Řešení rekurentních rovnic

prof. Ing. Pavel Tvrdík CSc.

Katedra počítačových systémů FIT
České vysoké učení technické v Praze

DSA, ZS 2009/10, Předn. 8

http://service.felk.cvut.cz/courses/X36DSA/

Příklady rekurentních rovnic

V předchozích přednáškách jsme viděli, že rekurzivní algoritmy dovolují okamžitě napsat výraz pro složitost ve formě rekurentní rovnice.

- QuickSort best case, MergeSort:

$$t(n) = 2t(n/2) + \Theta(n) \tag{1}$$

- Heapify:

$$t(n) \leq t(2n/3) + \Theta(1) \tag{2}$$

- QuickSort worst case:

$$t(n) = t(n-1) + \Theta(n) \tag{3}$$

- QuickSort average case:

$$t(n) = t(9n/10) + t(n/10) + O(n) \tag{4}$$

- Fibonacci:

$$t(n) = t(n-1) + t(n-2) + \Theta(1) \tag{5}$$

Metody řešení rekurentních rovnic

- Problém analýzy složitosti rekurzivního algoritmu se přesouvá na problém asymptotického řešení rekurentních rovnic.
- Ukážeme si 4 základní metody:
  1. Substituční.
  2. Iterační.
  3. Mistrovská.
  4. Anihilace posloupností.

Substituční metoda

1. Odhad (uhádnutí) tvaru řešení (=indukční hypotéza).
2. Pomocí matematické indukce nalezení konstant a ověření správnosti odhadnutého řešení.

Tuto metodu lze použít pro určení horní i dolní meze na řešení.

Příklad

Uvažujme rovnici

$$t(n) = 2t(\lfloor n/2 \rfloor) + n \tag{6}$$

Jako **horní** odhad řešení zkusme

$$t(n) \leq cn \log n, \quad \text{kde } c > 0 \text{ je vhodně zvolená konstanta.} \tag{7}$$

Indukcí dokažme správnost odhadu. Tedy, že pro řešení rovnice platí

$$t(n) = O(n \log n).$$

Příklad

Indukční krok: Předpokládejme, že (7) platí pro $\lfloor n/2 \rfloor$ a dosaďme $t(\lfloor n/2 \rfloor) \leq c \lfloor n/2 \rfloor \log \lfloor n/2 \rfloor$ do (6). Dostaneme

$$\begin{aligned} t(n) &\leq 2(c \lfloor n/2 \rfloor \log \lfloor n/2 \rfloor) + n \\ &\leq cn \log(n/2) + n \\ &= cn \log n - cn \log 2 + n \\ &= cn \log n - cn + n \\ &\leq cn \log n, \quad \text{pokud} \quad c \geq 1. \end{aligned}$$

Substituční metoda (pokračování příkladu)

Příklad

- **Indukční základ**: Je třeba dokázat, že $\exists$ konstanta $c \geq 1$ taková, že (7) přímo platí pro počáteční hodnoty n.
- Pro jednoduchost předpokládejme, že platí $t(1) = 1$. (Lze zvolit jakoukoli jinou konstantu.) Pak z (6) plyne $t(2) = 4$ a $t(3) = 5$.
- Pak ale (7) neplatí pro $n = 1$, neboť ne$\exists$ konstanta c taková, že $t(1) \leq c\,1 \log 1 = 0$.
- Naštěstí řešíme (6) asymptoticky, takže stačí ukázat, že (7) platí od nějakého $n_0 \geq 1$.
- Stačí proto ukázat, že lze zvolit dostatečně velkou konstantu c, aby platilo $t(2) = 4 \leq c\,2 \log 2$ nebo $t(3) = 5 \leq c\,3 \log 3$.
- Triviálně lze spočítat, že stačí zvolit $c \geq 2$.

