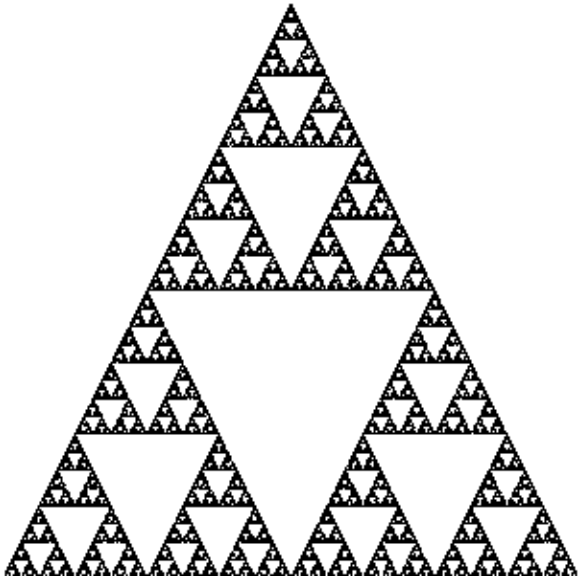
Algoritmizace

Rekurze



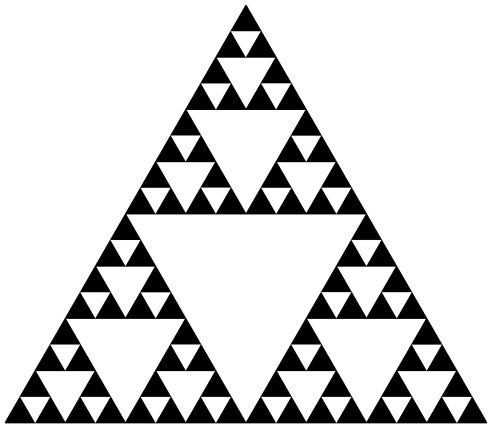
Rekurze

Objekt je definován pomocí sebe sama

- z latinského *recurro*

Příklady

- ze světa: rekurze u holiče
- z geometrie



- z matematiky

$$0! = 1$$

$$n! = n \cdot (n-1)!, n \geq 1$$

$$f_0 = f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2}, n \geq 2$$

Fibonacciho posloupnost

Rekurze v informatice

Algoritmus řešení problému

- pomocí řešení menších instancí téhož problému

Rekurzivní funkce (podprogram)

- volá sama sebe (přímo / nepřímo)
- na vstup menšího rozsahu
- podmínka ukončení (báze)

V teorii

- rekurzivní funkce : formální model algoritmu

V praxi

- logické / funkcionální programování : pouze rekurze
- imperativní programování: rekurzivní volání

Příklad: palindromy

Palindrom – čte se stejně zleva doprava
i zprava doleva

```
def palindrom(s):
    # rekurzivní test na palindrom
if len(s) <=1 :
      return True
  else:
      return s[0] == s[-1] and
             palindrom(s[1:-1])
```

Palindromy: rekurze vs. iterace

```
def palindrom_iter(s):
    # iterativní test na palindrom
    n = len(s)
    for i in range(n//2):
        if s[i] != s[n-i-1]:
            return False
    return True
```

Výhody a nevýhody rekurze

- ✔ jednoduchost, intuitivnost
- ✔ často efektivní řešení složitého problému
- ✖ obsluha rekurze není zadarmo
- ✖ může existovat efektivnější iterativní řešení

