

Exercício de Programação EP1 – PCS3438

Buscas

1. O problema

Para este EP usaremos o **problema do caixeiro viajante** (TSP - Travel Salesman Problem). Nesse problema, o objetivo é determinar a menor rota para percorrer todas as cidades uma única vez e retornar à cidade origem. Para definir o TSP, tem-se uma matriz de adjacências quadrada e simétrica contendo as distâncias entre cidades. A tabela 1 mostra um exemplo com 4 cidades.

Tabela 1 – Exemplo de entrada do TSP com 4 cidades.

	Cidade1	Cidade2	Cidade3	Cidade4
Cidade1	0	38	Inf	50
Cidade2	38	0	88	71
Cidade3	Inf	88	0	29
Cidade4	50	71	29	0

Todas as distâncias são definidas como 0, valores positivos ou Inf (infinito). Uma distância 0 indica que não há custos para ir de uma cidade para ela mesma, porém note que pela definição do problema uma cidade não pode ser visitada mais de uma vez. Por outro lado, a distância Inf indica que não é possível ir de uma cidade para outra diretamente, por exemplo, não é possível ir de Cidade1 para a Cidade3. Como a matriz de adjacências é simétrica, a distância entre Cidade1 e Cidade4 é igual à distância entre Cidade4 e Cidade1.

2. As entregas

O objetivo desse EP é criar uma função na linguagem R que resolva o problema. Serão 2 arquivos a serem entregues: um relatório em pdf descrevendo a solução e um arquivo compactado zip com os scripts R desenvolvidos. Você deve entregar no eDisciplinas os seguintes arquivos:

Nome do arquivo de script R: EP1_<NUSP>_scripts Nome do relatório: EP1_<NUSP>_relatorio
--

Dentro do script R EP1_<NUSP>_scripts é obrigatório a existência de uma função EP1_<NUSP> que resolve o TSP. Ou seja, para solucionar o TSP executaremos a função EP1_<NUSP>(arqEntrada, arqSaida) no console R. As entradas da função serão explicadas na próxima seção.

Enviar todos os arquivos necessários para a execução dentro do arquivo compactado zip EP1_<NUSP>_scripts.zip . Entre eles, obrigatoriamente o script EP1_NUSP.R

3. O Exercício

Para a resolução do EP1, será permitido criar um solucionador ou utilizar uma solução pronta disponível em um pacote R do CRAN. A função `EP1_<NUSP>` deve resolver o TSP e retornar a rota com seu respectivo custo e o tempo de execução (tempo de busca pela solução). Você deve minimizar ao máximo possível tanto o custo da rota quanto o tempo de busca.

a. Entrada da função:

`EP1_<NUSP>` receberá 2 strings: a primeira contém o caminho para um arquivo .csv descrevendo a matriz de adjacências; a segunda string, o caminho para um arquivo .csv de saída. Exemplo:

```
EP1_6426934("C:/PCS3834/entrada.csv", "C:/PCS3834/saida.csv")
```

`entrada.csv` é o arquivo com a matriz de adjacências e `C:/PCS3834/` é o local onde o arquivo se encontra.

b. Arquivo de Entrada:

A primeira linha do arquivo de entrada contém o nome de cada vértice, separados por vírgulas. As linhas seguintes possuem o nome do vértice e as suas distâncias para cada um dos vértices (incluindo ele próprio), separadas por vírgulas. Note que `Inf` indica uma distância infinita, ou seja, não existe uma rota direta entre `Cidade1` e `Cidade3`. No exemplo da tabela 1, tem-se o seguinte arquivo de entrada:

```
"Cidade1","Cidade2","Cidade3","Cidade4"
"Cidade1",0,38,Inf,50
"Cidade2",38,0,88,71
"Cidade3",Inf,88,0,29
"Cidade4",50,71,29,0
```

c. Saídas:

Ao terminar a execução, `EP1_<NUSP>` deverá imprimir em tela o custo da rota resultante, o tempo total de execução e os vértices da rota, além de imprimir estas informações no arquivo de saída.

```
Console Terminal x
~/
> source('D:/USP/Monitoria/PCS3834/TSP_6426934.R')
> TSP_6426934('D:/USP/Monitoria/PCS3834/EP1/tsp.csv', 'D:/USP/Monitoria/PCS3834/EP1/saida.csv')
Custo: 205
Tempo de Execucao: 0.3305011
Rota: 1 4 3 2
> |
```

O arquivo de saída conterá uma única linha com `<Custo>`, `<Tempo de Execução>`, `<Rota>`. Exemplo:

205,0.233767032623291,1,4,3,2

4. O relatório

O relatório deverá:

1. Explicar a busca utilizada nos termos vistos em aula (apresente o pseudo-código).
2. Descrever como a busca funciona com detalhes o suficiente para um leitor entender o seu funcionamento.
3. Justificar a heurística escolhida, explicando quais são suas vantagens e desvantagens e o motivo da escolha.
4. Se for necessário instalar e carregar algum pacote adicional, relacionar os pacotes R necessários para a execução da função.

