Matematika III – 11. přednáška
Toky v sítích, párování

Michal Bulant

Masarykova univerzita
Fakulta informatiky

9. 12. 2009

Obsah přednášky

1. Toky v sítích
2. Problém maximálního toku v síti
3. Další aplikace
   - Bipartitní párování
   - Stromové datové struktury a prefixové kódy
4. Vytvořující funkce
5. Operace s vytvořujícími funkcemi
   - Přehled mocninných řad

Doporučené zdroje

- Martin Panák, Jan Slovák, **Drsná matematika**, *e*-text.
- *Předmětové záložky v IS MU*

Doporučené zdroje

- Martin Panák, Jan Slovák, **Drsná matematika**, *e*-text.
- *Předmětové záložky v IS MU*
- Jiří Matoušek, Jaroslav Nešetřil, **Kapitoly z diskrétní matematiky**, Univerzita Karlova v Praze, Karolinum, Praha, 2000, 377 s.
- Petr Hliněný, **Teorie grafů**, studijní materiály, `http://www.fi.muni.cz/~{}hlineny/Vyuka/GT/`

Doporučené zdroje

- Martin Panák, Jan Slovák, **Drsná matematika**, *e*-text.
- *Předmětové záložky v IS MU*
- Jiří Matoušek, Jaroslav Nešetřil, **Kapitoly z diskrétní matematiky**, Univerzita Karlova v Praze, Karolinum, Praha, 2000, 377 s.
- Petr Hliněný, **Teorie grafů**, studijní materiály, `http://www.fi.muni.cz/~{}hlineny/Vyuka/GT/`
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest a Clifford Stein , **Introduction to Algorithms**, MIT Press, 2001.
  H. S. Wilf, **Generatingfunctionology**, Academic Press, druhé vydání, 1994 , (rovněž `http://www.math.upenn.edu/~wilf/DownldGF.html`)

Doporučené zdroje

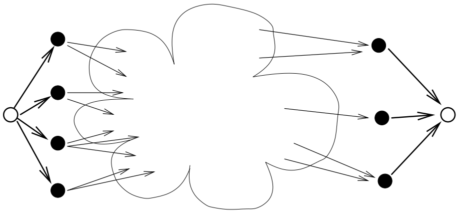
- Martin Panák, Jan Slovák, **Drsná matematika**, *e*-text.
- *Předmětové záložky v IS MU*
- Jiří Matoušek, Jaroslav Nešetřil, **Kapitoly z diskrétní matematiky**, Univerzita Karlova v Praze, Karolinum, Praha, 2000, 377 s.
- Petr Hliněný, **Teorie grafů**, studijní materiály, `http://www.fi.muni.cz/~{}hlineny/Vyuka/GT/`
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest a Clifford Stein , **Introduction to Algorithms**, MIT Press, 2001.
  H. S. Wilf, **Generatingfunctionology**, Academic Press, druhé vydání, 1994 , (rovněž `http://www.math.upenn.edu/~wilf/DownldGF.html`)
- R. Graham, D. Knuth, O. Patashnik, **Concrete Mathematics**, druhé vydání, Addison-Wesley, 1994.

Plán přednášky

1. Toky v sítích
2. Problém maximálního toku v síti
3. Další aplikace
   - Bipartitní párování
   - Stromové datové struktury a prefixové kódy
4. Vytvořující funkce
5. Operace s vytvořujícími funkcemi
   - Přehled mocninných řad

Další významná skupina aplikací jazyka teorie grafů se týká přesunu nějakého měřitelného materiálu v pevně zadané síti. Vrcholy v orientovaném grafu představují body, mezi kterými lze podél hran přenášet předem známá množství, která jsou zadána formou ohodnocení hran. Některé vybrané vrcholy představují **zdroj sítě**, jiné výstup ze sítě. Podle analogie potrubní sítě pro přenos kapaliny říkáme výstupním vrcholům **stok sítě**). Síť je tedy pro nás orientovaný graf s ohodnocenými hranami a vybranými vrcholy, kterým říkáme zdroje a stoky.

Je zřejmé, že se můžeme bez újmy na obecnosti omezit na orientované grafy s **jedním zdrojem** a **jedním stokem**.
V obecném případě totiž vždy můžeme přidat jeden stok a jeden zdroj navíc a spojit je vhodně orientovanými hranami se všemi zadanými zdroji a stoky tak, že ohodnocení přidaných hran bude zároveň zadávat maximální kapacity jednotlivých zdrojů a stoků.
Situace je naznačena na obrázku, kde černými vrcholy nalevo jsou zobrazeny všechny zadané zdroje, zatímco černé vrcholy napravo jsou všechny zadané stoky. Nalevo je jeden přidaný (virtuální) zdroj jako bílý vrchol a napravo jeden stok. Označení hran není v obrázku uvedeno.



Definice

Síť (*flow network*)je orientovaný graf $G = (V, E)$ s vybraným jedním vrcholem z nazvaným **zdroj** a jiným vybraným vrcholem s nazvaným **stok**, spolu s nezáporným ohodnocením hran $w : E \to \mathbb{R}_0^+$, nazývaným **kapacita hran**.

Definice

Síť (*flow network*)je orientovaný graf $G = (V, E)$ s vybraným jedním vrcholem z nazvaným **zdroj** a jiným vybraným vrcholem s nazvaným **stok**, spolu s nezáporným ohodnocením hran $w : E \to \mathbb{R}_0^+$, nazývaným **kapacita hran**. **Tokem** v síti $S = (V, E, z, s, w)$ rozumíme ohodnocení hran $f : E \to \mathbb{R}$ takové, že součet hodnot u vstupních hran u každého vrcholu v kromě zdroje a stoku je stejný jako součet u výstupních hran z téhož vrcholu (někdy nazýváno *Kirchhoffův zákon*), tj.

$$v \neq z, s \implies \sum_{e \in IN(v)} f(e) = \sum_{e \in OUT(v)} f(e)$$

a tok splňuje *kapacitní omezení* $f(e) \leq w(e)$.

Definice

Síť (*flow network*)je orientovaný graf $G = (V, E)$ s vybraným jedním vrcholem z nazvaným **zdroj** a jiným vybraným vrcholem s nazvaným **stok**, spolu s nezáporným ohodnocením hran $w : E \to \mathbb{R}_0^+$, nazývaným **kapacita hran**. **Tokem** v síti $S = (V, E, z, s, w)$ rozumíme ohodnocení hran $f : E \to \mathbb{R}$ takové, že součet hodnot u vstupních hran u každého vrcholu v kromě zdroje a stoku je stejný jako součet u výstupních hran z téhož vrcholu (někdy nazýváno *Kirchhoffův zákon*), tj.

$$v \neq z, s \implies \sum_{e \in IN(v)} f(e) = \sum_{e \in OUT(v)} f(e)$$

a tok splňuje *kapacitní omezení* $f(e) \leq w(e)$. **Velikost toku** f je dána celkovou balancí hodnot u zdroje

$$|f| = \sum_{e \in OUT(z)} f(e) - \sum_{e \in IN(z)} f(e).$$

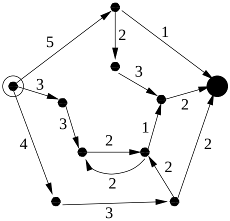
Z definice je zřejmé, že velikost toku můžeme stejně dobře vypočíst jako hodnotu

$$|f| = \sum_{e \in IN(s)} f(e) - \sum_{e \in OUT(s)} f(e).$$

Z definice je zřejmé, že velikost toku můžeme stejně dobře vypočíst jako hodnotu

$$|f| = \sum_{e \in IN(s)} f(e) - \sum_{e \in OUT(s)} f(e).$$

Na obrázku máme nakreslenu jednoduchou síť se zvýrazněným bílým zdrojem a černým stokem. Součtem maximálních kapacit hran vstupujících do stoku vidíme, že maximální možný tok v této síti je 5.



Plán přednášky

1. Toky v sítích
2. Problém maximálního toku v síti
3. Další aplikace
   - Bipartitní párování
   - Stromové datové struktury a prefixové kódy
4. Vytvořující funkce
5. Operace s vytvořujícími funkcemi
   - Přehled mocninných řad

Naší úlohou bude pro zadanou síť na grafu G určit maximální možný tok. Jde vlastně o speciální případ úlohy lineárního (celočíselného) programování, kde neznámými jsou toky na hranách a omezení plynou z podmínek na tok. Ukáže se, že pro řešení této úlohy existují jednoduché a přitom rychlé algoritmy.

Naší úlohou bude pro zadanou síť na grafu G určit maximální možný tok. Jde vlastně o speciální případ úlohy lineárního (celočíselného) programování, kde neznámými jsou toky na hranách a omezení plynou z podmínek na tok. Ukáže se, že pro řešení této úlohy existují jednoduché a přitom rychlé algoritmy.

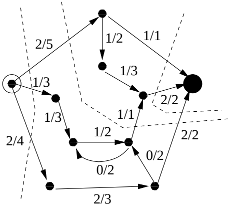
Definice

Řezem v síti $S = (V, E, z, s, w)$ rozumíme takovou množinu hran $C \subset E$, že po jejím odebrání nebude v grafu $G = (V, E \setminus C)$ žádná (orientovaná) cesta z z do s. Číslo

$$|C| = \sum_{e \in C} w(e)$$

nazýváme **kapacita (velikost) řezu** C.

Evidentně platí, že nikdy nemůžeme najít větší tok, než je kapacita kteréhokoliv z řezů. Na dalším obrázku máme zobrazen tok sítí s hodnotou 5 a čárkovanými lomenými čarami jsou naznačeny řezy o hodnotách 12, 8 a 5.



