 0: (0, 0) -> (1, 1) -> (2, 1) -> (3, 3) -> NULL
Adjacencias do vertice 1: (1, 0) -> (0, 1) -> (2, 4) -> (3, 5) -> NULL
Adjacencias do vertice 2: (2, 0) -> (0, 1) -> (1, 4) -> (3, 6) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 3) -> (1, 5) -> (2, 6) -> NULL

Melhor caminho: 0 1 3 2 0
Melhor distancia: 13
==43008==
==43008== HEAP SUMMARY:
==43008==     in use at exit: 0 bytes in 0 blocks
==43008==   total heap usage: 27 allocs, 27 frees, 5,568 bytes allocated
==43008==
==43008== All heap blocks were freed -- no leaks are possible
==43008==
==43008== For lists of detected and suppressed errors, rerun with: -s
==43008== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Tempo de execução do Teste 3:

```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ time ./exe < tests/teste3.in

Adjacencias do vertice 0: (0, 0) -> (1, 1) -> (2, 1) -> (3, 3) -> NULL
Adjacencias do vertice 1: (1, 0) -> (0, 1) -> (2, 4) -> (3, 5) -> NULL
Adjacencias do vertice 2: (2, 0) -> (0, 1) -> (1, 4) -> (3, 6) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 3) -> (1, 5) -> (2, 6) -> NULL

Melhor caminho: 0 1 3 2 0
Melhor distancia: 13

real    0m0,001s
user    0m0,001s
sys     0m0,000s
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Saída do Teste 4:

```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ valgrind ./exe < tests/teste4.in
==43106== Memcheck, a memory error detector
==43106== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==43106== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==43106== Command: ./exe
==43106==

Adjacencias do vertice 0: (0, 0) -> (2, 0) -> (1, 2) -> (3, 3) -> (4, 6) -> NULL
Adjacencias do vertice 1: (1, 0) -> (4, 0) -> (0, 2) -> (3, 3) -> (2, 4) -> NULL
Adjacencias do vertice 2: (0, 0) -> (2, 0) -> (4, 3) -> (1, 4) -> (3, 7) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 3) -> (1, 3) -> (4, 3) -> (2, 7) -> NULL
Adjacencias do vertice 4: (1, 0) -> (4, 0) -> (2, 3) -> (3, 3) -> (0, 6) -> NULL

Melhor caminho: 0 1 2 4 3 0
Melhor distancia: 15
==43106==
==43106== HEAP SUMMARY:
==43106==     in use at exit: 0 bytes in 0 blocks
==43106==   total heap usage: 37 allocs, 37 frees, 5,756 bytes allocated
==43106==
==43106== All heap blocks were freed -- no leaks are possible
==43106==
==43106== For lists of detected and suppressed errors, rerun with: -s
==43106== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Tempo de execução do Teste 4:

```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ time ./exe < tests/teste4.in

Adjacencias do vertice 0: (0, 0) -> (2, 0) -> (1, 2) -> (3, 3) -> (4, 6) -> NULL
Adjacencias do vertice 1: (1, 0) -> (4, 0) -> (0, 2) -> (3, 3) -> (2, 4) -> NULL
Adjacencias do vertice 2: (0, 0) -> (2, 0) -> (4, 3) -> (1, 4) -> (3, 7) -> NULL
Adjacencias do vertice 3: (3, 0) -> (0, 3) -> (1, 3) -> (4, 3) -> (2, 7) -> NULL
Adjacencias do vertice 4: (1, 0) -> (4, 0) -> (2, 3) -> (3, 3) -> (0, 6) -> NULL

Melhor caminho: 0 1 2 4 3 0
Melhor distancia: 15

real    0m0,001s
user    0m0,001s
sys     0m0,000s
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Saída do Teste 5:

```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ valgrind ./exe < tests/teste5.in
==43132== Memcheck, a memory error detector
==43132== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==43132== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==43132== Command: ./exe
==43132==

Adjacencias do vertice 0: (0, 0) -> (3, 0) -> (4, 1) -> (1, 5) -> (2, 10) -> NULL
Adjacencias do vertice 1: (1, 0) -> (2, 0) -> (4, 1) -> (0, 5) -> (3, 10) -> NULL
Adjacencias do vertice 2: (1, 0) -> (2, 0) -> (4, 1) -> (3, 2) -> (0, 10) -> NULL
Adjacencias do vertice 3: (0, 0) -> (3, 0) -> (4, 1) -> (2, 2) -> (1, 10) -> NULL
Adjacencias do vertice 4: (4, 0) -> (0, 1) -> (1, 1) -> (2, 1) -> (3, 1) -> NULL

Melhor caminho: 0 1 3 2 4 0
Melhor distancia: 19
==43132==
==43132== HEAP SUMMARY:
==43132==     in use at exit: 0 bytes in 0 blocks
==43132==   total heap usage: 37 allocs, 37 frees, 5,756 bytes allocated
==43132==
==43132== All heap blocks were freed -- no leaks are possible
==43132==
==43132== For lists of detected and suppressed errors, rerun with: -s
==43132== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Tempo de execução do Teste 5:

