Dokumentace úlohy číslo 14 Výpočet součinu dvou matic v rámci zkoušky z předmětu PřfUK Úvod do programování (ZS 2023/24)

Výpočet součinu dvou matic

Nechť A (m, n) a B (m, n) jsou obdélníkovými maticemi. Spočtěte jejich součin C = AB . Pokud není možné z důvodu nekompatibilních rozměrů výpočet provést, informujte o tom uživatele. Matice reprezentujte 2D listem, vstupní data načtěte/uložte z/do textového souboru.

Existující algoritmy

Pro násobení matic existuje standardní algoritmus, který využívá cyklus pro iteraci přes řádky a sloupce matic. Existuje další velké množství algoritmů pro výpočet násobení matic, mezi ně patří například Strassenův algoritmus pro efektivnější výpočet větších matic.

Zvolený algoritmus

Pro tento program byl zvolen standardní algoritmus pro výpočet matic, který sčítá násobky prvků řádků matice m1 a prvků sloupců matice m2.

Struktura programu

Nejdříve definujem třídu `Matrix`, která očekává dvě matice ve formě 2D listů.

```
# Define class matrix that asks the user for two 2D arrays m1 and m2
class Matrix:
    def __init__(self):
        self.m1 = self.getmatrix("first")
        self.m2 = self.getmatrix("second")
```

Poté definujeme metodu `getmatrix`, která se uživatele zeptá, jestli chce zadat matice manuálně nebo je načíst ze soubouru. Pokud uživatel zvolí cestu čtení ze souboru, program se zeptá na cestu k němu. Toto provede pro každou matici zvlášť.

```
# ask the user if they want to read the matrix from a file or input
def getmatrix(self, name):
    choice = input(f"Do you want to read the {name} matrix from a file? (y/n): ")
    if choice.lower() == "y":
        filename = input(f"Enter the filename for the {name} matrix: ")
        matrix = self.readmatrixfromfile(filename)
    else:
        matrix = self.readmatrixfrominput(name)

    return matrix
```

Pokud uživatel zvolí cestu čtení ze souboru, program otevře zmiňované soubory a předpokládá, že jsou v souboru zapsány po řádcích a jednotlivé hodnoty odděleny mezerami. Matice je potom uložena do 2D listu. Kód je opatřen chybovou hláškou, pokud daný soubor nenalezne.

```
# read the matrix from a file
def readmatrixfromfile(self, filename):
    matrix = []
    try:
        with open(filename, "r") as file:
            for line in file:
                row = [int(num) for num in line.strip().split()]
                matrix.append(row)
    except FileNotFoundError:
        print(f"File '{filename}' not found. Please try again.")
        matrix = self.readmatrixfromfile(filename)

    return matrix
```

Pokud uživatel zvolí cestu manuálního zadání matic. Kód se zeptá na počet řádků a počet sloupců. A poté se zeptá na jednotlivé členy matice. Matice je poté uložena do 2D listu.

```
# read the matrix from input
def readmatrixfrominput(self, name):
    rows = int(input(f"Number of rows of the {name} matrix: "))
    columns = int(input(f"Number of columns of the {name} matrix: "))
    matrix = []
    for i in range(rows):
        row = []
        for j in range(columns):
            element = int(input(f"Element at position ({i + 1}, {j + 1}) of the
{name} matrix: "))
            row.append(element)
        matrix.append(row)

    return matrix
```

Definujeme metodu na zobrazení zadaných matic.

```
# print the matrix in a readable format
def printmatrix(self, m1, m2):

    print("First matrix:")
    for row in m1:
        print(row)

    print("\nSecond matrix:")
    for row in m2:
        print(row)
```

Definujeme třídu `MatrixOperations`, která podle zásad OOP dědí metody třídy `Matrix`.

```
# define class operations that inherits from the matrix class
class MatrixOperations(Matrix):
    def __init__(self):
        super().__init__()
```

Definujeme metodu `multiply` na samotný výpočet součinu matic. Metoda zkontroluje zda-li je splněna podmínka pro výpočet součinu. Toto je opatřeno chybovou hláškou.

```
def multiply(self):
    # check if multiplication is not possible
    if len(self.m1[0]) != len(self.m2):
        # raise a ValueError if yes
        raise ValueError("The number of columns in the first matrix must "
                         "be equal to the number of rows in the second "
                         "matrix"
                         )
```

Poté je vytvořen prázdný 2D list pro výslednou matici.

```
# create a result matrix with zeros
result = [
    [0 for _ in range(len(self.m2[0]))] for _ in range(len(self.m1))
    ]
```

Samotný algoritmus na výpočet součinu matic se skládá z vnořených cyklů, které postupně násobí prvky obou matic, sčítají je a následně ukládají do předem definované matice `result`.

```
# iterate through the rows of m1
for i in range(len(self.m1)):
    # iterate through the columns of m2
    for j in range(len(self.m2[0])):
        # iterate through the rows of m2
        for k in range(len(self.m2)):
            # multiply the elements of m1 and m2 and add them to the
            # result matrix at the corresponding position
            result[i][j] += self.m1[i][k] * self.m2[k][j]
```

Výsledná matice je zobrazena.

```
print("\nResult matrix:")
    for row in result:
        print(row)
```

Uživatel dostává možnost uložení výsledné matice do souboru.

```
choice = input("Do you want to save the result matrix? (y/n): ")
if choice.lower() == "y":
    self.savematrix(result)
```

Definujeme metodu na uložení výsledné matice do souboru. Pokud se uložení podařilo je o tom uživatel oznámen.

```
def savematrix(self, matrix):
    with open('./matrix/result.txt', "w") as file:
        for row in matrix:
            file.write(" ".join(str(num) for num in row) + "\n")
    print("Matrix saved successfully")
```

Datové struktury

V rámci kódu se využívá takzvaný *nested list* (nebo také 2D list) pro práci s maticemi.

Vstupní/výstupní data

Vstupní data jsou dvě zadané matice `m1` a `m2` pomocí 2D listu. Jako výstupní data `result` je také matice ve formě 2D listu.

Problematická místa kódu

Ačkoli kód kontroluje, zda je násobení možné, ale neošetřuje případ, kdy jsou matice prázdné. Kód nekontroluje vstup zadaný uživatelem, program může selhat díky `ValueError`. Zároveň kód využívá pevné cesty k souboru, pokud kód nebude spuštěn v originální složce, selže.

Využitý algoritmus je neefektivní pro využití na větší matice.

Možná vylepšení

Jedním z možných vylepšení kódu by bylo přidání ošetření případu prázdných matic a pro matice vyšších řádů by bylo možné implementovat efektivnější algoritmus (jako třeba zmiňovaný Strassenův algoritmus). Zároveň by stálo za to implementovat postupy, které by umožnili lepší flexibilitu kódu.