Poznámka

Tok a kapacitu hran v síti obvykle zapisujeme v obrázku ve tvaru f/c, kde f je hodnota toku na dané hraně a c její kapacita.

Sestavíme algoritmus, který pomocí postupných konstrukcí vhodných cest najde řez s minimální možnou hodnotou a zároveň najde tok, který tuto hodnotu realizuje. Tím dokážeme následující větu:

Věta

Maximální velikost toku v dané síti $S = (V, E, z, s, w)$ je rovna minimální kapacitě řezu v této síti.

Myšlenka algoritmu – prohledáváme cesty mezi uzly grafu a snažíme se je *nasytit* co největším tokem. Zavedeme si za tímto účelem terminologii.

Myšlenka algoritmu – prohledáváme cesty mezi uzly grafu a snažíme se je *nasytit* co největším tokem. Zavedeme si za tímto účelem terminologii. O **neorientované** cestě v síti $S = (V, E, z, s, w)$ z vrcholu v do vrcholu w řekneme, že je **nenasycená**, jestliže pro všechny hrany této cesty orientované ve směru z v do w platí $f(e) < w(e)$ a $f(e) > 0$ pro hrany orientované opačně. Za **rezervu kapacity** hrany e pak označujeme číslo $w(e) - f(e)$ pro případ hrany orientované ve směru z v do w a číslo $f(e)$ při orientaci opačné. Pro zvolenou cestu bereme za rezervu kapacity minimální rezervu kapacity z jejích hran.

Ford-Fulkersonův algoritmus (1956)

Vstupem je síť $S = (V, E, z, s, w)$ a výstupem maximální možný tok $f : E \to \mathbb{R}$.

- *Inicializace:* zadáme $f(e) = 0$ pro všechny hrany $e \in E$ a najdeme množinu vrcholů $U \subset V$, do kterých existuje nenasycená cesta ze zdroje z.

Ford-Fulkersonův algoritmus (1956)

Vstupem je síť $S = (V, E, z, s, w)$ a výstupem maximální možný tok $f : E \to \mathbb{R}$.

- *Inicializace:* zadáme $f(e) = 0$ pro všechny hrany $e \in E$ a najdeme množinu vrcholů $U \subset V$, do kterých existuje nenasycená cesta ze zdroje z.
- *Hlavní cyklus:* Dokud $s \in U$ opakujeme

Ford-Fulkersonův algoritmus (1956)

Vstupem je síť $S = (V, E, z, s, w)$ a výstupem maximální možný tok $f : E \to \mathbb{R}$.

- *Inicializace:* zadáme $f(e) = 0$ pro všechny hrany $e \in E$ a najdeme množinu vrcholů $U \subset V$, do kterých existuje nenasycená cesta ze zdroje z.
- *Hlavní cyklus:* Dokud $s \in U$ opakujeme
  - zvolíme nenasycenou cestu P ze zdroje z do s a zvětšíme tok f u všech hran této cesty o její minimální rezervu

Ford-Fulkersonův algoritmus (1956)

Vstupem je síť $S = (V, E, z, s, w)$ a výstupem maximální možný tok $f : E \to \mathbb{R}$.

- *Inicializace:* zadáme $f(e) = 0$ pro všechny hrany $e \in E$ a najdeme množinu vrcholů $U \subset V$, do kterých existuje nenasycená cesta ze zdroje z.
- *Hlavní cyklus:* Dokud $s \in U$ opakujeme
  - zvolíme nenasycenou cestu P ze zdroje z do s a zvětšíme tok f u všech hran této cesty o její minimální rezervu
  - aktualizujeme U.

Ford-Fulkersonův algoritmus (1956)

Vstupem je síť $S = (V, E, z, s, w)$ a výstupem maximální možný tok $f : E \to \mathbb{R}$.

- *Inicializace:* zadáme $f(e) = 0$ pro všechny hrany $e \in E$ a najdeme množinu vrcholů $U \subset V$, do kterých existuje nenasycená cesta ze zdroje z.
- *Hlavní cyklus:* Dokud $s \in U$ opakujeme
  - zvolíme nenasycenou cestu P ze zdroje z do s a zvětšíme tok f u všech hran této cesty o její minimální rezervu
  - aktualizujeme U.
- na výstup dáme maximální tok f a minimální řez C tvořený všemi hranami vycházejícími z U a končícími v doplňku $V \setminus U$.

Důkaz správnosti algoritmu

Jak jsme viděli, velikost každého toku je nejvýše rovna kapacitě kteréhokoliv řezu. Stačí nám tedy ukázat, že v okamžiku zastavení algoritmu jsme vygenerovali řez i tok se stejnou hodnotou. Algoritmus se zastaví, jakmile neexistuje nenasycená cesta ze zdroje z do stoku s. To znamená, že U neobsahuje s a pro všechny hrany e z U do zbytku je $f(e) = w(e)$, jinak bychom museli koncový vrchol e přidat k U.

Důkaz správnosti algoritmu

Jak jsme viděli, velikost každého toku je nejvýše rovna kapacitě kteréhokoliv řezu. Stačí nám tedy ukázat, že v okamžiku zastavení algoritmu jsme vygenerovali řez i tok se stejnou hodnotou.
Algoritmus se zastaví, jakmile neexistuje nenasycená cesta ze zdroje z do stoku s. To znamená, že U neobsahuje s a pro všechny hrany e z U do zbytku je $f(e) = w(e)$, jinak bychom museli koncový vrchol e přidat k U.
Zároveň ze stejného důvodu všechny hrany e, které začínají v komplementu $V \setminus U$ a končí v U musí mít tok $f(e) = 0$.

Pro velikost toku celé sítě jistě platí

$$|f| = \sum_{\text{hrany z } U \text{ do } V \setminus U} f(e) - \sum_{\text{hrany z } V \setminus U \text{ do } U} f(e).$$

Tento výraz je ovšem v okamžiku zastavení roven

$$\sum_{\text{hrany z } U \text{ do } V \setminus U} f(e) = \sum_{\text{hrany z } U \text{ do } V \setminus U} w(e) = |C|,$$

což jsme chtěli dokázat.

Pro velikost toku celé sítě jistě platí

$$|f| = \sum_{\text{hrany z } U \text{ do } V \setminus U} f(e) - \sum_{\text{hrany z } V \setminus U \text{ do } U} f(e).$$

Tento výraz je ovšem v okamžiku zastavení roven

$$\sum_{\text{hrany z } U \text{ do } V \setminus U} f(e) = \sum_{\text{hrany z } U \text{ do } V \setminus U} w(e) = |C|,$$

což jsme chtěli dokázat.
Zbývá ovšem ukázat, že algoritmus skutečně zastaví.

Zastavení Ford-Fulkersonova algoritmu

Tvrzení

Pro celočíselné kapacity hran sítě uvedený algoritmus vždy skončí. V obecném případě nejen, že algoritmus skončit nemusí, příslušné toky dokonce ani nemusí k maximálnímu toku konvergovat.

Zastavení Ford-Fulkersonova algoritmu

Tvrzení

Pro celočíselné kapacity hran sítě uvedený algoritmus vždy skončí. V obecném případě nejen, že algoritmus skončit nemusí, příslušné toky dokonce ani nemusí k maximálnímu toku konvergovat.

Důkaz.

Důkaz ukončení v celočíselném případě vyplývá z toho, že vždy sytíme cestu o celočíselné hodnotě. □

Zastavení Ford-Fulkersonova algoritmu

Tvrzení

Pro celočíselné kapacity hran sítě uvedený algoritmus vždy skončí. V obecném případě nejen, že algoritmus skončit nemusí, příslušné toky dokonce ani nemusí k maximálnímu toku konvergovat.

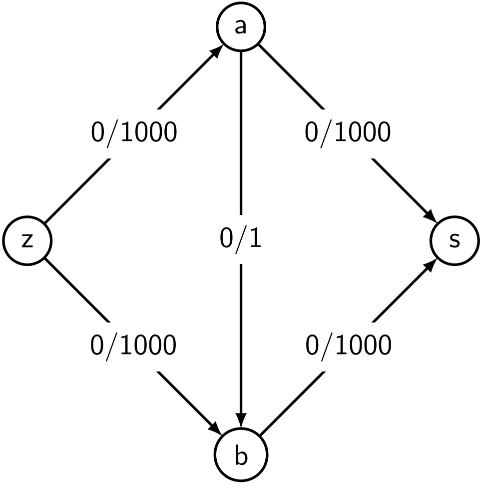
Důkaz.

Důkaz ukončení v celočíselném případě vyplývá z toho, že vždy sytíme cestu o celočíselné hodnotě. □

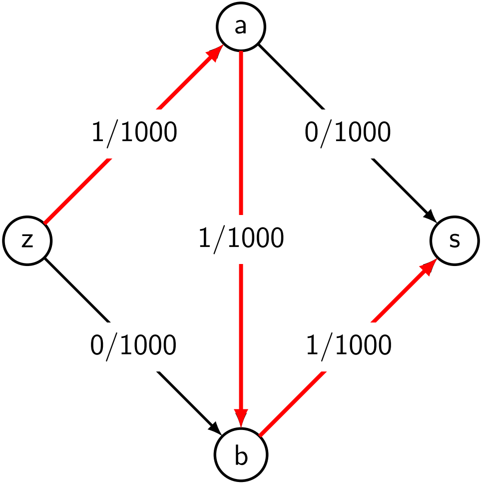
Poznámka

Ford-Fulkersonův algoritmus má složitost v nejhorším případě $O(E \cdot |f|)$, kde $|f|$ je hodnota maximálního toku.

