IB111

Základy programování

František Lachman
lachmanfrantisek@mail.muni.cz

cvičení 9

15. listopad 2017

Osnova

- kontrolní otázky
- rekurze
- změnši a panuj
- rozděl a panuj
- fraktály
- akumulátory

Kontrolní otázky

- Co je to rekurze?
- Uveďte příklad fraktálu s výraznou rekurzivní strukturou.
- Jak zapíšeme výpočet faktoriálu:
  - a) za použití rekurze,
  - b) bez použití rekurze?

Kontrolní otázky

- Jak můžeme rekurzivně zapsat funkci pro otočení řetězce?
- Co je efektivnější, rekurzivní, nebo iterativní řešení? Jaké jsou (ne)výhody jednotlivých přístupů?
- Jaká je základní myšlenka řešení úlohy Hanojské věže?

Rekurze

- definice pomocí sebe sama
  - bázový případ pro zastavení výpočtu
  - volání sama sebe s jinými parametry
- "spirála"

Rekurze

- rekurzivní definice:

```
0! = 1
n! = n * (n - 1)! pro n > 0
```

```
def faktorial(n):
    if n == 0:
        return 1
    else:
        return n * faktorial(n-1)

    print(faktorial(4))
```

- animace ve sbírce

NSD - Euklidův algoritmus (příklad)

$$nsd(a, b) = nsd(a \ \ mod \ \ b, b) \ \ pro \ \ b > 0$$

- Vytvořte funkci `euklid(a,b)` rekurzivně implementující výpočet největšího společného dělitele `a` a `b` pomocí euklidova algoritmu.

Zmenši a panuj

- Řešení získáme z řešení stejného problému "menší" velikosti.
- Lze jednoduše implementovat i iterativně.

Zmenši a panuj - příklady

- 8.1.3. Zašroubování žárovky

Napište rekurzivní funkci `screwing(n)`, která `n`-krát vypíše `twist`.

Zmenši a panuj - příklady

- 8.1.1. Součet čísel

Napište rekurzivní funkci `series_sum(n)`, která zjistí součet přirozených čísel od `1` do `n`. Při vymýšlení funkce `series_sum(n)` si představte, že máte k dispozici funkci `series_sum(k)` pro libovolné `k` < `n`. Nezapomeňte na bázový případ.

Zmenši a panuj - příklady

- 8.1.5. Binární vyhledávání

Napište funkci `binarny_search(value, numbers)`, která zjistí, zda se hodnota `value` nachází ve vzestupně uspořádaném seznamu `numbers`. Funkce musí mít logaritmickou časovou složitost. Pozor na to, že kopírování seznamu má lineární časovou složitost vzhledem k počtu prvků. Je tedy vhodné napsat si pomocnou funkci, která bude hledat zadané číslo v části seznamu ohraničeném 2 parametry (např. `left` a `right` pro levou a pravou mez intervalu, ve kterém hledáme).

Rozděl a panuj

0. řešení pro bázový případ
1. rozdělení na menší problémy stejného typu
2. vyřešení podproblémů
3. spojení podproblémů do řešení