Substituční metoda (pokračování příkladu)

Příklad

Naprosto analogickým způsobem lze ukázat, že pro řešení rovnice (6) platí

$$t(n) \geq cn \log n, \quad \text{kde } c > 0 \text{ je vhodně zvolená konstanta.} \tag{8}$$

Závěr:
Rovnice (6) má řešení $\Theta(n \log n)$.

Substituční metoda: Umění odhadu

- Neexistuje žádný obecný postup, který zaručí nalezení správného odhadu řešení.
- Umění dobrého odhadu je věc intuice a zkušenosti.
- Doporučují se dvě pomocné heuristiky:
  - Převedení nového neznámého případu na starý známý.
  - Postupné uzavírání do těsných dolních a horních mezí.

Příklad

Uvažujme rovnici

$$t(n) = 2t(\lfloor n/2 \rfloor + 20) + n \tag{9}$$

Pro velká n je rozdíl mezi rovnicemi (6) a (9) zanedbatelný, takže odhad (7) je použitelný. Je třeba pouze opravit konstanty.

Substituční metoda: Úskalí indukčního důkazu

- Indukční důkaz bude správný, pouze pokud jsme schopni dokázat **přesný** tvar indukční hypotézy.

Příklad

Uvažujme rovnici

$$t(n) = t(\lfloor n/2 \rfloor) + t(\lceil n/2 \rceil) + 1. \tag{10}$$

Zkusme ověřit odhad

$$t(n) \leq cn, \quad \text{kde } c > 0 \text{ je vhodná konstanta.} \tag{11}$$

Indukční krok: Substitucí do (10) dostaneme

$$t(n) \leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 = cn + 1.$$

Z toho ale **nevyplývá** $t(n) \leq cn$ pro žádnou konstantu c.

Nepřesnost je ale pouze v **aditivní** konstantě!!!!

Substituční metoda: Úskalí indukčního důkazu (pokračování příkladu)

Příklad

Náprava spočívá ve volbě silnější indukční hypotézy. Chceme-li totiž pro velká n dokázat přísnější podmínku, musíme předpokládat přísnější podmínku pro nižší n. Stačí **zlepšit** odhad o aditivní konstantu:

$$t(n) \leq cn - b, \quad \text{kde } c > 0, b > 0 \text{ jsou vhodné konstanty.} \tag{12}$$

Substitucí (12) do (10) dostaneme

$$t(n) \leq (c\lfloor n/2 \rfloor - b) + (c\lceil n/2 \rceil - b) + 1 = cn - 2b + 1 \leq cn - b,$$

pokud $b \geq 1$.

Substituční metoda: Chybování

Vždy, i přes volnost asymptotické notace, je třeba dokázat indukční hypotézu **přesně**, jinak může dojít k chybě.

Příklad

Např. pokud řešení rovnice (6) odhadneme pomocí hypotézy $t(n) \leq cn$ a dosadíme do (6), dostaneme

$$t(n) \leq 2(c\lfloor n/2 \rfloor) + n \leq cn + n,$$

což ale nedokazuje indukční hypotézu, přestože **asymptoticky** indukční hypotéza platí!!!!!

Převod nového tvaru na známý

Někdy lze novou neznámou rovnici substitucí převést na tvar, který už umíme vyřešit.

Příklad

$$t(n) = 2t(\lfloor \sqrt{n} \rfloor) + \log n, \tag{13}$$

Každé číslo n lze vyjádřit ve tvaru $n = 2^m$. Když zavedeme novou funkci $s(m) = t(2^m)$, pak rovnici (13) lze vyjádřit ve tvaru

$$s(m) = 2s(m/2) + m, \tag{14}$$

která je identická s rovnicí (6) a jejíž řešení je: $s(m) = \Theta(m \log m)$. Zapsáním v původní proměnné n dostaneme řešení

$$t(n) = \Theta(\log n \log \log n).$$

Iterační metoda

1. Rekurentní rovnici expandujeme její iterací, vedoucí na rozvoj konečné řady.
2. Řešením je součet vzniklé řady.

Také tuto metodu lze použít pro určení horní i spodní meze na řešení.

