Algoritmizace

Pojmy

- **Algoritmus** – konečná sekvence operací sloužící k řešení určitého typu úlohy
- **Program** – systematicky sestavený soubor algoritmů, zapsaný v určitém programovacím jazyce, který lze zkompilovat a spustit
- **Programovací jazyk** – prostředek pro zápis algoritmů a tvorbu programů

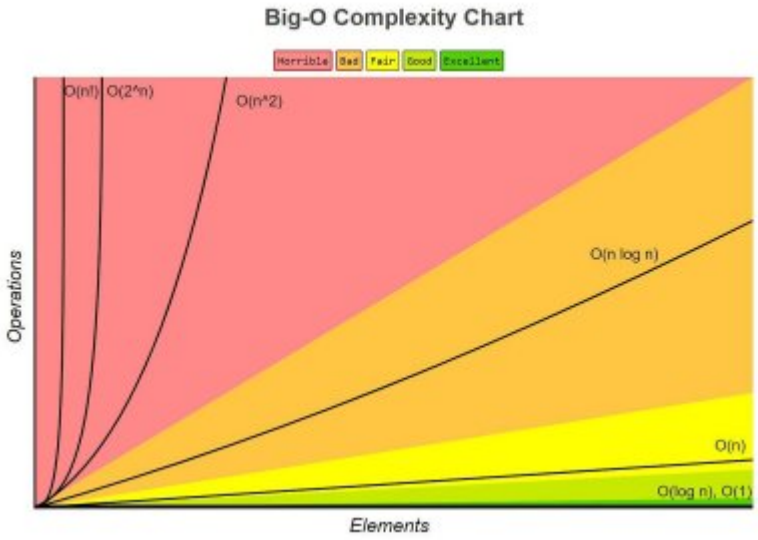
Pravidla programovacích jazyků

- **Syntaxe** (forma) – Definuje kombinaci symbolů, které jsou považovány za správně strukturovaný kód. V každém programovacím jazyce bývá syntaxe odlišná. Jejími typickými prvky jsou např. závorky, středníky, mezery a tabulátory. Je-li syntaxe chybná, výsledný program nelze zkompilovat.
- **Sémantika** (význam) – Zhodnocuje význam syntakticky platných řetězců a provádí jejich výpočty. Popisuje procesy, které počítač provádí při běhu programu. Například:
- *int number = 1 + 1;*
- Dojde k inicializaci deklarované proměnné „number“ – je do ní vložen výsledek součtu 1 + 1.

Vlastnosti algoritmu

- **Determinovanost** (Určenost)
  - Binární vlastnost - Odpovídá na otázku: „Je výsledek algoritmu vždy stejný pro stejné vstupní hodnoty?“
  - Algoritmus není determinovatelný pokud má vnitřní stav (například počítá kolikrát algoritmus byl spuštěn) nebo pokud používá skutečně náhodné hodnoty.
  - Základní kámen funkcionálního programování a jednou z podmínek takzvané "pure function"
- **Obecnost**
  - Popisuje pro jakou množinu vstupů algoritmus funguje.
  - Analogií obecnosti algoritmu je definiční obor funkce.
  - Algoritmus by měl řešit typovou úlohu co nejvíce obecně, například implementace kosinové věty je obecnější než implementace pythagorovy věty (funguje pro větší množinu úhlů).
- **Finitivnost** (Konečnost)
  - Binární vlastnost - Odpovídá na otázku: „Skončí algoritmus pro jakýkoliv vstup po konečně mnoha krocích?“
- **Resultativnost** (Výslednost)
  - Binární vlastnost - Odpovídá jestli má pro daný vstup algoritmus nějaký výstup nebo ne.
  - Algoritmus není resultativní pokud vrací void a neměnní vnější proměnné.
- **Korektnost** (Správnost)
  - Binární vlastnost - Vydá algoritmus pro všechny vstupy správný výsledek?
- **Efektivita**
  - Dělí se na paměťovou efektivitu (náročnost na paměť) a výpočetní efektivitu (náročnost

na výpočet), tyto dvě vlastnosti jsou často k sobě ve vztahu nepřímé úměry (například dynamické programování).
- Primárně se určuje pomocí velké O notace.
- Sekundárně se určuje podle praktických záležitostí (například cache miss nebo CPU branch prediction)



Známé algoritmy

Eratosthenovo síto

Algoritmus pro získání všech prvočísel od dvou po dané číslo.

Postup:

Krok 1: Vytvoření seznamu, obsahujícího všechna čísla v rozsahu 2 až n:

Krok 2: První číslo ze seznamu je zapsáno jako prvočíslo do seznamu prvočísel a ze seznamu je vymazáno společně se všemi jeho násobky.