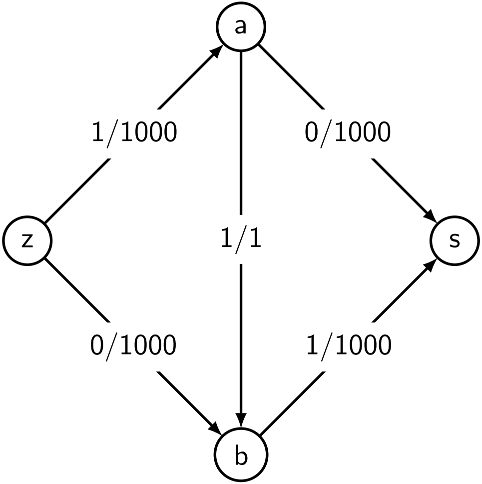
Příklad špatného chování F-F algoritmu



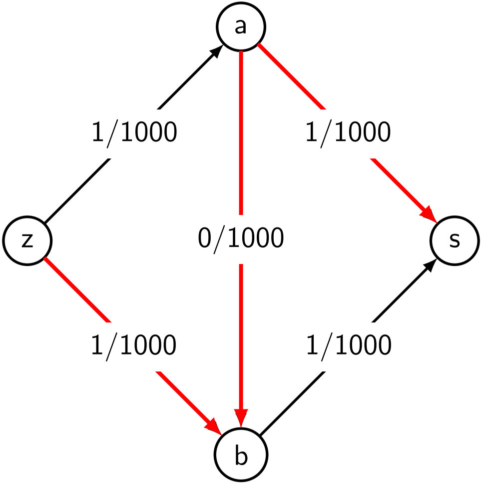
Příklad špatného chování F-F algoritmu



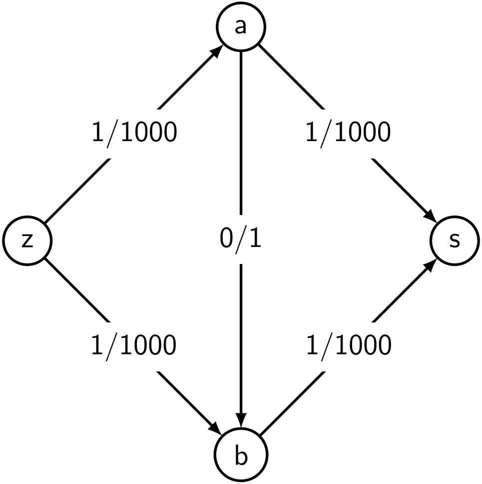
Příklad špatného chování F-F algoritmu



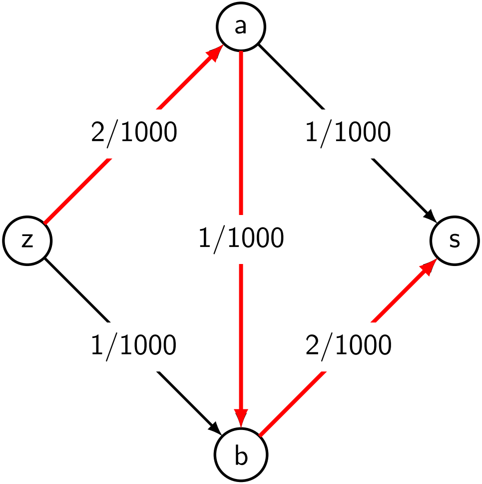
Příklad špatného chování F-F algoritmu



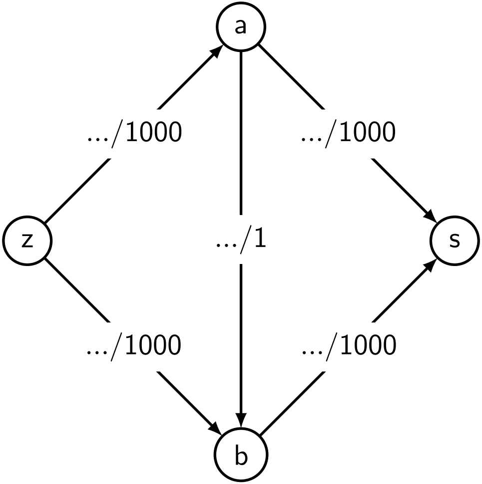
Příklad špatného chování F-F algoritmu



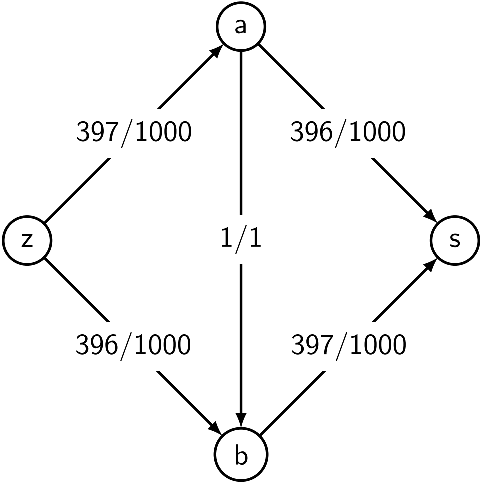
Příklad špatného chování F-F algoritmu



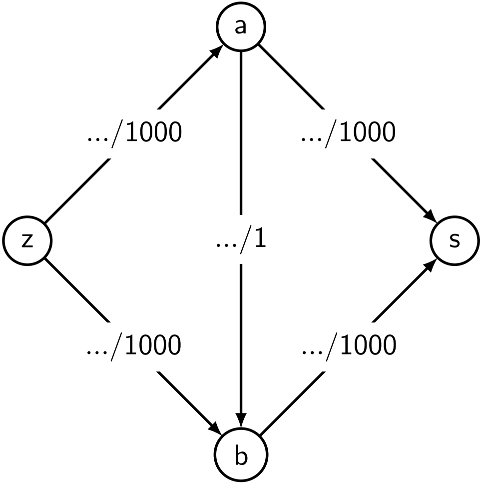
Příklad špatného chování F-F algoritmu



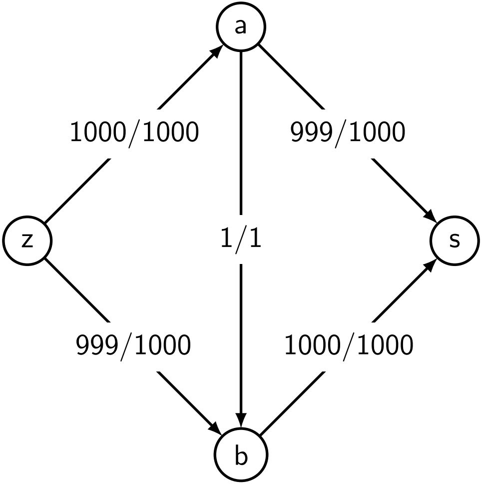
Příklad špatného chování F-F algoritmu



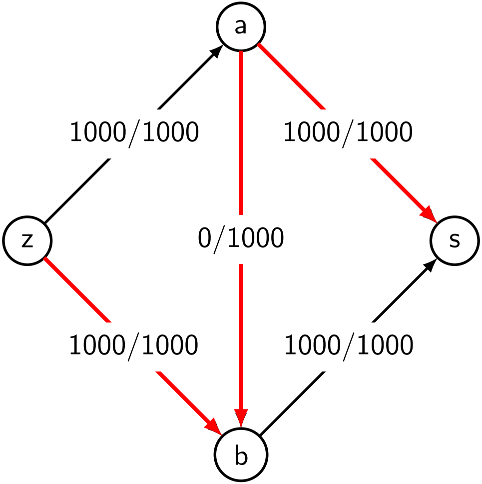
Příklad špatného chování F-F algoritmu



Příklad špatného chování F-F algoritmu

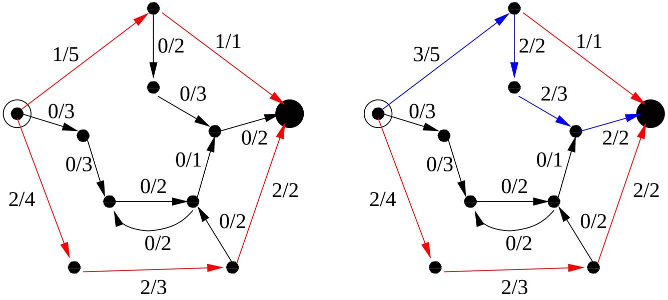


Příklad špatného chování F-F algoritmu



Edmonds-Karpův algoritmus

Vylepšením základního Ford-Fulkersonova algoritmu je *Edmonds-Karpův algoritmus*, ve kterém zvětšujeme tok podél *nejkratší* nenasycené cesty – tj. síť prohledáváme **do šířky**.
U tohoto algoritmu již je zaručeno zastavení, navíc je jeho časová složitost ohraničena $O(VE^2)$, nezávisle na hodnotě maximálního toku.



Chod algoritmu je ilustrován na obrázku. Vlevo jsou vybarveny dvě nejkratší nenasycené cesty ze zdroje do stoku (horní má dvě hrany, spodní tři). Jsou vyznačeny červeně. Napravo je pak nasycena další cesta v pořadí a je vyznačena modře. Je nyní zjevné, že nemůže existovat další nenasycená cesta ze zdroje do stoku. Proto algoritmus v tomto okamžiku skončí.

Moderní algoritmy pro maximální tok

Viz např. Petr Parubják, XXVI. ASR '2001 Seminar, Instruments and Control, Ostrava, April 26 - 27, 2001 (`http://www.fs.vsb.cz/akce/2001/asr2001/Proceedings/papers/55.pdf`

Diničův algoritmus

Zjednodušuje hledání nenasycené cesty konstrukcí tzv. *úrovňového grafu*, kdy *zlepšující hrany* uvažujeme pouze tehdy, pokud vedou mezi vrcholy různých vzdáleností od zdroje.
Složitost je $O(V^2E)$, což je u hustých grafů významné vylepšení oproti složitosti $O(VE^2)$ algoritmu Edmonds-Karpa.