Příklad

Uvažujme rovnici

$$t(n) = 3t(\lfloor n/4 \rfloor) + n \tag{15}$$

Protože platí $\lfloor \lfloor n/4 \rfloor /4 \rfloor = \lfloor n/4^2 \rfloor$ atd, postupnou iterací dostaneme

$$\begin{aligned} t(n) &= n + 3t(\lfloor n/4 \rfloor) \\ &= n + 3\lfloor n/4 \rfloor + 3^2 t(\lfloor n/4^2 \rfloor) \\ &= n + 3\lfloor n/4 \rfloor + 3^2 \lfloor n/4^2 \rfloor + 3^3 t(\lfloor n/4^3 \rfloor) \\ &= \dots \\ &= n + 3\lfloor n/4 \rfloor + 3^2 \lfloor n/4^2 \rfloor + 3^3 \lfloor n/4^3 \rfloor) + \cdots + 3^{\log_4 n}\Theta(1). \end{aligned}$$

Iterační metoda (pokračování)

Příklad

Po zanedbání zaokrouhlovacích chyb a doplněním na nekonečnou konvergentní geometrickou řadu dostaneme

$$t(n) \leq n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i \leq 4n.$$

Obecně je třeba

1. určit počet iterací, nutných pro dosažení koncových podmínek,
2. sečíst nebo odhadnout součet konečné řady.

Kontrolní otázky:

1. Jak se bude lišit řešení rovnice

$$t(n) = 3t(\lfloor n/4 \rfloor) + kn, \quad \text{kde } k \text{ je konstanta?} \tag{16}$$

2. Jak se bude lišit řešení rovnice

$$t(n) = 3t(\lceil n/4 \rceil) + n? \tag{17}$$

Iterační metoda pro substituční metodu

Postupná iterace umožní odhadnout řešení a pak použít substituční metodu.

Příklad

Ověřme, že odhad $t(n) \leq cn$ pro nějakou konstantu $c > 0$ vyhovuje rovnici (15).

- Indukční hypotéza: $t(\lfloor n/4 \rfloor) \leq c \lfloor n/4 \rfloor$.
- Substituce: $t(n) \leq 3c \lfloor n/4 \rfloor + n \leq cn$ platí pro všechny $c \geq 4$.

Iterační metoda s pomocí stromu rekurzivních volání (SRV)

SRV jako vizualizace rozvoje rekurze je užitečnou pomůckou při iterační metodě.

(1) Zkonstruuj SRV T.
(2) Vypočti jeho hloubku $h(T)$.
(3) Vypočti jeho šířku $w(T, i)$ v hloubce i.
(4) Vypočti složitost $c(i)$ výpočtu jedné podúlohy v uzlu v hloubce i.
(5) Sečti složitost výpočtů podúloh $cc(i) = w(T, i) * c(i)$ v uzlech v hloubce i.
Speciálně sečti složitosti uzlů v hloubce $h(T)$.
(6) Sečti složitosti výpočtů přes celý T (všechny hladiny).

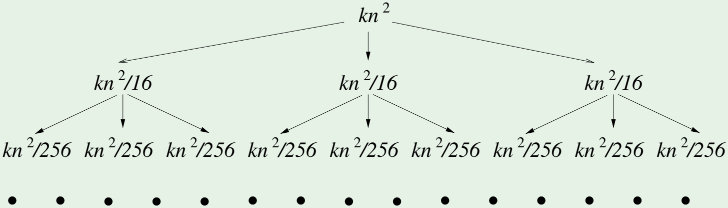
Příklad iterační metody s pomocí SRV

Příklad

$$t(n) = 3t(n/4) + kn^2, \tag{18}$$

kde $n/4$ může znamenat buď $\lfloor n/4 \rfloor$ nebo $\lceil n/4 \rceil$ a tyto rozdíly zanedbáváme.

(1) Zkonstruuj strom T rekurzivních volání.



