ADSO
11. Techniky návrhu efektivních algoritmů

(prezentace k učebnici)

Ivan Ryant

Agenda

- Základní pojmy
- Úloha abstraktních datových typů
- Posloupnosti a operace s nimi
- Vyhledávací datové struktury
- Vyhledávací posloupnost
- Algoritmy řazení
- Stromy
- Rozptýlené tabulky
- Prohledávání do hloubky a do šířky
- Práce s grafy
- **Techniky návrhu efektivních algoritmů**

Techniky návrhu efektivních algoritmů

- Odstranění opakovaných výpočtů
- Výpočet nové hodnoty z předchozí hodnoty
- Přímé generování hledaných údajů
- Předzpracování vstupních dat
- Náhrada rekurze cyklem
- Ořezávání
- Heuristika
- Rozděl a panuj

Odstranění opakovaných výpočtů

Jestliže se v algoritmu nějaký výpočet opakuje, bývá výhodnější uložit si výsledek výpočtu do proměnné a místo opakovaného výpočtu použít proměnnou. Odstranění opakovaných výpočtů je použito např. v těchto případech:

- zjištění délky seznamu
  - pamatuje se např. v datové položce `BaseLinkedList.size`
- výpočet váhy podstromu
  - datová položka `weight`
- Dijkstrův algoritmus
  - třída `GraphTasksVertex` s datovými položkami `status`, `pathLength` atd.
- Floydův algoritmus
  - pole `store` a položky jeho prvků: `value, isDirect, between`

Výpočet nové hodnoty z předchozí hodnoty

Jestliže algoritmus počítá posloupnost hodnot, bývá lepší počítat následující hodnotu z hodnoty předchozí než každou hodnotu samostatně. Např.:

- generování Fibonacciho posloupnosti
- Dijkstrův algoritmus
- Floydův algoritmus

Přímé generování hledaných údajů

Než bychom generovali mnoho možných řešení a pak vybírali nejlepší z nich, raději vygenerujeme nejlepší řešení přímo. Např.:

- hladový algoritmus minimální kostry grafu

Předzpracování vstupních dat

Jsou-li vstupní data nepříznivě uspořádaná nebo nevhodně reprezentovaná, komplikují a brzdí algoritmus. Pomůže nám, když změníme jejich

- reprezentaci
  - např. ze znakové na binární
- uspořádání
  - např. z posloupnosti položek na pole, které můžeme přímo indexovat klíčem položky

Např.:

- načtení posloupnosti hran do seznamu následníků v grafových úlohách
- hladový algoritmus minimální kostry grafu

Náhrada rekurze cyklem

Někdy to jde bez pomocných datových struktur, jindy si musíme pomoci zásobníkem. Touto náhradou se někdy podaří program zrychlit, většinou se zmenší nároky na paměť. Příklady použití:

- výpočet faktoriálu nebo Fibonacciho čísel
- hledání největšího společného dělitele
- prohledávání stavového prostoru do hloubky

Ořezávání

Při průchodu stavovým prostorem do hloubky někdy dokážeme rozhodnout, že některé větve vedoucí z daného vrcholu nemusíme prohlížet. Např.:

- procházení vyhledávacím stromem
  - neprocházíme celý strom, nýbrž jen větve vedoucí k cíli
- násobení nulou při vyhodnocení aritmetického výrazu
- implementace Robinsonovy procedury v jazyce Prolog

Heuristika

Dobrý nápad. I když nefunguje zaručeně vždycky, stačí, když zabere většinou. V ostatních případech bude výpočet pomalý nebo jinak náročný, nicméně správný. Např.:

- proskákání šachovnice koněm
- reorganizovaná posloupnost
- transformační funkce rozptýlené tabulky
- výběr pivota v quicksortu

Rozděl a panuj

Technika, která nám umožňuje zvládnout složitý problém. Složitý problém rozdělíme na několik jednodušších úloh a ty pak řešíme každou zvlášť. Např.:

- řazení rozdělováním (quicksort)
- řazení sléváním (mergesort)
- hanojské věže

Konec

Tato prezentace patří k učebnici *Algoritmy a datové struktury objektově* od Ivana Ryanta. Obsahuje texty a obrázky z této učebnice.

Tato prezentace smí být volně šířena, ale vždy i s tímto snímkem. Citujete-li, vyznačte zřetelně citát v plném rozsahu a uveďte zdroj:

RYANT, Ivan. *Algoritmy a datové struktury objektově.* Praha: Ivan Ryant, 2017. ISBN 978-80-270-1660-0