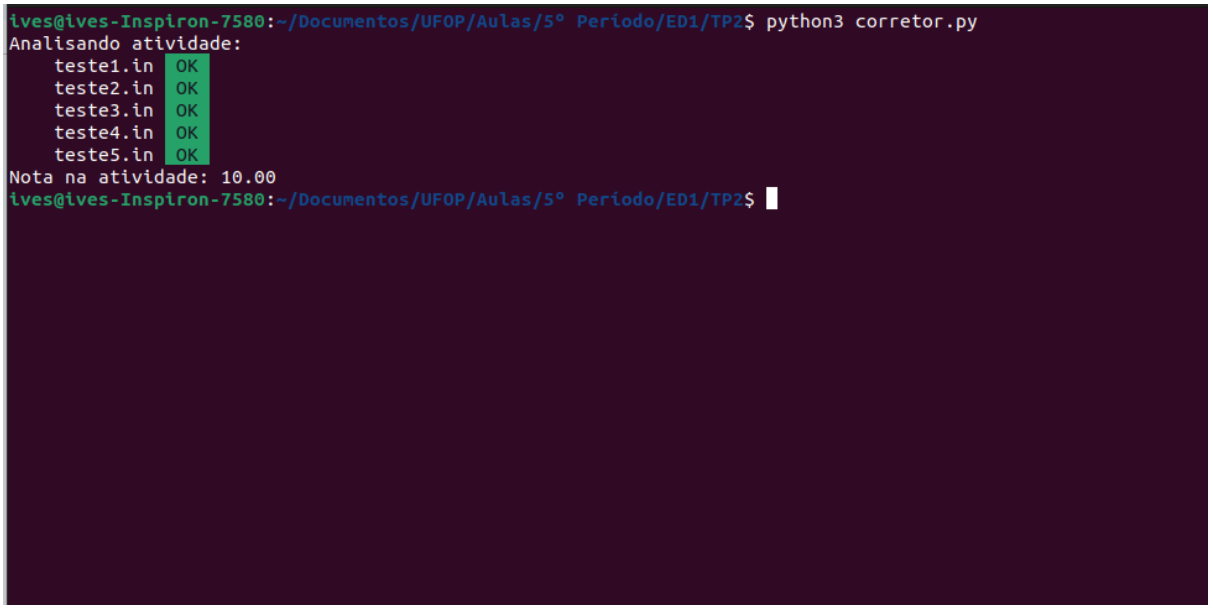
```
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$ time ./exe < tests/teste5.in

Adjacencias do vertice 0: (0, 0) -> (3, 0) -> (4, 1) -> (1, 5) -> (2, 10) -> NULL
Adjacencias do vertice 1: (1, 0) -> (2, 0) -> (4, 1) -> (0, 5) -> (3, 10) -> NULL
Adjacencias do vertice 2: (1, 0) -> (2, 0) -> (4, 1) -> (3, 2) -> (0, 10) -> NULL
Adjacencias do vertice 3: (0, 0) -> (3, 0) -> (4, 1) -> (2, 2) -> (1, 10) -> NULL
Adjacencias do vertice 4: (4, 0) -> (0, 1) -> (1, 1) -> (2, 1) -> (3, 1) -> NULL

Melhor caminho: 0 1 3 2 4 0
Melhor distancia: 19

real    0m0,001s
user    0m0,001s
sys     0m0,000s
ives@ives-Inspiron-7580:~/Documentos/UFOP/Aulas/5º Período/ED1/TP2$
```

Saída do Corretor:



5. Conclusão

Este estudo abordou o desafiador Problema do Caixeiro Viajante, cujo cerne reside na busca pela rota mais curta entre um conjunto de cidades conhecidas e suas distâncias respectivas. Apesar de ser um problema de otimização matemática, até o momento não existe uma fórmula definitiva para prever a menor rota para o viajante.

O objetivo central deste trabalho foi desenvolver um algoritmo capaz de calcular as distâncias e determinar a melhor rota entre todas as possíveis. Os resultados obtidos foram promissores, uma vez que o algoritmo produziu os resultados esperados nos cinco testes realizados, sem apresentar vazamentos de memória e com um tempo de execução inferior a 1 segundo em todos os casos.

É válido ressaltar que os maiores desafios enfrentados durante o desenvolvimento estiveram relacionados à criação de uma função recursiva para computar todas as diferentes rotas possíveis do viajante, bem como à implementação da ordenação de um vetor de listas de adjacências contendo as distâncias entre todas as combinações de cidades.

Este estudo não apenas reforçou a complexidade intrínseca do Problema do Caixeiro Viajante, mas também destacou a importância da eficiência algorítmica na resolução de desafios computacionais. Os resultados obtidos contribuem para a compreensão e aplicação de métodos de otimização em problemas práticos, embora continuem a evidenciar a necessidade de estratégias criativas para enfrentar problemas de grande escala de forma mais eficaz.

6. Bibliografia

- Aula 096 - Ordenando uma lista encadeada: https://youtu.be/vledOKDREnQ?si=MglRy70gXastOVE3