Příklad

(2) Vypočti jeho hloubku $h(T)$.

- Velikost podúlohy v hloubce i je $n/4^i$.
- Listy v poslední hladině stromu odpovídají podúlohám konstantní velikosti, čili platí $n/4^{h(T)} = \Theta(1)$.
- Protože $n = 4^{\log_4 n}$, dostáváme $h(T) = \log_4 n$ a T má $\log_4 n + 1$ hladin uzlů.

(3) Vypočti jeho šířku $w(T, i)$ v hloubce i.

- T je úplný 3-ární strom.
- V hloubce i je tedy $w(T, i) = 3^i$ podúloh.
- V poslední hladině $h(T)$ je tedy $w(T, h(T)) = 3^{h(T)} = 3^{\log_4 n} = n^{\log_4 3}$ podúloh.

Příklad

(4) Vypočti složitost výpočtu podúlohy $c(i)$ v uzlu v hloubce i.

- Ze zadání plyne $c(0) = kn^2$.
- V hloubce 1 je $c(1) = k(n/4)^2$.
- V hloubce i je $c(i) = k(n/4^i)^2$.
- V hloubce $h(T)$ je $c(h(T)) = \Theta(1)$ (tak byla hloubka stromu T určená).

(5) Sečti složitost výpočtů podúloh $cc(i) = w(T, i) * c(i)$ v uzlech v hloubce i.
Speciálně sečti složitosti uzlů v posledním hladině stromu.

- Triviálním násobením vyjde $cc(i) = \left(\frac{3}{16}\right)^i kn^2$ pro $i < h(T)$.
- Celková složitost v poslední hladině je $cc(h(T)) = n^{\log_4 3}\Theta(1) = \Theta(n^{\log_4 3}) = o(n)$.

Příklad

(6) Sečti složitosti výpočtů přes celý strom (všechny hladiny).

$$\begin{aligned}
t(n) &= kn^2 + \tfrac{3}{16}kn^2 + \left(\tfrac{3}{16}\right)^2 kn^2 + \cdots + \left(\tfrac{3}{16}\right)^{\log_4 n} kn^2 + \Theta(n^{\log_4 3}) \\
&\leq kn^2 \left(\textstyle\sum_{i=0}^{\infty} \left(\tfrac{3}{16}\right)^i\right) + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-\frac{3}{16}} kn^2 + \Theta(n^{\log_4 3}) \\
&= \tfrac{16}{13} kn^2 + o(n) = \Theta(n^2).
\end{aligned}$$

Mistrovská metoda

Věta

Nechť $a \geq 1$ *a* $b > 1$ *jsou konstanty,* $f(n)$ *funkce jedné proměnné. Uvažujme rekurentní rovnici*

$$t(n) = at(n/b) + f(n), \tag{19}$$

kde n/b *může znamenat buď* $\lfloor n/b \rfloor$ *nebo* $\lceil n/b \rceil$ *a tyto rozdíly zanedbáváme. Pak* $t(n)$ *má následující asymptotické řešení:*

(1) *Pokud* $f(n) = O(n^{\log_b a - \varepsilon})$ *pro nějakou konstantu* $\varepsilon > 0$*, pak* $t(n) = \Theta(n^{\log_b a})$.

(2) *Pokud* $f(n) = \Theta(n^{\log_b a})$*, pak* $t(n) = \Theta(n^{\log_b a} \log n)$.

(3) *Pokud* $f(n) = \Omega(n^{\log_b a + \varepsilon})$ *pro nějakou konstantu* $\varepsilon > 0$ *a pokud* $af(n/b) \leq cf(n)$ *pro nějakou konstantu* $c < 1$ *a všechny* $n \geq n_0$*, pak* $t(n) = \Theta(f(n))$.

Důkaz. Dlouhý, vynechán.

