MAC0110 – Introdução à Computação

Exercício-Programa 4 (EP4)

Data de entrega: 18 de dezembro de 2020

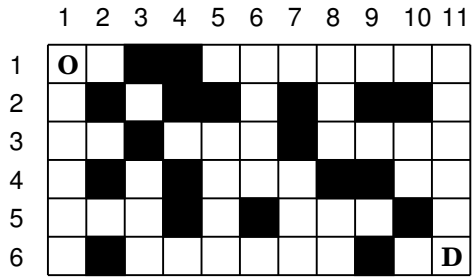
Procurando um caminho num labirinto

Vamos representar um labirinto retangular através de uma matriz chamada `labirinto`, cujos elementos são 0 ou -1, conforme a posição correspondente do labirinto seja uma passagem livre ou uma parede, respectivamente. Dados dois pontos deste labirinto, chamados de `origem` e `destino`, desejamos saber se existe um caminho entre a origem e o destino. Se existente, desejamos encontrar um caminho mínimo entre tais pontos. Vamos impor que, a cada passo, seja permitido andar somente nas direções horizontal ou vertical.

O propósito do exercício-programa (EP4) é resolver o problema acima, conforme as instruções dadas a seguir. Propomos dois procedimentos separados para encontrar (se existente) um caminho mínimo (mais curto possível) da `origem` para o `destino`. Primeiramente, você deve usar um algoritmo (explicado a seguir) para marcar umas posições da matriz. Depois, veremos como podemos usar essas marcações para encontrar (se existente) um caminho mínimo.

> Algoritmo de marcação:
>
> Marcar com um inteiro k $(k = 1, 2, \ldots)$ todas as posições livres da matriz `labirinto` que estejam a exatamente $k - 1$ passos de distância do `destino`, pelo caminho mais curto possível. Para isto, marque inicialmente a posição do `destino` com 1 e, para cada $k \geq 2$, marque com k todas as posições livres ainda não marcadas e que sejam adjacentes a alguma posição marcada com $k - 1$. A marcação termina quando para algum k não é mais possível marcar nenhuma posição.

Como exemplo, considere o labirinto representado (pictoricamente) abaixo, no qual indicamos as posições da `origem`(**O**) e do `destino`(**D**):



A matriz `labirinto` correspondente ao labirinto acima é uma matriz cujas entradas são 0 ou -1 (indicando passagem livre ou parede). Após a fase de marcação, ela deve ficar com as seguintes entradas.

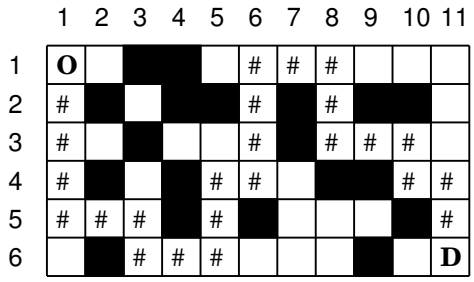
26	27	-1	-1	12	11	10	9	8	7	6
25	-1	0	-1	-1	12	-1	8	-1	-1	5
24	25	-1	15	14	13	-1	7	6	5	4
23	-1	21	-1	15	14	15	-1	-1	4	3
22	21	20	-1	16	-1	16	17	18	-1	2
23	-1	19	18	17	18	17	18	-1	2	1

Dizemos que uma posição da matriz está marcada se, após a fase de marcação, essa posição contém um inteiro positivo. Se a posição da `origem` estiver marcada (no caso do exemplo, o valor marcado é 26), isto significa que existe um caminho da `origem` para o `destino` (passando por 24 outras posições do labirinto). Vejamos como encontrar um tal caminho (que vai ser necessariamento mínimo).

> Algoritmo para encontrar um caminho mínimo:
>
> Partir da `origem` e, a cada passo, movimentar-se para uma posição adjacente cuja numeração seja consecutivamente menor do que a da posição atual, até chegar na posição marcada com 1 (o destino). (No caso do exemplo, a partir do valor 26, procura-se uma posição adjacente com o valor 25, e assim sucessivamente até chegar no valor 1).

Note que, pode haver mais de um caminho mínimo. Se isto ocorrer, basta apresentar apenas um. Observe que, no exemplo dado, estando na posição marcada com 15, há duas posições adjacentes marcadas com 14 (escolha uma delas); mais adiante, estando na posição marcada com 9 também há duas posições adjacentes marcadas com 8. Ou seja, no labirinto do exemplo há mais de um caminho mínimo entre a `origem` e o `destino` dados.

Com relação ao exemplo dado, podemos indicar um caminho mínimo da seguinte forma (ou uma outra forma alternativa que deixe bem claro quais são as posições livres, as paredes, e o caminho):



Também é conveniente indicar o caminho encontrado através de uma sequência de pares, representando os índices de linha e de coluna de cada uma de suas posições:

```
(1, 1)   (2, 1)   (3, 1)   (4, 1)   (5, 1)   (5, 2)   (5, 3)   (6, 3)   (6, 4)   (6, 5)
(5, 5)   (4, 5)   (4, 6)   (3, 6)   (2, 6)   (1, 6)   (1, 7)   (1, 8)   (2, 8)   (3, 8)
(3, 9)   (3, 10)  (4, 10)  (4, 11)  (5, 11)  (6, 11)
```

Escreva um programa em Python 3.x que, dados um labirinto e as posições da origem e do destino, determina e imprime, se existir, **um caminho de comprimento mínimo** da origem para o destino, conforme detalhamos mais adiante. Se não existir nenhum caminho, imprima a mensagem correspondente. Veja a seguir as especificações que você deve seguir para implementar o seu programa.