5. A execução

Durante uma aula a ser definida futuramente, todos os scripts serão executados duas vezes. Na primeira o TSP terá poucos vértices e, na segunda, vários vértices. Em ambos os casos os tempos de execução e o custo serão anotados para a definição das notas.

6. A nota

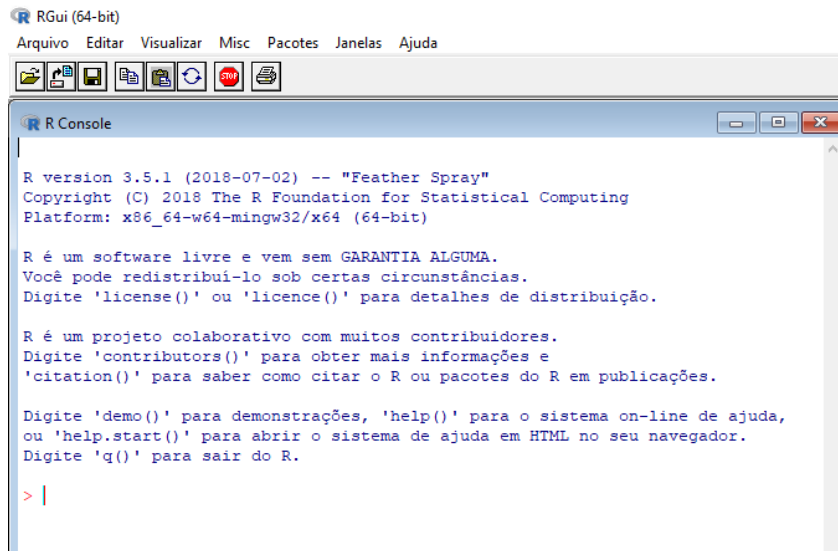
A nota desse EP1 é composta por 4 componentes:

Nota = 0,1*tempo_execução + 0,1*custo_rota + 0,2*código + 0,6*relatório

- Tempo de execução: as notas serão proporcionais ao tempo de execução, em que o mais rápido recebe 10, o segundo tem um desconto proporcional até o último colocado que recebe 0.
- Custo da rota: as notas serão proporcionais ao custo (mínimo custo recebe 10, o máximo custo recebe 0 e os outros serão valores proporcionais).
- Código: serão avaliados a clareza, os comentários, etc

Linguagem R

O R está disponível para download em <https://www.r-project.org/> . Após a instalação, é possível executar scripts no console R.



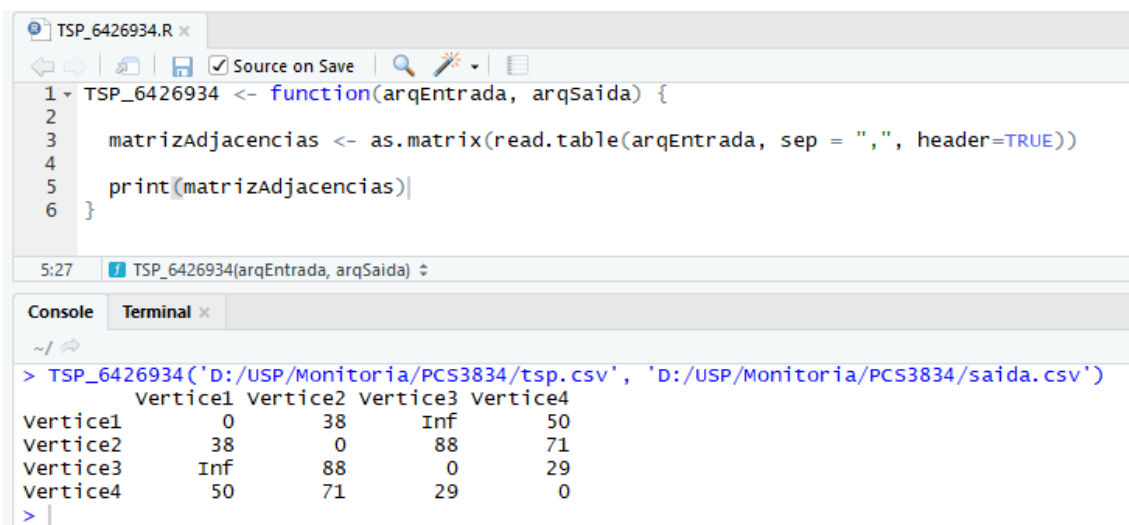
Existem IDEs que facilitam a codificação e depuração de script R . Exemplos: RStudio, Eclipse, Rattle.