Moderní algoritmy pro maximální tok

Viz např. Petr Parubják, XXVI. ASR '2001 Seminar, Instruments and Control, Ostrava, April 26 - 27, 2001 (`http://www.fs.vsb.cz/akce/2001/asr2001/Proceedings/papers/55.pdf`

Diničův algoritmus

Zjednodušuje hledání nenasycené cesty konstrukcí tzv. *úrovňového grafu*, kdy *zlepšující hrany* uvažujeme pouze tehdy, pokud vedou mezi vrcholy různých vzdáleností od zdroje.
Složitost je $O(V^2E)$, což je u hustých grafů významné vylepšení oproti složitosti $O(VE^2)$ algoritmu Edmonds-Karpa.

MPM algoritmus

Malhotra, Pramodh Kumar a Maheshwari přišli s algoritmem složitosti $O(V^3)$. Konstruují stejným způsobem úrovňový graf, ale vylepšují hledání maximální toku v tomto úrovňovém grafu.

Viz `http://www.cosc.canterbury.ac.nz/tad.takaoka/alg/`

Zobecnění sítí

V praktických aplikacích mohou být na sítě kladeny další požadavky:

- maximální kapacita vrcholů – snadno se převede na základní případ *zdvojením* vrcholů, které spojíme hranou o dané kapacitě;
- minimální kapacita hran – např. aby nedocházelo k zanášení potrubí. V tomto případě lze modifikovat inicializaci, je ale třeba otestovat existenci přípustného toku (podobně jako v úloze lineárního progamování).

Plán přednášky

1. Toky v sítích
2. Problém maximálního toku v síti
3. Další aplikace
   - Bipartitní párování
   - Stromové datové struktury a prefixové kódy
4. Vytvořující funkce
5. Operace s vytvořujícími funkcemi
   - Přehled mocninných řad

Věta (Mengerova)

Pro každé dva vrcholy v a w v grafu $G = (V, E)$ je počet hranově různých cest z v do w roven minimálnímu počtu hran, které je třeba odstranit, aby se v a w ocitly v různých komponentách vzniklého grafu.

Důkaz.

Plyne snadno z věty o maximálním toku a minimálním řezu v síti, kde jsou kapacity všech hran (v obou směrech) rovny 1. $\square$

Bipartitní párování

Hezkým využitím toků v sítí je řešení úlohy bipartitního párování. Úkolem je najít v bipartitním grafu maximální podmnožinu hran takovou, aby žádné dvě hrany nesdílely vrchol.

Bipartitní párování

Hezkým využitím toků v sítí je řešení úlohy bipartitního párování. Úkolem je najít v bipartitním grafu maximální podmnožinu hran takovou, aby žádné dvě hrany nesdílely vrchol.
Jde o abstraktní variantu docela obvyklé úlohy – třeba spárování kluků a holek k tanci v tanečních, kdybychom měli předem známé možnosti, ze kterých vybíráme (např. aby dvojici netvořil pár příliš výškově nesourodý).

Bipartitní párování

Hezkým využitím toků v sítí je řešení úlohy bipartitního párování. Úkolem je najít v bipartitním grafu maximální podmnožinu hran takovou, aby žádné dvě hrany nesdílely vrchol.
Jde o abstraktní variantu docela obvyklé úlohy – třeba spárování kluků a holek k tanci v tanečních, kdybychom měli předem známé možnosti, ze kterých vybíráme (např. aby dvojici netvořil pár příliš výškově nesourodý).
Tento problém docela snadno převedeme na hledání maximálního toku. Přidáme si uměle navíc ke grafu zdroj, který propojíme hranami jdoucími do všech vrcholů v jedné skupině v bipartitním grafu, zatímco ze všech vrcholů ve druhé skupině vedeme hranu do přidaného stoku. Všechny hrany opatříme maximální kapacitou 1 a hledáme maximální tok. Za páry pak bereme hrany s nenulovým (tj. zřejmě jednotkovým) tokem.

Maďarský algoritmus (König, Egerváry, Kuhn)

Označujme kvůli jednoduchosti popisu vrcholy jedné skupiny v bipartitním grafu jako **červené**, vrcholy druhé skupiny jako **modré**. Budeme předpokládat, že vrcholy jsou obarveny tak, že počet červených vrcholů nepřevyšuje počet modrých.

Maďarský algoritmus (König, Egerváry, Kuhn)

Označujme kvůli jednoduchosti popisu vrcholy jedné skupiny v bipartitním grafu jako **červené**, vrcholy druhé skupiny jako **modré**. Budeme předpokládat, že vrcholy jsou obarveny tak, že počet červených vrcholů nepřevyšuje počet modrých.

Definice

Nechť je dán bipartitní graf $G = (V, E)$ a párování $M \subseteq E$. *M-alternující cestou v G* nazveme takovou cestu v G, jejíž hrany tvoří střídavě hrany z M a z $E \setminus M$.

Maďarský algoritmus (König, Egerváry, Kuhn)

Označujme kvůli jednoduchosti popisu vrcholy jedné skupiny v bipartitním grafu jako **červené**, vrcholy druhé skupiny jako **modré**. Budeme předpokládat, že vrcholy jsou obarveny tak, že počet červených vrcholů nepřevyšuje počet modrých.

Definice

Nechť je dán bipartitní graf $G = (V, E)$ a párování $M \subseteq E$. *M-alternující cestou v G* nazveme takovou cestu v G, jejíž hrany tvoří střídavě hrany z M a z $E \setminus M$. *M-rozšiřující cestou* v G nazveme M-alternující cestu, která spojuje dosud nepřiřazený červený vrchol u s dosud nepřiřazeným modrým vrcholem v.

Algoritmus pro nalezení maximálního párování

1. Nalezneme libovolné párování M. Označíme všechny červené vrcholy jako přípustné.
2. Vyberme některý dosud nepřiřazený přípustný červený vrchol v (pokud neexistuje, jsme hotovi) a nalezněme pro něj (např. prostřednictvím prohledávání do hloubky) M-alternující strom s kořenem ve v (pokud neexistuje, označme v jako nepřípustný a proces opakujme s jiným vrcholem). Pokud strom obsahuje nějakou M-rozšiřující cestu, odstraníme M-hrany v této cestě z M a ostatní hrany této cesty do M přidáme. Označmě všechny červené vrcholy jako přípustné a proces opakujme.

Datové struktury

Obvykle (Kořenové pěstěné) binární stromy nesoucí informaci v uzlech.

- AVL stromy (vyvážené), podobně *red-black* stromy

Datové struktury

Obvykle (Kořenové pěstěné) binární stromy nesoucí informaci v uzlech.

- AVL stromy (vyvážené), podobně *red-black* stromy
- B-stromy (2-3 stromy, B* stromy)

Datové struktury

Obvykle (Kořenové pěstěné) binární stromy nesoucí informaci v uzlech.

- AVL stromy (vyvážené), podobně *red-black* stromy
- B-stromy (2-3 stromy, B* stromy)
- binární halda, Fibonacciho halda

Datové struktury

Obvykle (Kořenové pěstěné) binární stromy nesoucí informaci v uzlech.

- AVL stromy (vyvážené), podobně *red-black* stromy
- B-stromy (2-3 stromy, B* stromy)
- binární halda, Fibonacciho halda
- a mnoho dalších

Datové struktury

Obvykle (Kořenové pěstěné) binární stromy nesoucí informaci v uzlech.

- AVL stromy (vyvážené), podobně *red-black* stromy
- B-stromy (2-3 stromy, B^* stromy)
- binární halda, Fibonacciho halda
- a mnoho dalších

Datové struktury

Obvykle (Kořenové pěstěné) binární stromy nesoucí informaci v uzlech.

- AVL stromy (vyvážené), podobně *red-black* stromy
- B-stromy (2-3 stromy, B^* stromy)
- binární halda, Fibonacciho halda
- a mnoho dalších

V této oblasti odkážeme na předmět *Návrh algoritmů* a další, my si pouze ukážeme využití binárních stromů v kódování (Huffmanův algoritmus).

Pracujeme s pěstěnými binárními stromy, kde máme navíc každou hranu *obarvenou* některým symbolem z dané výstupní abecedy A (často $A = \{0, 1\}$). Kódovými slovy C jsou slova nad abecedou A, na která převádíme symboly vstupní abecedy. Našim úkolem je reprezentovat daný text pomocí vhodných kódových slov nad výstupní abecedou.

Je snadno vidět, že je užitečné chtít, aby seznam kódových slov byl *bezprefixový* (v opačném případě může nastat problém s dekódováním).

Pracujeme s pěstěnými binárními stromy, kde máme navíc každou hranu *obarvenou* některým symbolem z dané výstupní abecedy A (často $A = \{0, 1\}$). Kódovými slovy C jsou slova nad abecedou A, na která převádíme symboly vstupní abecedy. Našim úkolem je reprezentovat daný text pomocí vhodných kódových slov nad výstupní abecedou.

Je snadno vidět, že je užitečné chtít, aby seznam kódových slov byl *bezprefixový* (v opačném případě může nastat problém s dekódováním).

Příklad

Text: `MADAM, I'M ADAM`

$C = \{0, 1, 00, 01, 10, 11, 000\}$ Pokud symboly přiřazujeme tak, jak přicházejí na řadu, dostaneme

$M = 0, A = 1, D = 00, , = 01, I = 10, ' = 11, . = 000$. Přitom ale např. není jasné dekódování řetězce 01001 – je to MADA, MI, nebo ,DA?

