

Základní pojmy z algoritmické složitosti

Kristýna Zemková

Algoritmus

- = přesný návod či postup, kterým lze vyřešit daný typ úlohy
- nečastěji se objevuje při programování (teoretický princip řešení problému)
- obecně v jakémkoli odvětví
- příkladem může být i kuchařský recept

Způsob zápisu algoritmu

- lineárním vyjádřením
 - pomocí přirozeného jazyka doplněného klíčovými slovy (tj. metajazyk nebo pseudokód)
 - zápisem v některém z programovacích jazyků
- grafickým zápisem
 - v podobě vývojového diagramu
 - v podobě strukturogramu

Vlastnosti algoritmů

Konečnost (finitnost)

- každý algoritmus musí skončit v konečném počtu kroků

Obecnost (univerzálnost)

- neřeší jeden konkrétní problém (např. $3 \cdot 7$), ale obecnou třídu obdobných problémů (násobení přirozených čísel)

Determinovanost

- každý krok algoritmu musí být jednoznačně a přesně definován
- v každé situaci musí být naprosto zřejmé, *co* a *jak* se má provést
- pro stejné vstupy dostaneme pokaždé stejné výsledky

Výstup (resultativnost)

- algoritmus má alespoň jeden výstup
- algoritmus vede od zpracování hodnot k výstupu

Elementárnost

- algoritmus se skládá z konečného počtu jednoduchých (elementárních) kroků

Složitost

- jsme omezeni prostředky, které máme k dispozici (čas, paměť, počet registrů...)
- = vztah dané metody k daným prostředkům
- dělíme na:
 - časovou složitost
 - prostorovou (paměťovou) složitost

Časová složitost

- = funkce, která každé množině vstupních dat přiřazuje počet operací vykonaných při výpočtu podle daného algoritmu
- kolik času potřebuje program na své vykonání

Prostorová složitost

- = závislost paměťových nároků algoritmu na vstupních datech
 - kolik paměti (úložného prostoru) program ke svému běhu potřebuje
- Obě složitosti jsou do jisté míry komplementární

Asymptotická složitost

- snaha vyjádřit složitost jako funkci vstupu
- říct, jak roste složitost algoritmu vzhledem k rostoucímu vstupu
- složitost algoritmu udává, jak je daný algoritmus rychlý (kolik provede elementárních operací) vzhledem k množině vstupních dat
- můžeme uvažovat:
 - složitost v *nejlepší*m případě
 - složitost v *průměrném* případě
 - složitost v *nejhorším* případě
 - *amortizovanou* složitost – určuje časovou složitost algoritmu v sekvenci nejhorších možných vstupních dat – nevyužívá pravděpodobnosti, a je proto zaručená

Základní třídy složitosti

název	složitost	příklad
konstantní	$O(1) = c$	pole – n tý prvek
logaritmická	$O(\log n)$	binární vyhledávání
lineární	$O(n)$	hledání v neseřazených posloupnostech
lineárnělogaritmická	$O(n * \log n)$	
kvadratická	$O(n^2)$	
kubická	$O(n^3)$	
obecná polynomiální	$O(n^c)$	
exponenciální	$O(c^n)$	
faktoriálová	$O(n!)$	

Složitost algoritmu vs. složitost problému

Složitost algoritmu

- Složitostí algoritmu rozumíme složitost konkrétní instance zadaného algoritmu (implementovaného v nějakém programovacím jazyce). Algoritmy pracující s lepší než exponenciální/faktoriálovou složitostí označujeme jako *efektivní*.

Složitost problému

- Složitostí problému rozumíme složitost *optimálního* algoritmu konkrétně řešícího zadaný problém.