Rozděl a panuj - příklady

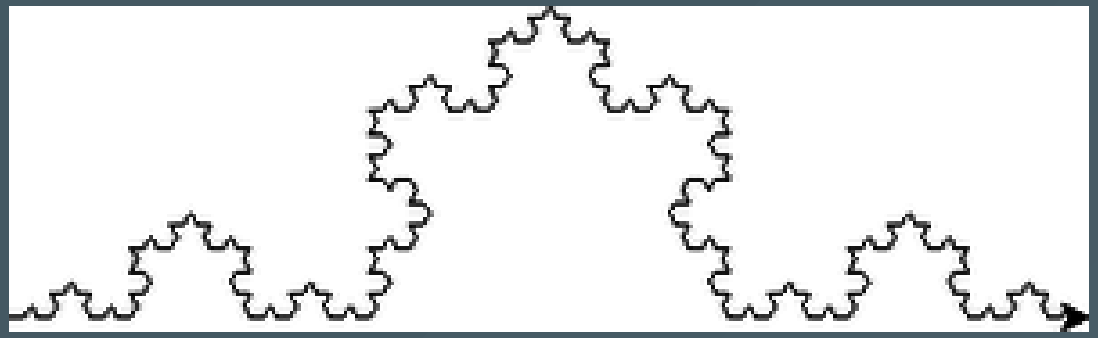
- 7.2.6. Merge sort

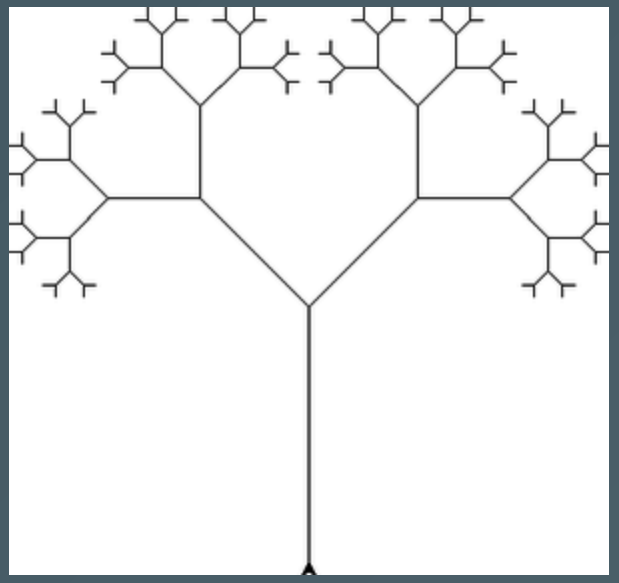
Implementujte Merge sort pomocí funkce `merge_sort(values)`. Funkce nemusí měnit původní seznam (může vrátit nový, seřazený).

- *** 8.2.4. Rychlé řazení/quicksort

Fraktály

- "sebepodobné obrazce"
- části se podobají celku

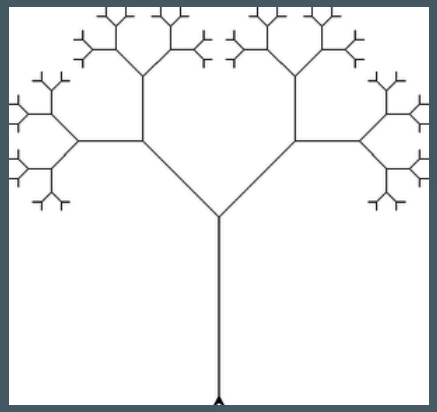




Fraktály - příklady

- 8.3.2. Strom

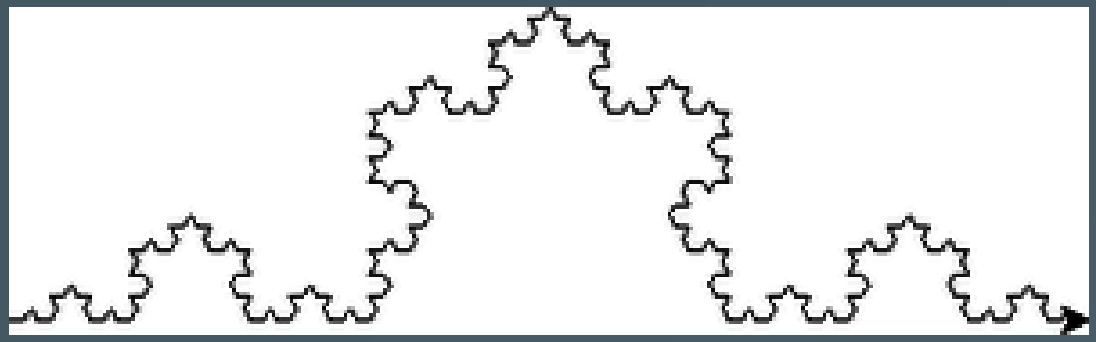
S využitím techniky rozděl a panuj napište rekurzivní funkci `tree(length)` pro vykreslení stromku, jako vidíte na obrázku. Po vykreslení základního stromku můžete vyzkoušet různé variace, např. změnit stupeň větvení, délku větví, naklonění větví atd.



Fraktály - příklady

- 8.3.3. Kochova křivka/vločka

Napište rekurzivní funkci `koch(depth, size)` pro vykreslení Kochovy křivky. Vhodným složením tří Kochových křivek (jakoby do trojúhelníka) získáte Kochovu vločku.



Akumulátory

- zefektivnění rekurze
- ušetření duplicitních výpočtů

Akumulátory - příklady

- 8.4. Akumulátor

Napiště funkci `fibonacci_accumulator(n, fib_act, fib_next)`, pomocí které lze efektivněji rekurzivně implementovat faktoriál.

```
def fibonacci_accumulator(n, fib_act, fib_next):
    pass

def fibonacci(n):
    return fibonacci_accumulator(n, 0, 1)
```

Závěr

- rekurze
- zmenši a panuj
- rozděl a panuj
- fraktály
- akumulátory
- druhá vnitrosemestrálka (bude rekurze)