Prefixový kód C splňuje, že žádný prvek C není prefixem jiného slova z C.

Ke konstrukci binárních prefixových kódů (tj. nad abecedou $A = \{0, 1\}$) využijeme binárních stromů. Označíme-li hrany vycházející z každého uzlu 0, resp. 1, a označíme-li navíc listy stromu symboly vstupní abecedy, dostaneme prefixový kód nad A pro tyto symboly zřetězením označení hran na cestě z kořene do příslušného listu.

Takto vytvořený kód je zřejmě *prefixový*.

Prefixový kód C splňuje, že žádný prvek C není prefixem jiného slova z C.

Ke konstrukci binárních prefixových kódů (tj. nad abecedou $A = \{0, 1\}$) využijeme binárních stromů. Označíme-li hrany vycházející z každého uzlu 0, resp. 1, a označíme-li navíc listy stromu symboly vstupní abecedy, dostaneme prefixový kód nad A pro tyto symboly zřetězením označení hran na cestě z kořene do příslušného listu.

Takto vytvořený kód je zřejmě *prefixový*.

Uděláme-li tuto konstrukci navíc tak, abychom odrazili četnost symbolů vstupní abecedy v kódovaném textu, dosáhneme tak dokonce *bezztrátové komprese dat*.

Huffmanův algoritmus

Nechť M je seznam četností symbolů vstupní abecedy v textu. Algoritmus postupně zkonstruuje optimální binární strom (tzv. *minimum-weight binary tree*) a přiřazení symbolů listům.

- Vyber dvě nejmenší četnosti w_1, w_2 z M. Vyrob strom se dvěma listy označenými příslušnými symboly a kořenem označeným $w_1 + w_2$, odeber z M hodnoty w_1, w_2 a nahraď je hodnotou $w_1 + w_2$.

Huffmanův algoritmus

Nechť M je seznam četností symbolů vstupní abecedy v textu. Algoritmus postupně zkonstruuje optimální binární strom (tzv. *minimum-weight binary tree*) a přiřazení symbolů listům.

- Vyber dvě nejmenší četnosti w_1, w_2 z M. Vyrob strom se dvěma listy označenými příslušnými symboly a kořenem označeným $w_1 + w_2$, odeber z M hodnoty w_1, w_2 a nahraď je hodnotou $w_1 + w_2$.
- Tento krok opakuj; pouze v případě, že vybraná hodnota z M je součtem, pak nevyráběj nový list, ale "připoj" příslušný již existující podstrom.

Plán přednášky

1. Toky v sítích
2. Problém maximálního toku v síti
3. Další aplikace
   - Bipartitní párování
   - Stromové datové struktury a prefixové kódy
4. Vytvořující funkce
5. Operace s vytvořujícími funkcemi
   - Přehled mocninných řad

Motto: *spojité a diskrétní modely se vzájemně potřebují a doplňují.*

Motto: *spojité a diskrétní modely se vzájemně potřebují a doplňují.*

Příklad

Máme v peněžence 4 korunové mince, 5 dvoukorunových a 3 pětikorunové. Z automatu, který nevrací, chceme minerálku za 22 Kč. Kolika způsoby to umíme, aniž bychom ztratili přeplatek?

Motto: *spojité a diskrétní modely se vzájemně potřebují a doplňují.*

Příklad

Máme v peněžence 4 korunové mince, 5 dvoukorunových a 3 pětikorunové. Z automatu, který nevrací, chceme minerálku za 22 Kč. Kolika způsoby to umíme, aniž bychom ztratili přeplatek?

Hledáme zjevně čísla i, j a k taková, že $i + j + k = 22$ a zároveň

$$i \in \{0, 1, 2, 3, 4\},\ j \in \{0, 2, 4, 6, 8, 10\},\ k \in \{0, 5, 10, 15\}.$$

Uvažme součin polynomů (třeba nad reálnými čísly)

$$(x^0+x^1+x^2+x^3+x^4)(x^0+x^2+x^4+x^6+x^8+x^{10})(x^0+x^5+x^{10}+x^{15}).$$

Mělo by být zřejmé, že hledaný počet řešení je díky (Cauchyovskému) způsobu násobení polynomů právě koeficient u x^{22} ve výsledném polynomu.

Motto: *spojité a diskrétní modely se vzájemně potřebují a doplňují.*

Příklad

Máme v peněžence 4 korunové mince, 5 dvoukorunových a 3 pětikorunové. Z automatu, který nevrací, chceme minerálku za 22 Kč. Kolika způsoby to umíme, aniž bychom ztratili přeplatek?

Hledáme zjevně čísla i, j a k taková, že $i + j + k = 22$ a zároveň

$$i \in \{0, 1, 2, 3, 4\},\ j \in \{0, 2, 4, 6, 8, 10\},\ k \in \{0, 5, 10, 15\}.$$

Uvažme součin polynomů (třeba nad reálnými čísly)

$$(x^0+x^1+x^2+x^3+x^4)(x^0+x^2+x^4+x^6+x^8+x^{10})(x^0+x^5+x^{10}+x^{15}).$$

Mělo by být zřejmé, že hledaný počet řešení je díky (Cauchyovskému) způsobu násobení polynomů právě koeficient u x^{22} ve výsledném polynomu. Skutečně tak dostáváme **čtyři možnosti** $3*5+3*2+1*1$, $3*5+2*2+3*1$, $2*5+5*2+2*1$ a $2*5+4*2+4*1$.

Předchozí příklad asi vypadal spíš jako složitý zápis jednoduchých "backtrackingových úvah". Následující příklad ukazuje, že tento postup lze ale s výhodou zobecnit.

Předchozí příklad asi vypadal spíš jako složitý zápis jednoduchých "backtrackingových úvah". Následující příklad ukazuje, že tento postup lze ale s výhodou zobecnit.

Nechť I, J jsou konečné množiny nezáporných celých čísel. Potom je pro dané $r \in \mathbb{N}$ počet řešení (i, j) rovnice $i + j = r$ splňujících $i \in I, j \in J$ roven koeficientu u x^r v polynomu $(\sum_{i \in I} x^i)(\sum_{j \in J} x^j)$.

Předchozí příklad asi vypadal spíš jako složitý zápis jednoduchých "backtrackingových úvah". Následující příklad ukazuje, že tento postup lze ale s výhodou zobecnit.
Nechť I, J jsou konečné množiny nezáporných celých čísel. Potom je pro dané $r \in \mathbb{N}$ počet řešení (i, j) rovnice $i + j = r$ splňujících $i \in I, j \in J$ roven koeficientu u x^r v polynomu $(\sum_{i \in I} x^i)(\sum_{j \in J} x^j)$.

Příklad

Kolika způsoby můžeme pomocí mincí (1, 2, 5, 10, 20 a 50 Kč) zaplatit platbu 100 Kč?

Předchozí příklad asi vypadal spíš jako složitý zápis jednoduchých "backtrackingových úvah". Následující příklad ukazuje, že tento postup lze ale s výhodou zobecnit.

Nechť I, J jsou konečné množiny nezáporných celých čísel. Potom je pro dané $r \in \mathbb{N}$ počet řešení (i, j) rovnice $i + j = r$ splňujících $i \in I, j \in J$ roven koeficientu u x^r v polynomu $(\sum_{i \in I} x^i)(\sum_{j \in J} x^j)$.

Příklad

Kolika způsoby můžeme pomocí mincí (1, 2, 5, 10, 20 a 50 Kč) zaplatit platbu 100 Kč?

Hledáme přirozená čísla $a_1, a_2, a_5, a_{10}, a_{20}$ a a_{50} taková, že a_i je násobkem i pro všechna $i \in \{1, 2, 5, 10, 20, 50\}$ a zároveň $a_1 + a_2 + a_5 + a_{10} + a_{20} + a_{50} = 100$.

Předchozí příklad asi vypadal spíš jako složitý zápis jednoduchých "backtrackingových úvah". Následující příklad ukazuje, že tento postup lze ale s výhodou zobecnit.
Nechť I, J jsou konečné množiny nezáporných celých čísel. Potom je pro dané $r \in \mathbb{N}$ počet řešení (i, j) rovnice $i + j = r$ splňujících $i \in I, j \in J$ roven koeficientu u x^r v polynomu $(\sum_{i \in I} x^i)(\sum_{j \in J} x^j)$.

Příklad

Kolika způsoby můžeme pomocí mincí (1, 2, 5, 10, 20 a 50 Kč) zaplatit platbu 100 Kč?

Hledáme přirozená čísla $a_1, a_2, a_5, a_{10}, a_{20}$ a a_{50} taková, že a_i je násobkem i pro všechna $i \in \{1, 2, 5, 10, 20, 50\}$ a zároveň $a_1 + a_2 + a_5 + a_{10} + a_{20} + a_{50} = 100$. Podobně jako výše je vidět, že požadovaný počet lze získat jako koeficient u x^{100} v

$$(1 + x + x^2 + \dots)(1 + x^2 + x^4 + \dots)(1 + x^5 + x^{10} + \dots)$$
$$(1 + x^{10} + x^{20} + \dots)(1 + x^{20} + x^{40} + \dots)(1 + x^{50} + x^{100} + \dots)$$

Podobným způsobem můžeme znovu velmi snadno odvodit některé kombinatorické vztahy, které známe již z dřívějška. Využijeme přitom **binomickou větu.**

Podobným způsobem můžeme znovu velmi snadno odvodit některé kombinatorické vztahy, které známe již z dřívějška. Využijeme přitom **binomickou větu.**

Věta (binomická)

Pro $n \in \mathbb{N}$ *a* $r \in \mathbb{R}$ *platí*

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n.$$

Na levou stranu se můžeme dívat jako na součin n polynomů, pravá je zápisem polynomu vzniklého jejich roznásobením.

