Cílem osmé úlohy je vyzkoušet si vzdálené volání procedur v jazyce Python pomocí protokolu XML-RPC (ang. Remote Procedure Call) [1]. Pro kódování zpráv mezi klientem a serverem je použito značkovacího jazyka XML a přenos je realizován pomocí HTTP protokolu aplikační vrstvy. Přenášenou zprávou bude výsledek Fibonnacciho posloupnosti [2] vypočtené pomocí jedné z technik využívaných v rámci dynamického programování.

Následující body by vás měly postupně provést detaily implementace jednotlivých úkolů. Pro spuštění šablony budete potřebovat doinstalovat do svého virtuálního prostředí knihovnu `matplotlib` pro vykreslování grafů:

```
pip install matplotlib
```

Úkol 1: (0 bodů)

Implementace naivního výpočtu Fibonacciho posloupnosti.

Vytvořte funkci `Fib(n)`, která pro zadané celé číslo `n` vrátí výsledek podle následující rekurzivní definice

$$F(n) = \begin{cases} 0 & \text{pro } n = 0, \\ 1 & \text{pro } n = 1, \\ F(n-1) + F(n-2) & \text{pro } n \geq 2. \end{cases}$$

Časová složitost takovéhoto algoritmu bude odpovídat $O(1{,}6180^n)$ (exponenciální).

Úkol 2: (1 bod)

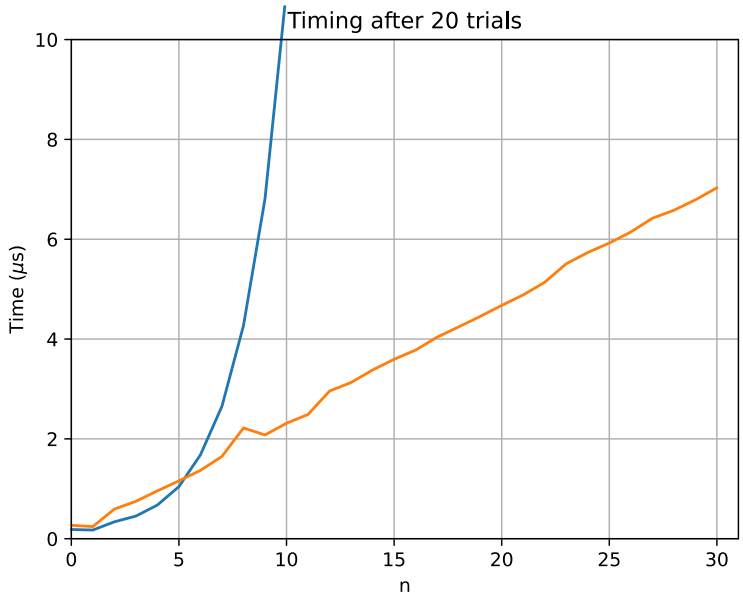
Implementace Fibonacciho posloupnosti pomocí memoizace.

Uvedený algoritmus pro výpočet Fibonacciho posloupnosti z prvního úkolu lze jednoduše upravit pomocí techniky používané v dynamickém programovaní zvané memoizace (caching) [3]. Cílem memoizace je si zapamatovat jednou pracně vypočtené výsledky pro jejich případné následné znovupoužití bez nutnosti opakovat zdlouhavý výpočet. Hodnoty jsou obvykle ukládány do slovníků, které si na základě zadaného klíče dokážou (při použití efektivního hashování) vybavit uložené hodnoty s časovou složitostí $O(1)$ (konstantní). V případě výpočtu Fibonacciho posloupnosti jsme pak při využití memoizace schopni snadno dosáhnout výrazně lepší výsledné časové složitost, a to $O(n)$ (lineární). Stačí, když každý nově vypočtený (mezi)výsledek uložíme do pomocného slovníku (ozn. `memo`) pod klíčem, který je roven aktuální hodnotě parametru `n`. Zároveň musíme zajistit, že v případě existence klíče `n` ve slovníku `memo`, vrátíme hodnotu uloženou pod tímto klíčem namísto provádění rekurze. Tímto postupem zásadně redukujeme časovou složitost algoritmu na úkor prostorové. Tedy další otázkou při snaze o zefektivnění uvedeného algoritmu může být určení počtu prvků, které si (nutně) musíme pamatovat (promyslete).

Úkol 3: (1 bod)

Vykreslení časové složitosti obou implementací.

Pomocí předpřipraveného skriptu využívajícího knihovnu `matplotlib` vykreslete časy běhu obou funkcí pro danou hodnotu parametru `n`. Střední dobu běhu jednotlivých volání testovaných funkcí uložte do pomocného seznamu v mikrosekundách. Test pro každé n z intervalu $\langle 0, N \rangle$ opakujte M-krát pro minimalizaci chyby měření. Do grafu můžete doplnit legendu (modrá čára je čas naivní implementace a oranžová odpovídá memoizované verzi). Výsledný graf by měl vypadat přibližně takto:



Průběhy obou grafů korespondují s očekávanou časovou složitostí obou implementací. Pro co nejpřesnější měření délky časového úseku běhu zkoumané procedury můžete využít funkci `perf_counter` z modulu `time`:

```
t0 = time.perf_counter()
f = F(n)
t1 = time.perf_counter()
dt = t1 - t0
```

Úkol 4: (1 bod)

Vzdálené volání procedur.

Doplňte do šablony kód, který vytvoří XML-RPC server (třída `SimpleXMLRPCServer` z modulu `xmlrpc.server` [4]) na lokální adrese (`'127.0.0.1'` nebo `'localhost'`) a portu 10001. Dále na serveru zaregistrujte vybranou funkci pro výpočet Fibonacciho posloupnosti pod jménem `'fibonacci'`. Rovněž na serveru zaregistrujte pomocí metody `register_introspection_functions` sadu funkcí pro introspekci. Nakonec spusťte nekonečnou smyčku serveru.

Následně vytvořte nový skript, ve kterém inicializujete XML-RPC klienta (třída `ServerProxy` z modulu `xmlrpc.client` [5]), který se připojí k běžícímu serveru, vypíše seznam všech na něm zaregistrovaných

funkcí, vypíše nápovědu k funkci `'fibonacci'` (předmětný doc string je uveden v kódu na serveru) a zavolá vzdálený výpočet Fibonacciho posloupnosti pro zadané n. Výsledek vytiskne a ukončí se.

Reference

[1] https://www.tutorialspoint.com/xml-rpc/xml_rpc_intro.htm
[2] https://cs.wikipedia.org/wiki/Fibonacciho_posloupnost
[3] https://en.wikipedia.org/wiki/Memoization
[4] https://docs.python.org/3/library/xmlrpc.server.html
[5] https://docs.python.org/3/library/xmlrpc.client.html