Krok 3: Opakuj krok 2, dokud není původní seznam prázdný.

Krok 4: Seznam prvočísel obsahuje všechna prvočísla od 2 po n

Časová složitost: O(n log(log(n)))

Paměťová složitost: O(1)

Ukázka erastotenova síta na číslech od 2 do 120

Euklidův algoritmus

Algoritmus pro výpočet největšího společného dělitele (dále jen NSD) dvou čísel.

Zde příklad: jsou zadána dvě čísla 140 a 15.

Postup:

- Nejprve zjistíme zbytek po dělení většího čísla číslem menším. (V našem případě 140 = 9 * 15 + **5**)
- Nyní zopakujeme první krok, ale s dělením menšího čísla zbytkem po dělení (15 = 3 * 5 + **0**)
- Vyšel nám zbytek 0, takže NSD je rovno 5, pokud by nám nevyšel zbytek 0 aplikovali bychom znovu krok jedna

Časová složitost: O(log(min(a, b)))

Paměťová složitost: O(1)

Podrobnější vysvětlení

Djikstrův algoritmus

Algoritmus sloužící pro výběr nejlepší trasy z bodu A do bodu B.

Postup:

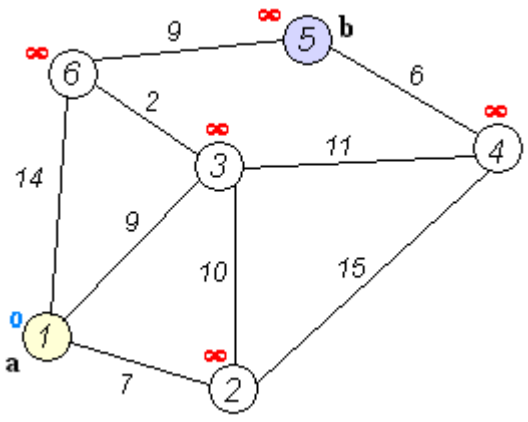
- Každá cesta mezi jednotlivými body dostane hodnotu, podle „náročnosti“ (délka trasy, povolená rychlost, ...).
- Algoritmus si postupně vypočítává délku cesty z bodu A do všech sousedních bodů a z nich do dalších bodů.
- Pokud algoritmus najde novou cestu do již objeveného bodu, pomalejší cestu k tomuto bodu odstraní.
- Na konci zůstane pouze jedna nejrychlejší cesta do každého z bodů.
- Algoritmus pak jako výstup vydá nejrychlejší cestu do požadovaného bodu B.

Nejhorší možná časová složitost: O(V^2)

Nějhorší možná paměťová složitost: O(V + H)

V = počet vrcholů

H = počet hran



Pravděpodobnostní algoritmy:

Algoritmus Las Vegas

Algoritmus hledající prvek žádaného typu v množině s více typy prvků. V základní podobě algoritmu je narušen jak princip determinismu, tak princip konečnosti (není-li běhový čas algoritmu či počet opakování cyklu nijak omezen, běhová doba algoritmu se teoreticky může blížit nekonečnu …).

Postup:

- Algoritmus vybere prvek množiny s náhodně přiděleným pořadovým číslem v rámci povoleného rozsahu odpovídajícím velikosti možiny.
- Poté algoritmus u získaného prvku ověřuje typ, je-li typ shodný s typem hledaným, prvek vrátí jako výsledný výstup.
- Pokud však typ prvku neodpovídá, postup se opakuje a to tak dlouho, dokud není nalezen prvek typově odpovídající.

Algoritmus Monte Carlo

Algoritmus s cílem analogickým k výše zmíněnému. Je alternativou k algoritmu Las Vegas, neboť nabízí konečnost (ukončení běhu cyklu v závislosti na parametru maximálního běhového času či počtu opakování cyklu) výměnou za jistou pravděpodobnost nedosažení cíle (není-li v rámci daného času či počtu opakování nalezen prvek žádaného typu, algoritmus skončí a vrátí se „s prázdnou“).

Postup:

- Algoritmus vybere prvek množiny s náhodně přiděleným pořadovým číslem v rámci povoleného rozsahu odpovídajícím velikosti možiny.
- Poté algoritmus u získaného prvku ověřuje typ, je-li typ shodný s typem hledaným, prvek vrátí jako výsledný výstup.
- Pokud však typ prvku neodpovídá a zároveň není dosaženo maximálního času či počtu opakování cyklu, postup je zopakován.