Podobným způsobem můžeme znovu velmi snadno odvodit některé kombinatorické vztahy, které známe již z dřívějška. Využijeme přitom **binomickou větu.**

Věta (binomická)

Pro $n \in \mathbb{N}$ a $r \in \mathbb{R}$ platí

$$(1+x)^n = \binom{n}{0} + \binom{n}{1}x + \binom{n}{2}x^2 + \cdots + \binom{n}{n}x^n.$$

Na levou stranu se můžeme dívat jako na součin n polynomů, pravá je zápisem polynomu vzniklého jejich roznásobením. Dosazením čísel $x = 1$, resp. $x = -1$ dostáváme známé vzorce:

Důsledek

- $\sum_{k=0}^{n} \binom{n}{k} = 2^n$,
- $\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0$.

Podíváme se teď na obě strany v binomické větě "spojitýma očima" a s využitím vlastností derivací odvodíme další vztah mezi kombinačními čísly.

Důsledek

Platí

$$\sum_{k=0}^{n} k\binom{n}{k} = n2^{n-1}.$$

Podíváme se teď na obě strany v binomické větě "spojitýma očima" a s využitím vlastností derivací odvodíme další vztah mezi kombinačními čísly.

Důsledek

Platí

$$\sum_{k=0}^{n} k\binom{n}{k} = n2^{n-1}.$$

Důkaz.

Na obě strany binomické věty se podíváme jako na polynomiální funkce. Derivací levé strany dostaneme $n(1+x)^{n-1}$, derivací pravé strany (člen po členu) pak $\sum_{k=1}^{n} k\binom{n}{k}x^{k-1}$. Dosazením $x = 1$ dostaneme tvrzení. $\square$

(Formální) mocninné řady

Definice

Buď dána nekonečná posloupnost $a = (a_0, a_1, a_2, \ldots)$. Její **vytvořující funkcí** rozumíme (formální) mocninnou řadu tvaru

$$\sum_{i=0}^{\infty} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots .$$

(Formální) mocninné řady

Definice

Buď dána nekonečná posloupnost $a = (a_0, a_1, a_2, \ldots)$. Její **vytvořující funkcí** rozumíme (formální) mocninnou řadu tvaru

$$\sum_{i=0}^{\infty} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots .$$

Poznámka

O **formální** mocninné řadě hovoříme proto, že se zatím na tuto řadu díváme čistě formálně jako na jiný zápis dané posloupnosti a nezajímáme se o konvergenci. Na druhou stranu to ale znamená, že formální mocninná řada není funkce a nemůžeme do ní dosazovat.

(Formální) mocninné řady

Definice

Buď dána nekonečná posloupnost $a = (a_0, a_1, a_2, \dots)$. Její **vytvořující funkcí** rozumíme (formální) mocninnou řadu tvaru

$$\sum_{i=0}^{\infty} a_i x^i = a_0 + a_1 x + a_2 x^2 + \cdots .$$

Poznámka

O **formální** mocninné řadě hovoříme proto, že se zatím na tuto řadu díváme čistě formálně jako na jiný zápis dané posloupnosti a nezajímáme se o konvergenci. Na druhou stranu to ale znamená, že formální mocninná řada není funkce a nemůžeme do ní dosazovat. To ovšem vzápětí napravíme, když s využitím znalostí z analýzy nekonečných řad přejdeme od formálních mocninnných řad k příslušným funkcím.

Příklad

Posloupnosti samých jedniček odpovídá formální mocninná řada $1 + x + x^2 + x^3 + \cdots$. Z analýzy víme, že stejně zapsaná mocninná řada konverguje pro $x \in (-1, 1)$ a její součet je roven funkci $1/(1 - x)$. Stejně tak obráceně, rozvineme-li tuto funkci do Taylorovy řady v bodě 0, dostaneme zřejmě původní řadu. Takovéto "zakódování" posloupnosti čísel do funkce a zpět je klíčovým obratem v teorii vytvořujících funkcí.

Příklad

Posloupnosti samých jedniček odpovídá formální mocninná řada $1 + x + x^2 + x^3 + \cdots$. Z analýzy víme, že stejně zapsaná mocninná řada konverguje pro $x \in (-1, 1)$ a její součet je roven funkci $1/(1 - x)$. Stejně tak obráceně, rozvineme-li tuto funkci do Taylorovy řady v bodě 0, dostaneme zřejmě původní řadu. Takovéto "zakódování" posloupnosti čísel do funkce a zpět je klíčovým obratem v teorii vytvořujících funkcí.

Jak jsme již zmínili, tento obrat lze ale použít pouze tehdy, pokud víme, že řada alespoň v nějakém okolí 0 konverguje. Často ale "diskrétní" matematici používají následující "podvod":

- pomocí formálních mocninných řad odvodí nějaký vztah (formuli, rekurenci,...) bez toho, aby se zajímali o konvergenci

Příklad

Posloupnosti samých jedniček odpovídá formální mocninná řada $1 + x + x^2 + x^3 + \cdots$. Z analýzy víme, že stejně zapsaná mocninná řada konverguje pro $x \in (-1, 1)$ a její součet je roven funkci $1/(1 - x)$. Stejně tak obráceně, rozvineme-li tuto funkci do Taylorovy řady v bodě 0, dostaneme zřejmě původní řadu. Takovéto "zakódování" posloupnosti čísel do funkce a zpět je klíčovým obratem v teorii vytvořujících funkcí.

Jak jsme již zmínili, tento obrat lze ale použít pouze tehdy, pokud víme, že řada alespoň v nějakém okolí 0 konverguje. Často ale "diskrétní" matematici používají následující "podvod":

- pomocí formálních mocninných řad odvodí nějaký vztah (formuli, rekurenci,...) bez toho, aby se zajímali o konvergenci
- jinými prostředky (často matematickou indukcí) tento vztah dokážou

Vytvořující funkce v praxi využíváme:

- k nalezení **explicitní formule** pro n-tý člen posloupnosti;
- často vytvořující funkce vycházejí z rekurentních vztahů, občas ale díky nim odvodíme rekurentní vztahy nové;

Vytvořující funkce v praxi využíváme:

- k nalezení **explicitní formule** pro *n*-tý člen posloupnosti;
- často vytvořující funkce vycházejí z rekurentních vztahů, občas ale díky nim odvodíme rekurentní vztahy nové;
- výpočet průměrů či jiných statistických závislostí (např. průměrná složitost algoritmu);

Vytvořující funkce v praxi využíváme:

- k nalezení **explicitní formule** pro *n*-tý člen posloupnosti;
- často vytvořující funkce vycházejí z rekurentních vztahů, občas ale díky nim odvodíme rekurentní vztahy nové;
- výpočet průměrů či jiných statistických závislostí (např. průměrná složitost algoritmu);
- důkaz různých identit;
- často je nalezení přesného vztahu příliš obtížné, ale mnohdy stačí vztah přibližný nebo asymptotické chování.

Exponenciální vytvořující funkce

Kromě výše zmíněných vytvořujících funkcí se v praxi rovněž často objevují jejich tzv. *exponenciální* varianty.

$$g(x) = \sum_{n \geq 0} g_n \frac{x^n}{n!}.$$

Exponenciální vytvořující funkce

Kromě výše zmíněných vytvořujících funkcí se v praxi rovněž často objevují jejich tzv. *exponenciální* varianty.

$$g(x) = \sum_{n\geq 0} g_n \frac{x^n}{n!}.$$

Poznámka

Jméno vychází z toho, že exponenciální funkce e^x je (exponenciální) vytvořujícífunkcí pro základní posloupnost $(1, 1, 1, 1, \dots)$.
Později ukážeme některé příklady (např. Cayleyho větu), kdy je použití exponenciálních vytvořujících funkcí výhodnější.

Dosazování do mocninných řad

Následující větu znáte z matematické analýzy z loňského semestru:

Věta

Buď $(a_0, a_1, a_2, \dots)$ *posloupnost reálných čísel. Platí-li pro nějaké* $K \in \mathbb{R}$, *že pro všechna* $n \geq 1$ *je* $|a_n| \leq K^n$, *pak řada*

$$a(x) = \sum_{n \geq 0} a_n x^n$$

konverguje pro každé $x \in (-\frac{1}{K}, \frac{1}{K})$. *Součet této řady tedy definuje funkci na uvedeném intervalu, tuto funkci označujeme rovněž* $a(x)$.

Dosazování do mocninných řad

Následující větu znáte z matematické analýzy z loňského semestru:

Věta

Buď $(a_0, a_1, a_2, \dots)$ posloupnost reálných čísel. Platí-li pro nějaké $K \in \mathbb{R}$, že pro všechna $n \geq 1$ je $|a_n| \leq K^n$, pak řada

$$a(x) = \sum_{n \geq 0} a_n x^n$$

konverguje pro každé $x \in (-\frac{1}{K}, \frac{1}{K})$. Součet této řady tedy definuje funkci na uvedeném intervalu, tuto funkci označujeme rovněž $a(x)$. Hodnotami funkce $a(x)$ na libovolném okolí 0 je jednoznačně určena původní posloupnost, neboť má $a(x)$ v 0 derivace všech řádů a platí

$$a_n = \frac{a^{(n)}(0)}{n!}.$$

Plán přednášky

1. Toky v sítích
2. Problém maximálního toku v síti
3. Další aplikace
   - Bipartitní párování
   - Stromové datové struktury a prefixové kódy
4. Vytvořující funkce
5. Operace s vytvořujícími funkcemi
   - Přehled mocninných řad

Některým jednoduchým operacím s posloupnostmi odpovídají jednoduché operace nad mocninnými řadami:

Některým jednoduchým operacím s posloupnostmi odpovídají jednoduché operace nad mocninnými řadami:

- Sčítání $(a_i + b_i)$ posloupností člen po členu odpovídá součet $a(x) + b(x)$ příslušných vytvořujících funkcí.