Diskuze Mistrovské metody

- Rovnice odpovídá rekurzivnímu algoritmu, který dělí problém o velikosti n na a částí o velikosti n/b a na dělení a zpětné kombinování výsledků potřebuje $f(n)$ času.
- Ve všech 3 případech se $f(n)$ srovnává s $n^{\log_b a}$ a řešení je dáno větším z nich.
  (1) $f(n)$ je **polynomiálně menší** než $n^{\log_b a}$, která se nakonec prosadí.
  (2) Obě funkce jsou stejného řádu a proto výsledkem je $\Theta(f(n)\log n) = \Theta(n^{\log_b a}\log n)$.
  (3) Opak případu (1), $f(n)$ je **polynomiálně větší** než $n^{\log_b a}$.
- Tyto 3 případy **nepokrývají** všechny případy.
  - Pokud je rozdíl mezi funkcemi menší než polynomiální, nelze tuto metodu použít!!!!!!

Příklad

Rovnice $t(n) = 2t(n/2) + n\log n$ není mistrovkou metodou řešitelná.

Příklady aplikace Mistrovské metody

Příklad

- Rovnice

$$t(n) = 6t(n/4) + n$$

$a = 6,\ b = 4,\ n^{\log_4 6} \doteq n^{1.3} = \Omega(n) \quad \Rightarrow \quad f(n) = O(n^{\log_4 6 - 0,1})$
$\Rightarrow$ případ (1). Čili

$$t(n) = \Theta(n^{\log_4 6}).$$

- Rovnice (MergeSort)

$$t(n) = 2t(n/2) + n$$

$a = 2,\ b = 2,\ n^{\log_2 2} = n = \Theta(n) \quad \Rightarrow$ případ (2). Čili

$$t(n) = \Theta(n \log n).$$

Příklad

- Rovnice
$$t(n) = 3t(n/4) + n$$
$a = 3$, $b = 4$, $n^{\log_4 3} \doteq n^{0,7} = o(n)$ a platí, že $3 \cdot (n/4) \leq cn$ pro nějakou $c < 1$ $\Rightarrow$ případ (3). Čili
$$t(n) = \Theta(n).$$

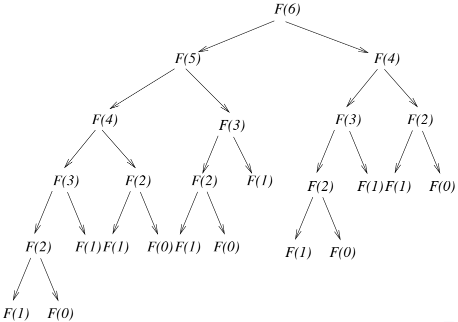
- Rovnice
$$t(n) = 3t(n/4) + n^2$$
$a = 3$, $b = 4$, $n^{\log_4 3} \doteq n^{0,7} = o(n^2)$ a platí, že $3 \cdot \left(\frac{n}{4}\right)^2 \leq cn^2$ pro nějakou $c < 1$ $\Rightarrow$ případ (3). Čili
$$t(n) = \Theta(n^2).$$

Fibonnacciho posloupnost

$$F(n) = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases} \tag{20}$$

Označme $F(i) = f_i$. Fibonacciho posloupnost $F = \langle f_i \rangle_{i=0}^{\infty}$ je:

$$F = \langle 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots \rangle$$



Složitost rekurzivního výpočtu Fibonnacciho posloupnosti

Složitost rekurzivního algoritmu pro výpočet Fibonacciho čísla f_n je

$$t(n) = t(n-1) + t(n-2) + \Theta(1), \tag{21}$$

kde pro jednoduchost položíme $t(0) = t(1) = 1$. Ukážeme nejprve řešení pro **homogenní rovnici**

$$t(n) = t(n-1) + t(n-2), \tag{22}$$

která je **téměř** identická s definiční rovnicí pro výpočet Fibonacciho čísel, odlišnost je pouze v počáteční podmínce pro $n = 0$, srovnej s (20).

