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 definujeme matice pro výpočet pomocí 2D listů:

```
m1 = [[1, 2, 3],
      [1, 2, 3],
      [1, 2, 3],
      [1, 2, 3]]
m2 = [[1, 2, 3],
      [1, 2, 3],
      [1, 2, 3]]
```

Definujeme třídu `Multiply`, která má dvě matice jako vstupní parametry.

```
class Multiply:
    def __init__(self, m1, m2):
        self.m1 = m1
        self.m2 = m2
```

Definujeme metodu `multiply` a jako první definujeme kontrolu, zda je násobení zadaných matic možné. Pokud možné není, program zobrazí `ValueError` a stručný popis chybové hlášky.

```
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"
                             )
```

V rámci metody `multiply` se vytvoří matice `result` na základě velikostí zadaných matic s nulovými hodnotami.

```
# 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`.

Poté metoda vypíše výsledný 2D seznam.

```
# 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]

        print(result)
```

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.

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).