Některým jednoduchým operacím s posloupnostmi odpovídají jednoduché operace nad mocninnými řadami:

- Sčítání $(a_i + b_i)$ posloupností člen po členu odpovídá součet $a(x) + b(x)$ příslušných vytvořujících funkcí.
- Vynásobení $(\alpha \cdot a_i)$ všech členů posloupnosti stejným skalárem α odpovídá vynásobení $\alpha \cdot a(x)$ příslušné vytvořující funkce.

Některým jednoduchým operacím s posloupnostmi odpovídají jednoduché operace nad mocninnými řadami:

- Sčítání $(a_i + b_i)$ posloupností člen po členu odpovídá součet $a(x) + b(x)$ příslušných vytvořujících funkcí.
- Vynásobení $(\alpha \cdot a_i)$ všech členů posloupnosti stejným skalárem α odpovídá vynásobení $\alpha \cdot a(x)$ příslušné vytvořující funkce.
- Vynásobení vytvořující funkce $a(x)$ monomem x^k odpovídá posunutí posloupnosti doprava o k míst a její doplnění nulami.

Některým jednoduchým operacím s posloupnostmi odpovídají jednoduché operace nad mocninnými řadami:

- Sčítání $(a_i + b_i)$ posloupností člen po členu odpovídá součet $a(x) + b(x)$ příslušných vytvořujících funkcí.
- Vynásobení $(\alpha \cdot a_i)$ všech členů posloupnosti stejným skalárem α odpovídá vynásobení $\alpha \cdot a(x)$ příslušné vytvořující funkce.
- Vynásobení vytvořující funkce $a(x)$ monomem x^k odpovídá posunutí posloupnosti doprava o k míst a její doplnění nulami.
- Pro posunutí posloupnosti doleva o k míst (tj. vynechání prvních k míst posloupnosti) nejprve od $a(x)$ odečteme polynom $b_k(x)$ odpovídají posloupnosti $(a_0, \ldots, a_{k-1}, 0, \ldots)$ a poté podělíme vytvořující funkci x^k.

Některým jednoduchým operacím s posloupnostmi odpovídají jednoduché operace nad mocninnými řadami:

- Sčítání $(a_i + b_i)$ posloupností člen po členu odpovídá součet $a(x) + b(x)$ příslušných vytvořujících funkcí.
- Vynásobení $(\alpha \cdot a_i)$ všech členů posloupnosti stejným skalárem α odpovídá vynásobení $\alpha \cdot a(x)$ příslušné vytvořující funkce.
- Vynásobení vytvořující funkce $a(x)$ monomem x^k odpovídá posunutí posloupnosti doprava o k míst a její doplnění nulami.
- Pro posunutí posloupnosti doleva o k míst (tj. vynechání prvních k míst posloupnosti) nejprve od $a(x)$ odečteme polynom $b_k(x)$ odpovídají posloupnosti $(a_0, \dots, a_{k-1}, 0, \dots)$ a poté podělíme vytvořující funkci x^k.
- Substitucí polynomu $f(x)$ s nulovým absolutním členem za x vytvoříme specifické kombinace členů původní posloupnosti. Jednoduše je vyjádříme pro $f(x) = \alpha x$, což odpovídá vynásobení k–tého členu posloupnosti skalárem α^k. Dosazení $f(x) = x^n$ nám do posloupnosti mezi každé dva členy vloží $n - 1$ nul.

Příklad

Viděli jsme, že $1/(1-x)$ je vytvořující funkcí posloupnosti ze samých jedniček. Substitucí $2x$ za x tak dostaneme tvrzení, že $1/(1-2x)$ je vytvořující funkcí posloupnosti $(1, 2, 4, 8, \dots)$.

Příklad

Viděli jsme, že $1/(1-x)$ je vytvořující funkcí posloupnosti ze samých jedniček. Substitucí $2x$ za x tak dostaneme tvrzení, že $1/(1-2x)$ je vytvořující funkcí posloupnosti $(1, 2, 4, 8, \dots)$.
S využitím substituce $-x$ za x dostaneme, že je-li $a(x)$ vytvořující pro $(a_0, a_1, \dots)$, je $(a(x) + a(-x))/2$ vytvořující pro $a_0, 0, a_2, 0, \dots)$.

Příklad

Viděli jsme, že $1/(1-x)$ je vytvořující funkcí posloupnosti ze samých jedniček. Substitucí $2x$ za x tak dostaneme tvrzení, že $1/(1-2x)$ je vytvořující funkcí posloupnosti $(1, 2, 4, 8, \dots)$.
S využitím substituce $-x$ za x dostaneme, že je-li $a(x)$ vytvořující pro $(a_0, a_1, \dots)$, je $(a(x) + a(-x))/2$ vytvořující pro $a_0, 0, a_2, 0, \dots)$.

Příklad

Určete vytvořující funkci posloupnosti

$$(1, 1, 2, 2, 4, 4, 8, 8, \dots).$$

Příklad

Viděli jsme, že $1/(1-x)$ je vytvořující funkcí posloupnosti ze samých jedniček. Substitucí $2x$ za x tak dostaneme tvrzení, že $1/(1-2x)$ je vytvořující funkcí posloupnosti $(1, 2, 4, 8, \dots)$.
S využitím substituce $-x$ za x dostaneme, že je-li $a(x)$ vytvořující pro $(a_0, a_1, \dots)$, je $(a(x) + a(-x))/2$ vytvořující pro $a_0, 0, a_2, 0, \dots)$.

Příklad

Určete vytvořující funkci posloupnosti

$$(1, 1, 2, 2, 4, 4, 8, 8, \dots).$$

Řešení

Podle předchozího příkladu je $1/(1-2x)$ vytvořující funkcí posloupnosti $(1, 2, 4, 8, \dots)$. Substitucí x^2 za x dostaneme vytvořující funkci $1/(1-2x^2)$ pro $(1, 0, 2, 0, 4, 0, \dots)$ a celkem pak je $(1+x)/(1-2x^2)$ hledanou vytvořující funkcí.

Dalšími důležitými operacemi, které se při práci s vytvořujícími funkcemi často objevují, jsou:

- Derivování podle x: funkce $a'(x)$ vytvořuje posloupnost $(a_1, 2a_2, 3a_3, \dots)$, člen s indexem k je $(k+1)a_{k+1}$ (tj. mocninnou řadu derivujeme člen po členu).

Dalšími důležitými operacemi, které se při práci s vytvořujícími funkcemi často objevují, jsou:

- Derivování podle x: funkce $a'(x)$ vytvořuje posloupnost $(a_1, 2a_2, 3a_3, \dots)$, člen s indexem k je $(k+1)a_{k+1}$ (tj. mocninnou řadu derivujeme člen po členu).
- Integrování: funkce $\int_0^x a(t)\,\mathrm{d}t$ vytvořuje posloupnost $(0, a_0, \frac{1}{2}a_1, \frac{1}{3}a_2, \frac{1}{4}a_3, \dots)$, pro $k \geq 1$ je člen s indexem k je $\frac{1}{k}a_{k-1}$ (zřejmě je derivací příslušné mocninné řady člen po členu původní funkce $a(x)$).

Dalšími důležitými operacemi, které se při práci s vytvořujícími funkcemi často objevují, jsou:

- Derivování podle x: funkce $a'(x)$ vytvořuje posloupnost $(a_1, 2a_2, 3a_3, \ldots)$, člen s indexem k je $(k+1)a_{k+1}$ (tj. mocninnou řadu derivujeme člen po členu).
- Integrování: funkce $\int_0^x a(t)\,\mathrm{d}t$ vytvořuje posloupnost $(0, a_0, \frac{1}{2}a_1, \frac{1}{3}a_2, \frac{1}{4}a_3, \ldots)$, pro $k \geq 1$ je člen s indexem k je $\frac{1}{k}a_{k-1}$ (zřejmě je derivací příslušné mocninné řady člen po členu původní funkce $a(x)$).
- Násobení řad: součin $a(x)b(x)$ je vytvořující funkcí posloupnosti $(c_0, c_1, c_2, \ldots)$, kde

$$c_k = \sum_{i+j=k} a_i b_j.$$

  (tj. členy v součinu až po c_k jsou stejné jako v součinu $(a_0 + a_1x + a_2x^2 + \cdots + a_kx^k)(b_0 + b_1x + b_2x^2 + \cdots b_kx^k)$. Posloupnost c bývá také nazývána *konvolucí* posloupností a, b.

Příklad

Jaká je vytvořující funkce pro posloupnost druhých mocnin $(1^2, 2^2, 3^2, \dots)$?

Řešení

Lze očekávat, že bude snazší bude nejprve určit vytvořující funkce pro $(1, 2, 3, \dots)$. Podle předchozího víme, že $1/(1-x)$ vytvoří posloupnost samých jedniček a její derivace $1/(1-x)^2$ pak posloupnost $(1, 2, 3, \dots)$. Jak ale zjistit funkci odpovídající druhým mocninám?

Příklad

Jaká je vytvořující funkce pro posloupnost druhých mocnin $(1^2, 2^2, 3^2, \dots)$?