$$F(n) = F(n-1) + F(n-2). \tag{23}$$

Operátory nad posloupnostmi

Definice

Uvažujme dvě celočíselné nekonečné posloupnosti $X = \langle x_i \rangle_{i=0}^{\infty}$ *a* $Y = \langle y_i \rangle_{i=0}^{\infty}$. *Definujme nad nimi následující* **binární a unární operátory** *(funkce):*

- **Sčítání** *dvou posloupností:* $X + Y = \langle x_i + y_i \rangle_{i=0}^{\infty}$.
- **Odčítání** *dvou posloupností:* $X - Y = \langle x_i - y_i \rangle_{i=0}^{\infty}$.
- **Násobení** *posloupnosti konstantou* a*:* $aX = \langle a x_i \rangle_{i=0}^{\infty}$.
- **Posun** *posloupnosti doleva a odstranění prvního prvku:* $\mathbf{E}X = \langle x_i \rangle_{i=1}^{\infty}$.

Příklad

Nechť $D = \langle 2^i \rangle_{i=0}^{\infty} = \langle 1, 2, 2^2, 2^3, 2^4, \ldots \rangle$.

Pak $2D = \langle 2, 2^2, 2^3, 2^4, \ldots \rangle$.

$\mathbf{E}D = \langle 2, 2^2, 2^3, 2^4, \ldots \rangle$.

Je zřejmé, že $2D = \mathbf{E}D$ a proto

$\mathbf{E}D - 2D = \langle 0, 0, 0, 0, 0, \ldots \rangle$ je **nulová** posloupnost.

Anihilátory

Věta

Operátory $\mathbf{E}$ *a* a *jsou distributivní vzhledem k + a −.*

Příklad

$\mathbf{E}X + aX = (\mathbf{E} + a)X.$

Definice

Je dána posloupnost $X = \langle x_i \rangle_{i=0}^{\infty}$*. Pak operátor* O*, pro který platí, že* OX *je nulová, je* ***anihilátorem*** X*.*

Příklad

- Operátor $(\mathbf{E} - 2)$ je anihilátorem posloupnosti D.
- Operátor $(\mathbf{E} - 1)$ je anihilátorem jakékoli konstantní posloupnosti $\langle c, c, c, c, \ldots \rangle$.

Anihilátory

Věta

Mějme dány nenulové konstanty a, b, c a celé číslo $k \geq 0$, Pak operátor $(\mathbf{E} - b)$ je anihilátorem geometrické posloupnosti $A = \langle ca^i \rangle_{i=k}^{\infty}$ právě když $b = a$.

Důkaz. Zvolme bez ztráty obecnosti $k = 0$ a $A = \langle c, ca, ca^2, ca^3, \ldots \rangle$.

$(\mathbf{E} - a)A = \mathbf{E}A - aA = \langle ca, ca^2, ca^3, \ldots \rangle - \langle ca, ca^2, ca^3, \ldots \rangle = \langle 0, 0, 0, 0, 0, \ldots \rangle$.

Nechť $b \neq a$. Pak

$(\mathbf{E}-b)A = \mathbf{E}A - bA = \langle ca, ca^2, ca^3, \ldots \rangle - \langle bc, bca, bca^2, \ldots \rangle = (a-b)A$.

■

Důsledek

Pokud operátor $(\mathbf{E} - a)$ anihiluje posloupnost A, pak A musí být tvaru $\langle ca^i \rangle_{i=k}^{\infty}$ pro nějaké konstanty $c > 0$ a $k \geq 0$.

Skládání operátorů

Protože operátory jsou v podstatě deterministické funkce nad posloupnostmi, lze je skládat (kombinovat).

Definice

$(O_1O_2)X = O_1(O_2X)$.

Příklad

- $c(dX) = (cd)X$.
- $\mathbf{E}^2X = \mathbf{E}(\mathbf{E}X) = \langle x_i \rangle_{i=2}^{\infty}$.
- $(\mathbf{E} - a)(\mathbf{E} - b)X = \mathbf{E}^2X - (a + b)\mathbf{E}X + (ab)X$.