Sobre a implementação do programa:

(a) Os dados de entrada devem ser lidos de um arquivo texto com o seguinte formato:

Primeira linha: inteiros positivos `m` e `n`, representando o número de linhas e o número de colunas do labirinto.

Segunda linha: inteiros `lin_origem` e `col_origem`, representando os índices de linha e de coluna da posição da `origem` no labirinto.

Terceira linha: inteiros `lin_destino` e `col_destino`, representando os índices de linha e de coluna da posição do `destino` no labirinto.

Próximas `m` linhas: cada linha do arquivo representa a configuração inicial da respectiva linha do labirinto (0 indica uma passagem livre e -1 indica uma parede).

(b) Implemente em seu programa, obrigatoriamente, pelo menos as funções cujos protótipos estão descritos a seguir. Não altere o protótipo de nenhuma delas.

```
def main():
    """ () -> NoneType

    ...
    """


def le_cria_labirinto():
    """ () ->  matriz, int, int, int int

    Esta função lê todos os dados de um arquivo cujo nome deve ser fornecido
    pelo usuário (conforme descrito no item (a)). Ela cria uma matriz
    (com  moldura) com as informações lidas, e retorna essa matriz, os índices
    de linha e de coluna da posição da origem e os índices de linha e de coluna
    da posição do destino.
    """


def marca_labirinto(matrizL, lin_destino, col_destino):
    """ (matriz, int, int) -> NoneType

    Recebe uma matriz de inteiros (com moldura) matrizL, representando um labirinto,
    e dois inteiros lin_destino e col_destino que são os índices de linha e de coluna
    da posição do destino nesse labirinto. Efetua a marcação da matrizL, conforme
    o algoritmo que foi descrito.
    """
```

Para fazer a marcação na função acima, você deve utilizar uma lista para armazenar os pares ("2-uplas") contendo os índices de linha e de coluna das posições já marcadas do labirinto. Chame essa lista de `pares_linha_coluna`. No exemplo dado, teremos `pares_linha_coluna` = [(6,10)], e em seguida `pares_linha_coluna` = [(6,10), (5,11)]. Note que esses pares são as posições da matriz marcadas com o valor 2. Tal lista vai crescendo com os pares correspondentes a $k = 3, 4, \ldots$; e deve ser usada para fazer de modo eficiente a marcação da matriz. Detalhes sobre este procedimento serão explicados na aula.

```
def determina_um_caminho(matrizL, lin_origem, col_origem):
    """ (matriz, int, int) -> matriz, list

    Recebe uma matriz de inteiros (com moldura) matrizL, representando um labirinto
    já marcado, e dois inteiros lin_origem e col_origem que são os índices de linha
    e de coluna da posição da origem nesse labirinto.

    A função supõe que existe um caminho da origem para o destino e determina um tal
    caminho de comprimento mínimo (fazendo uso do algoritmo descrito). [Ou seja, esta
    função só deve ser chamada quando se sabe que existe um tal caminho.]

    A função cria uma matriz de caracteres (como mostrada no exemplo), diferenciando
    as posições livres das posições que representam paredes, e indicando nessa matriz
    as posições da origem, do destino e do caminho encontrado.
    (Para facilitar, pode criar essa matriz com moldura.)

    Esta função cria também uma lista, onde cada elemento é um par representando
    os índices de linha e de coluna de uma posição do caminho encontrado.
    A função retorna a matriz e a lista criadas.
    """


def imprime_labirinto_numericamente(matrizL):
    """ (matriz) -> NoneType

    Recebe uma matriz de inteiros (com moldura) matrizL, representando
    um labirinto (antes ou depois da marcação).
    Imprime o labirinto (sem a moldura), no formato de matriz e ajustada
    nas colunas (exibindo também os índices das linhas e das colunas).
    """


def imprime_labirinto_simbolicamente(matrizC):
    """ (matriz) -> NoneType

    Recebe uma matriz de caracteres (com moldura) matrizC, representando
    um labirinto já com um caminho de comprimento mínimo.
    Imprime o labirinto (sem a moldura), no formato de matriz e ajustada
    nas colunas (exibindo também os índices das linhas e das colunas).
    """


def imprime_caminho(lin_col_caminho):
    """ (list) -> NoneType

    Recebe uma lista lin_col_caminho, onde cada elemento é um par representando
    os índices de linha e de coluna de uma posição do caminho encontrado.
    Imprime os índices das posições do caminho encontrado (conforme o exemplo dado).
    """
```

(c) Para a saída, inicialmente, o programa deve fornecer uma descrição resumida do problema e todas as informações do labirinto considerado. Em seguida, deve imprimir numericamente o labirinto antes e depois da marcação. Se existir um caminho de comprimento mínimo da origem para o destino, deve imprimir simbolicamente o labirinto com a indicação do caminho encontrado. Deve imprimir também as

posições (índices de linha e de coluna) desse caminho. Em caso contrário, deve imprimir uma mensagem correspondente.

Observação: Utilize em seu programa apenas os recursos da linguagem python vistos em aula.