Řešení

Lze očekávat, že bude snazší bude nejprve určit vytvořující funkce pro $(1, 2, 3, \dots)$. Podle předchozího víme, že $1/(1-x)$ vytvoří posloupnost samých jedniček a její derivace $1/(1-x)^2$ pak posloupnost $(1, 2, 3, \dots)$. Jak ale zjistit funkci odpovídající druhým mocninám? Druhou derivací dostaneme $2/(1-x)^3$, k ní odpovídající posloupnost je $(1 \times 2, 2 \times 3, 3 \times 4, \dots)$, jejíž člen s indexem k je $(k+2)(k+1)$. Snadno vidíme, že výslednou vytvořující funkcí je tedy

$$a(x) = \frac{2}{(1-x)^3} - \frac{1}{(1-x)^2}.$$

Příklad

Uvažme jeden speciální případ násobení vytvořujících funkcí $a(x)$ a $b(x)$, je-li $a(x) = 1/(1-x)$. Pak konvolucí příslušných posloupností je posloupnost, jejíž vytvořující funkce je dána mocninnou řadou

$$\begin{aligned}(1 + x + x^2 + x^3 + \cdots)(b_0 + b_1x + b_2x^2 + \cdots) = \\ = b_0 + (b_0 + b_1)x + (b_0 + b_1 + b_2)x^2 + \cdots\end{aligned}$$

Vyjádřeno slovy, vynásobením funkce $b(x)$ funkcí $1/(1-x)$ dostaneme vytvořující funkci posloupnosti částečných součtů původní posloupnosti $(b_0, b_1, b_2, \dots)$.

Přehled mocninných řad:

$$\frac{1}{1-x} = \sum_{n\geq 0} x^n,$$

$$\ln \frac{1}{1-x} = \sum_{n\geq 1} \frac{x^n}{n},$$

$$e^x = \sum_{n\geq 0} \frac{x^n}{n!},$$

$$\sin x = \sum_{n\geq 0} (-1)^n \frac{x^{2n+1}}{(2n+1)!},$$

$$\cos x = \sum_{n\geq 0} (-1)^n \frac{x^{2n}}{(2n)!},$$

$$(1+x)^r = \sum_{k\geq 0} \binom{r}{k} x^k.$$

Poznámka

- Poslední vzorec

$$(1+x)^r = \sum_{k\geq 0} \binom{r}{k} x^k$$

je tzv. **zobecněná binomická věta**, kde pro $r \in \mathbb{R}$ je binomický koeficient definován vztahem

$$\binom{r}{k} = \frac{r(r-1)(r-2)\cdots(r-k+1)}{k!}.$$

Speciálně klademe $\binom{r}{0} = 1$.

Poznámka

- Poslední vzorec

$$(1+x)^r = \sum_{k\geq 0} \binom{r}{k} x^k$$

je tzv. **zobecněná binomická věta**, kde pro $r \in \mathbb{R}$ je binomický koeficient definován vztahem

$$\binom{r}{k} = \frac{r(r-1)(r-2)\cdots(r-k+1)}{k!}.$$

Speciálně klademe $\binom{r}{0} = 1$.

- Pro $n \in \mathbb{N}$ z uvedeného vztahu snadno dostaneme

$$\frac{1}{(1-x)^n} = \binom{n-1}{n-1} + \binom{n}{n-1} x + \cdots + \binom{n+k-1}{n-1} x^k + \cdots.$$

Fibonacciho čísla a zlatý řez

Připomeňme, že Fibonacciho čísla jsou dána rekurentním předpisem

$$F_0 = 0, F_1 = 1, F_{n+2} = F_{n+1} + F_n.$$

Již dříve jste si uváděli všemožné výskyty této posloupnosti v přírodě, v matematice nebo v teoretické informatice. Našim cílem bude (opět) najít formuli pro výpočet n-tého členu posloupnosti.

Fibonacciho čísla a zlatý řez

Připomeňme, že Fibonacciho čísla jsou dána rekurentním předpisem

$$F_0 = 0, F_1 = 1, F_{n+2} = F_{n+1} + F_n.$$

Již dříve jste si uváděli všemožné výskyty této posloupnosti v přírodě, v matematice nebo v teoretické informatice. Našim cílem bude (opět) najít formuli pro výpočet n-tého členu posloupnosti.

Poznámka

(Nejen) pro manipulace se sumami používají autoři *Concrete mathematics* velmi vhodné označení `[logický predikát]` – výraz je roven 1 v případě splnění predikátu, jinak 0.
Např. $[n = 1], [2|n]$ apod.

Fibonacciho čísla a zlatý řez

Připomeňme, že Fibonacciho čísla jsou dána rekurentním předpisem

$$F_0 = 0, F_1 = 1, F_{n+2} = F_{n+1} + F_n.$$

Již dříve jste si uváděli všemožné výskyty této posloupnosti v přírodě, v matematice nebo v teoretické informatice. Našim cílem bude (opět) najít formuli pro výpočet n-tého členu posloupnosti.

Poznámka

(Nejen) pro manipulace se sumami používají autoři *Concrete mathematics* velmi vhodné označení [*logický predikát*] – výraz je roven 1 v případě splnění predikátu, jinak 0.
Např. $[n = 1], [2|n]$ apod.
Pro vyjádření koeficientu u x^n ve vytvořující funkci $F(x)$ se pak často používá zápis $[x^n]F(x)$.

Příklad – pokr.

Uvažme vytvořující funkci $F(x)$ Fibonacciho posloupnosti. Pak zřejmě

$$F(x) - xF(x) - x^2F(x) = x,$$

a tedy

$$F(x) = \frac{x}{1 - x - x^2}.$$

Našim cílem je tedy odvodit vztah pro n-tý člen posloupnosti odpovídající vytvořující funkci $F(x) = \frac{x}{1-x-x^2}$.

Příklad – pokr.

Uvažme vytvořující funkci $F(x)$ Fibonacciho posloupnosti. Pak zřejmě

$$F(x) - xF(x) - x^2F(x) = x,$$

a tedy

$$F(x) = \frac{x}{1 - x - x^2}.$$

Našim cílem je tedy odvodit vztah pro n-tý člen posloupnosti odpovídající vytvořující funkci $F(x) = \frac{x}{1-x-x^2}$.
Využijeme k tomu rozklad na parciální zlomky a dostaneme

$$\frac{x}{1 - x - x^2} = \frac{A}{x - x_1} + \frac{B}{x - x_2},$$

kde x_1, x_2 jsou kořeny polynomu $1 - x - x^2$ a A, B vhodné konstanty odvozené z počátečních podmínek.

Příklad – pokr.

Uvažme vytvořující funkci $F(x)$ Fibonacciho posloupnosti. Pak zřejmě

$$F(x) - xF(x) - x^2F(x) = x,$$

a tedy

$$F(x) = \frac{x}{1 - x - x^2}.$$

Našim cílem je tedy odvodit vztah pro n-tý člen posloupnosti odpovídající vytvořující funkci $F(x) = \frac{x}{1-x-x^2}$.
Využijeme k tomu rozklad na parciální zlomky a dostaneme

$$\frac{x}{1 - x - x^2} = \frac{A}{x - x_1} + \frac{B}{x - x_2},$$

kde x_1, x_2 jsou kořeny polynomu $1 - x - x^2$ a A, B vhodné konstanty odvozené z počátečních podmínek. Po substituci $\lambda_1 = 1/x_1, \lambda_2 = 1/x^2$ dostáváme vztah

$$\frac{x}{1 - x - x^2} = \frac{a}{1 - \lambda_1 x} + \frac{b}{1 - \lambda_2 x},$$

Příklad – závěr

S využitím počátečních podmínek dostáváme

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right].$$

Jistě je zajímavé, že tento výraz plný iracionálních čísel je vždy celočíselný.

Příklad – závěr

S využitím počátečních podmínek dostáváme

$$F_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right].$$

Jistě je zajímavé, že tento výraz plný iracionálních čísel je vždy celočíselný.
Uvážíme-li navíc, že $(1-\sqrt{5})/2 \approx -0.618$, vidíme, že pro všechna přirozená čísla lze F_n snadno spočítat zaokrouhlením čísla $\frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n$.

Příklad – závěr

S využitím počátečních podmínek dostáváme

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right].$$

Jistě je zajímavé, že tento výraz plný iracionálních čísel je vždy celočíselný.
Uvážíme-li navíc, že $(1-\sqrt{5})/2 \approx -0.618$, vidíme, že pro všechna přirozená čísla lze F_n snadno spočítat zaokrouhlením čísla $\frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n$. Navíc je vidět, že $\lim_{n\to\infty} F_n/F_{n+1} = 1/\lambda_1 \approx 0.618$, což je poměr známý jako zlatý řez – objevuje se již od antiky v architektuře, výtvarném umění i hudbě.

Příklad – závěr

S využitím počátečních podmínek dostáváme

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right].$$

Jistě je zajímavé, že tento výraz plný iracionálních čísel je vždy celočíselný.
Uvážíme-li navíc, že $(1-\sqrt{5})/2 \approx -0.618$, vidíme, že pro všechna přirozená čísla lze F_n snadno spočítat zaokrouhlením čísla $\frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^n$. Navíc je vidět, že $\lim_{n\to\infty} F_n/F_{n+1} = 1/\lambda_1 \approx 0.618$, což je poměr známý jako zlatý řez – objevuje se již od antiky v architektuře, výtvarném umění i hudbě.
Analogický postup je možné použít při řešení obecných lineárních diferenčních rovnic k-tého stupně s konstatními koeficienty. Má-li charakteristická rovnice jednoduché kořeny, je situace jednodušší – viz dříve.