Složené anihilátory

Z předchozího vyplývá, že na anihilaci kombinované posloupnosti $\langle c_1 a^i + c_2 b^i \rangle_{i=0}^{\infty}$ potřebujeme postupně aplikovat (v libovolném pořadí) anihilátory $(\mathbf{E} - a)$ a $(\mathbf{E} - b)$. Každý z nich si vezme na starost svou část a na druhou nemá vliv.

Důsledek

Pro $a \neq b$, pokud operátor $(\mathbf{E} - a)(\mathbf{E} - b)$ anihiluje posloupnost A, pak A musí být tvaru $\langle c_1 a^i + c_2 b^j \rangle_{i=k,j=l}^{\infty}$ pro nějaké konstanty c_1, c_2, k, l.

Věta

Anihilátorem Fibonacciho posloupnosti F je $\mathbf{E}^2 - \mathbf{E} - 1$.

Důkaz.

$$\begin{array}{lclcl}
\mathbf{E}^2 F & = & \langle f_{i+2} \rangle_{i=0}^{\infty} & = & \langle 1, 2, 3, 5, 8, 13, \ldots \rangle. \\
\mathbf{E} F & = & \langle f_{i+1} \rangle_{i=0}^{\infty} & = & \langle 1, 1, 2, 3, 5, 8, \ldots \rangle. \quad . \\
F & = & \langle f_i \rangle_{i=0}^{\infty} & = & \langle 0, 1, 1, 2, 3, 5, \ldots \rangle.
\end{array}$$

Tudíž $(\mathbf{E}^2 - \mathbf{E} - 1)F = \langle f_{i+2} - f_{i+1} - f_i \rangle_{i=0}^{\infty} = \langle 0, 0, 0, 0, 0, \ldots \rangle$.

Nerekurzivní výraz pro hodnotu Fibonacciho čísel

1. Rozklad polynomu $\mathbf{E}^2 - \mathbf{E} - 1$. Jedná se o obyčejnou kvadratickou rovnici:
$$\mathbf{E}^2 - \mathbf{E} - 1 = (\mathbf{E} - \phi)(\mathbf{E} - \hat{\phi}),$$
kde $\phi = (1 + \sqrt{5})/2 \doteq 1.618034$ (hodnota **zlatého řezu**) a $\hat{\phi} = (1 - \sqrt{5})/2 = 1 - \phi = -1/\phi$.
2. Generická podoba posloupnosti F je tedy
$$\langle c_1\phi^i + c_2\hat{\phi}^j \rangle_{i=k,j=l}^{\infty}$$
pro kterou lze počáteční podmínky $f_0 = 0$ a $f_1 = 1$ nejjednodušeji splnit volbou $k = l = 0$ a kdy konstanty c_1 a c_2 vyjdou $c_1 = 1/\sqrt{5}$ a $c_2 = -1/\sqrt{5}$.
3. Výsledný tvar je
$$f_n = F(n) = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}} = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n. \quad (24)$$

Složitost kaskádní rekurze

- Rovnice (23) je tím vyřešena.
- Rovnice (21) pro složitost rekurzivního algoritmu pro výpočet Fibonacciho čísel se liší aditivními konstantami. Předpokládejme

$$t(n) = \begin{cases} 1 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ t(n-1) + t(n-2) + k & \text{if } n > 1. \end{cases} \tag{25}$$

- Anihilátorem je $(\mathbf{E} - \phi)(\mathbf{E} - \hat{\phi})(\mathbf{E} - 1)$.
- Generické řešení je tedy

$$t(n) = c_1\phi^n + c_2\hat{\phi}^n + c_3.$$

- Konstanty lze vypočítat z okrajových podmínek pro $n = 0, 1, 2$.
- Řešení je překvapivě jednoduché:

$$t(n) = (k+1)F(n+1) - k = \frac{k+1}{\sqrt{5}}\left(\phi^{n+1} - \hat{\phi}^{n+1}\right) - k$$