Matrizes e Objetos Multidimensionais

MC102

Matrizes

Importância para computação

- Processamento de imagem:
  - Uma imagem pode ser vista como uma matriz de pixels.
  - Operações como blur, rotação e detecção de bordas usam vizinhanças da matriz.
- Mapas e geolocalização:
  - Grades de terreno, mapas de calor e distâncias em malhas são modeladas com matrizes.
- LLM e redes neurais:
  - Entradas, pesos e ativações são organizados como vetores e matrizes.
  - Multiplicação de matrizes aparece o tempo todo no treinamento e inferência.
- Jogos de tabuleiro:
  - Tabuleiros de xadrez, damas e sudoku são representados como matrizes.
  - Regras de movimento e validação consultam posições por linha e coluna.

Breve revisão de listas em Python

- Criando listas:

```
numeros = [10, 20, 30]
nomes = ["Ana", "Bruno", "Carla"]
misturada = [1, "abc", True]
```

- Acessando elementos (índices começam em 0):

```
numeros[0]   # 10
numeros[2]   # 30
numeros[-1]  # último elemento
```

- Inserindo no final com `.append()`:

```
numeros.append(40)
# numeros agora é [10, 20, 30, 40]
```

Matrizes em Python

Em python, matrizes são representadas como **listas de listas**.

Isto é, podemos representa a matriz:

$$\begin{pmatrix} 7 & 0 & 2 & 3 \\ 3 & 1 & 4 & 2 \\ 0 & 3 & 2 & 7 \end{pmatrix}$$

como `m = [[7, 0, 2, 3], [3, 1, 4, 2], [0, 3, 2, 7]]`.

- Isto é uma linha de linhas da matriz.
- Onde cada linha é, também, uma lista.
- O elemento na linha `i` e coluna `j` é acessado por `m[i][j]`.

Entrada de matriz

- Como ler uma lista de números?

```
linha = input().split()
for i, elemento in enumerate(linha):
    linha[i] = int(elemento)
```

```
linha = [int(x) for x in input().split()]
```

- Se uma matriz é uma lista de listas (as linhas) e sabemos ler uma linha, como ler várias linhas?

```
n = int(input())  # número de linhas
matriz = []
for _ in range(n):
    linha = [int(x) for x in input().split()]
    matriz.append(linha)
```

```
n = int(input())  # número de linhas
matriz = [[int(x) for x in input().split()] for _ in range(n)]
```

Entrada de matriz

</> Code

Faça uma função que recebe um número n e retorna uma matriz n x n identidade.

```python
def identidade(n: int) -> list[list[int]]:
    matriz = []
    for i in range(n):
        linha = [0 for _ in range(n)]  # Cria uma linha de zeros
        linha[i] = 1                   # Coloca um 1 na posição diagonal
        matriz.append(linha)
    return matriz
```

Entrada de matriz

</> Code

Faça uma função que recebe uma lista de listas e retorna True ou False se é uma matriz ou não.

```python
def eh_matriz(lista: list[list]) -> bool:
    tamanho = len(lista[0])        # Tamanho da primeira linha
    for linha in lista:
        if len(linha) != tamanho:  # Verifica se todas as linhas têm o mesmo tamanho
            return False
    return True
```

Entrada de matriz

</> Code

Escreva uma função que imprime uma matriz de forma legível.

```python
def imprimir_matriz(matriz: list[list]):
    for linha in matriz:
        print(*linha)
```

Acesso a elementos

- Suponha a seguinte matriz:

```
m = [
  [7, 0, 2, 3],
  [3, 1, 4, 2],
  [0, 3, 2, 7],
]
```

- Acessando um elemento específico (`m[i][j]`):

```
m[1][2]   # 4  (linha 1, coluna 2)
m[0][0]   # 7
```

- Acessando linha inteira e coluna inteira:

```
linha_1 = m[1]
# [3, 1, 4, 2]
coluna_2 = [linha[2] for linha in m]
# [2, 4, 2]
```

Acesso a elementos

</> Code

Escreva uma função que recebe uma matriz e retorna a transposta dela.

```python
def transposta(matriz: list[list]) -> list[list]:
    n_linhas = len(matriz)
    m_colunas = len(matriz[0])
    transposta = [[0 for _ in range(n_linhas)] for _ in range(m_colunas)]

    for i in range(n_linhas):
        for j in range(m_colunas):
            transposta[j][i] = matriz[i][j]

    return transposta
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna a soma delas.

```python
def soma_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:
    n_linhas = len(m1)
    m_colunas = len(m1[0])
    soma = [[0 for _ in range(m_colunas)] for _ in range(n_linhas)]

    for i in range(n_linhas):
        for j in range(m_colunas):
            soma[i][j] = m1[i][j] + m2[i][j]

    return soma
```

Acesso a elementos

</> Code

Escreva uma função que recebe duas matrizes e retorna o produto delas.

```python
def produto_matrizes(m1: list[list[int]], m2: list[list[int]]) -> list[list[int]]:
    ... # Assumindo que eu tenho as dimensões das matrizes
    produto = [[0 for _ in range(m_colunas_m2)] for _ in range(n_linhas_m1)]

    for i in range(n_linhas_m1):
        for j in range(m_colunas_m2):
            for k in range(m_colunas_m1):
                produto[i][j] += m1[i][k] * m2[k][j]

    return produto
```

Objetos Multidimensionais

Matrizes de matrizes

- Até agora, cada posição da matriz guardava apenas um número.
- Mas podemos guardar estruturas maiores em cada posição, como listas, vetores ou até outras matrizes.
- Quando isso acontece, o objeto deixa de ser bidimensional e passa a ser multidimensional.

- Exemplo de um objeto 3D em Python:

```
objeto = [
  [
    [1, 2],
    [3, 4],
  ],
  [
    [5, 6],
    [7, 8],
  ],
]
```

- Para acessar um elemento específico, usamos um índice para cada nível:

```
objeto[1][0][1]   # 6
```

- Leitura intuitiva: primeiro escolhemos a camada, depois a linha, e por fim a coluna ou posição interna.

Dúvidas?