- Como instalar e usar pacotes:

Para instalar: `install.packages("<nome_pacote>")`

Para carregar pacotes: `library(<nome_pacote>)`

- Como carregar funções em outros arquivos:
`source("<local_e_nome_do_arquivo.R>")`
- Como ler a matriz de entrada:
`variavel <- as.matrix(read.table("<local_e_nome_do_arquivo>", sep = ",", header=TRUE))`



- Como escrever em um arquivo de saída:

```
write.table(variavel, file = "<local_e_nome_do_arquivo>", sep = ",", row.names = FALSE, col.names = FALSE)
```

- Como contabilizar tempo de execução:

```
tempo_inicial <- Sys.time()
```

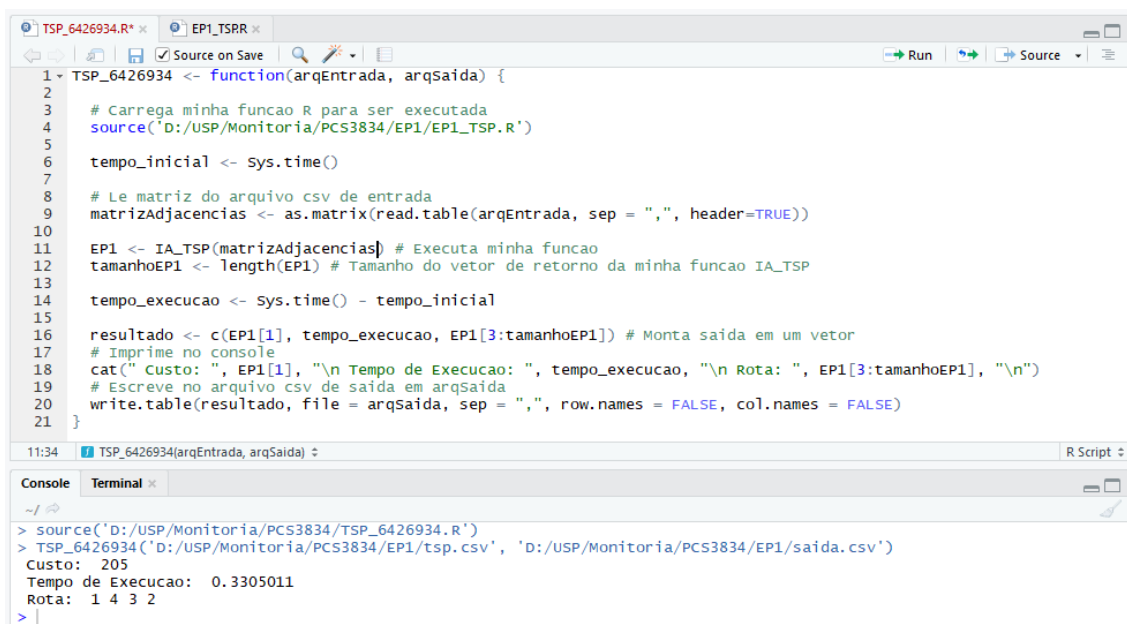
...

```
tempo_execucao <- Sys.time()-tempo_inicial
```

- Alguns comandos para quem não conhece R:

- matrix(0,n,n) – cria uma matrix nxn só com 0.
- c(elemento1,...elementoN) – cria um vetor contendo [elemento1,...,elementoN].
- 1:5 – cria um vetor com os elementos 1,2,3,4,5
- variavel1 <- Inf – a variavel variavel1 recebe o valor infinito
- mat[,1] –primeira coluna da matriz mat
- mat[1,] – primeira linha da matriz mat
- sort(variavel1) – retorna os elementos de variavel1 ordenados
- # Comentario

- Exemplo:



```

1 TSP_6426934 <- function(arqEntrada, arqSaida) {
2
3   # Carrega minha funcao R para ser executada
4   source('D:/USP/Monitoria/PCS3834/EP1/EP1_TSP.R')
5
6   tempo_inicial <- Sys.time()
7
8   # Le matriz do arquivo csv de entrada
9   matrizAdjacencias <- as.matrix(read.table(arqEntrada, sep = ",", header=TRUE))
10
11   EP1 <- IA_TSP(matrizAdjacencias) # Executa minha funcao
12   tamanhoEP1 <- length(EP1) # Tamanho do vetor de retorno da minha funcao IA_TSP
13
14   tempo_execucao <- Sys.time() - tempo_inicial
15
16   resultado <- c(EP1[1], tempo_execucao, EP1[3:tamanhoEP1]) # Monta saida em um vetor
17   # Imprime no console
18   cat(" Custo: ", EP1[1], "\n Tempo de Execucao: ", tempo_execucao, "\n Rota: ", EP1[3:tamanhoEP1], "\n")
19   # Escreve no arquivo csv de saida em arqSaida
20   write.table(resultado, file = arqSaida, sep = ",", row.names = FALSE, col.names = FALSE)
21 }

```

```

> source('D:/USP/Monitoria/PCS3834/TSP_6426934.R')
> TSP_6426934('D:/USP/Monitoria/PCS3834/EP1/tsp.csv', 'D:/USP/Monitoria/PCS3834/EP1/saida.csv')
Custo: 205
Tempo de Execucao: 0.3305011
Rota: 1 4 3 2
>

```