✎ Problém: rekurze vs. iterace

① V jazyce Python sestavte funkce

- pro výpočet funkce $n!$
- pro výpočet n-tého prvku Fibonacciho posloupnosti

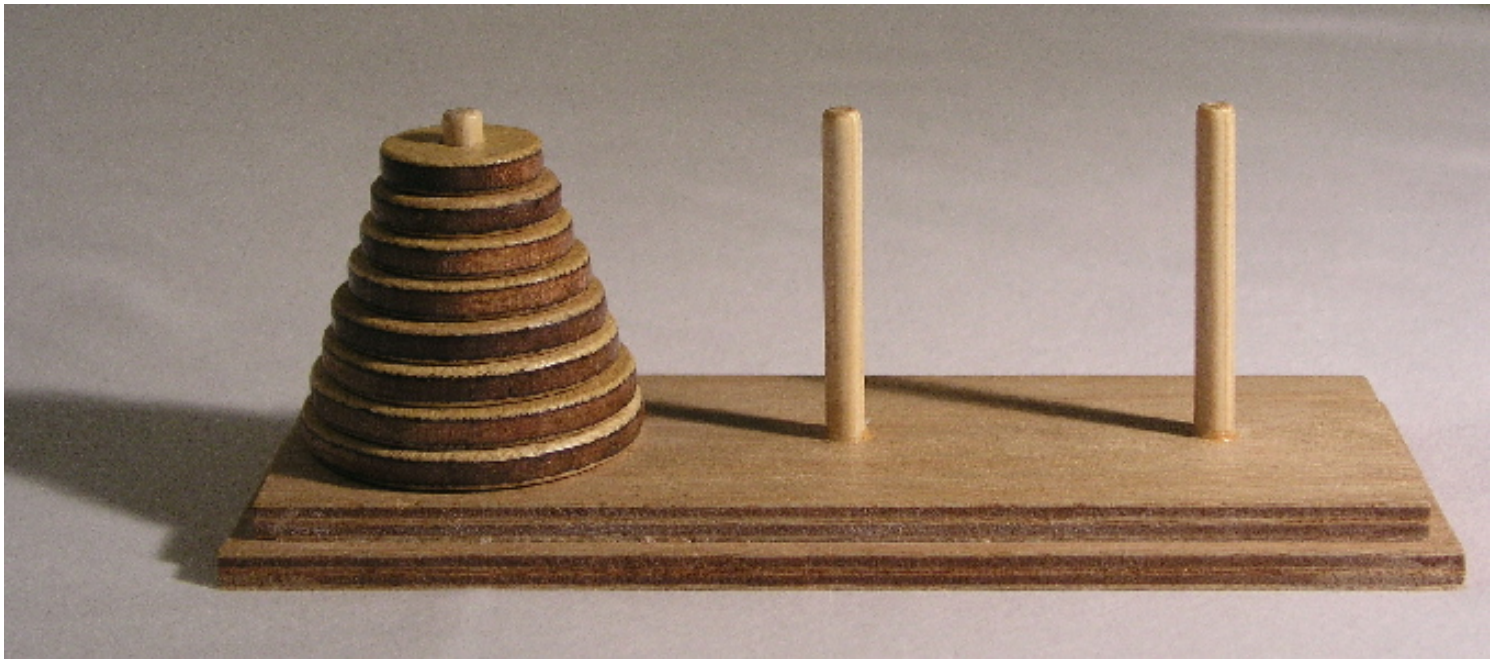
Pro každý problém se pokuste navrhnout dvě řešení

- rekurzivní
- iterativní (bez rekurze)

a porovnat jejich efektivitu.

Příklad: Hanojská věž

Édouard Lucas (1883)



Hlavolam: Jak přemístit všech n kotoučů na prostřední "věž"?

- v jednom tahu se přesouvá vždy jeden kotouč
- větší kotouč nesmí nikdy ležet na menším

Hanojská věž: rekurzivní řešení

```
def hanoj(n,odkud,kam,rezervni):
  if n == 1:
    print("presuň kotouč z věže",
          odkud, "na věž", kam)
    else:
        hanoj(n-1,odkud,rezervni,kam)
        hanoj(1,odkud,kam,rezervni)
        hanoj(n-1,rezervni,kam,odkud)
```

✎ Problémy: Hanojská věž

② Kolik tahů potřebuje náš (rekurzivní) algoritmus na přemístění n kotoučů?

③ Dle legendy existuje v Asii chrám, v němž každý den v poledne mniši slavnostně přemístí jeden z 64 zlatých kotoučů. Jakmile bude přemístěna celá “věž”, nastane konec světa. Spočítejte, za jak dlouho k tomu může dojít.