16. Algoritmizace: vlastnosti algoritmu, způsoby zápisu algoritmu, časová a paměťová složitost

POČÍTAČOVÉ SÍTĚ A PROGRAMOVÁNÍ

- Algoritmus je přesný návod či postup, kterým lze vyřešit daný typ úlohy.
- Zapisuje se tak teoretický princip řešení problému.

Vlastnosti algoritmů

Elementárnost

- Algoritmus se skládá z konečného počtu jednoduchých kroků.

Konečnost (finitnost)

- Každý algoritmus musí skončit v konečném počtu kroků. Tento počet může být libovolně velká, ale pro každý jednotlivý postup konečný.

Univerzálnost

- Algoritmus neřeší jeden konkrétní problém, ale obecnou třídu obdobných problémů.
- Př.: neřeší 3*7, ale jak spočítat součin dvou celých čísel.

Determinovanost

- Algoritmus je determinovaný, pokud za stejných podmínek poskytuje stejný výstup.
- Tato vlast je požadována u velké části úloh.
- Existují ale také algoritmy, kde je vyžadována náhodnost (míchání karet).
- Je obtížné dosáhnout náhodnost na počítači.

Determinismus

- Každá krok algoritmu musí být jednoznačně a přesně definován.
- Přirozené jazyky neposkytují takovou přesnost definice, a proto jsou lepší programovací, kde každý příkaz má jeden jasný význam.
- Některé algoritmy jsou determinované, ale ne deterministické (řadící algoritmus quick sort as náhodnou volbou pivota).

Výstup

- Algoritmus má alespoň jeden výstup, který je v požadovaném vztahu k zadaným vstupům, a tím tvoří odpověď na problém, který řeší.

Druhy algoritmů

Rekurzivní algoritmy

- Algoritmy, které volají samy sebe.
- Tento typ algoritmu je založen na rekurzi.
- V rekurzi je problém vyřešen jeho rozdělením na dílčí problémy stejného typu a opakovaným voláním vlastního já, dokud není problém vyřešen pomocí základní podmínky.
- Některé běžné problémy, které se řeší pomocí rekurzivních algoritmů, jsou Factorial of a Number, Fibonacciho posloupnost, Hanojská věž, DFS pro Graph atd.

Pravděpodobnostní algoritmy

- Provádějí některá rozhodnutí pseudonáhodně.

Genetické algoritmy

- Pracují na základě napodobování biologických evolučních procesů.
- Mutace a křížení.

Heuristické algoritmy

- Neklade si za cíl nalézt přesné řešení, ale pouze vhodné přiblížení.
- Používá se, kde zdroje nepostačují k využití exaktních algoritmů.

Brute Force algoritmus

- Toto je nejzákladnější a nejjednodušší typ algoritmu.
- Algoritmus brutální síly je přímý přístup k problému, tj. První přístup, který nás napadne při pohledu na problém.
- Techničtěji je to jako opakovat všechny dostupné možnosti řešení tohoto problému.

A další …

Paradigmata návrhů algoritmů

Rozděl a panuj

- Algoritmy typu rozděl a panuj dělí problém na menší podproblémy, na něž se rekurzivně aplikují, po čemž se dílčí řešení vhodným způsobem sloučí.
- Např. binární vyhledávání, quick sort.

Hladový algoritmus

- Přímočarý postup k řešení určité třídy optimalizačních úloh.
- Zpracovává se množina V složená z n údajů.
    - Úkolem je najít podmnožinu W množiny V, která vyhovuje určitým podmínkám a při tom optimalizuje předepsanou účelovou funkci.
- Jakákoliv množina vyhovující podmínkám se nazývá *Přípustné řešení.*
- Algoritmus bude procházet prvky a rozhodovat, zda vyhovuje nebo ne.
- Např.: cesta grafu

Dynamické programování

- Používá se v případech, kdy lze optimální řešení složit z řešení jednodušších.
- Protože se požadavky na řešení jednodušších podproblémů můžou mnohokrát opakovat, je nutné zvolit správné pořadí a výsledky si pamatovat pro opakované použití.
- Opírá se o princip optimality.
    - Optimální posloupnost rozhodnutí má tu vlastnost, že ať je počáteční stav a rozhodnutí jakékoliv, musí být všechna následující rozhodnutí optimální vzhledem k výsledkům rozhodnutí prvního.
- Např.: grafové úlohy

Známé algoritmy

- Eratosthenovo síto
    - Nalezení prvočísel.
- Euklidův algorimus
    - Největší společný dělitel.

- Dijkstrův algoritmus
    - Hledání nejkratší cesty.

Způsoby zápisu algoritmu

Přirozený jazyk

- Výhody
    - Jednoduchost
- Nevýhody
    - Nejednoznačnost, nesrozumitelnost pro cizince
- Využití
    - Jen pro obecnou analýzu a popis koncepce řešení.

Vyšší programovací jazyk

- Výhody
    - Snadné vytvoření programu
- Nevýhody
    - Nutná znalost konkrétního jazyka, nepřehlednost algoritmu

Grafická forma zápisu

- Zvýšení přehlednosti algoritmu.
    - Rozhodovací tabulka
    - Strukturogram
    - Vývojový diagram

Časová a paměťová složitost

Časová složitost

- Časová složitost, nebo také Asymptotická složitost je nástroj pro porovnání efektivity algoritmů.
- Zapisuje se pomocí Landauovy notace, známá jako big O notation.
    - O(f(N))
- Abychom zvolili nejlepší algoritmus, musí být časová složitost co nejlepší.
- **Rozdělení časové složitosti:**
    - O(N)
        - Lineární
        - S větším počtem vstupů se čas lineárně zvětšuje.
    - O(1)
        - Čas se s rostoucími vstupy neprodlužuje.
    - $O(N^2)$
        - Exponenciálně
    - O(log N)
        - Logaritmicky

Paměťová složitost

- Jedná se o množství paměti, kterou potřebujeme při provádění výpočtu.
- Může ji představovat např.:
    - Maximální počet bitů nutných pro uložení všech dat v každé jednotlivé konfiguraci.

    - Maximální počet paměťových buněk použitých během výpočtu.
- Mnohdy je paměťová složitost menší než časová.
    - Insertion sort má paměťovou složitost O(n), ale časovou má $O(N^2)$.
- **Orientační hodnoty pro běžná PC**
    - O(N)
        - až 100 000 000
    - O(N log N)
        - až 1 000 000
    - $O(N^2)$
        - až 10 000
    - $O(N^3)$
        - až 1000
    - $2^{O(N)}$
        - 20 – 30