Složitost algoritmů

Hodnocení algoritmů by mělo být nezávislé na konkrétním počítači ⟶ při posuzování složitosti uvažujeme tzv. hypotetický počítač podobný skutečným počítačům, pro který platí :

- **paměť počítače je dostatečně velká**
- **výpočet se provádí sekvenčně**
- **vstupy a výstupy algoritmu jsou uloženy v paměti**

Posuzování složitosti - v závislosti na rozměru úlohy.
Rozměr úlohy N - obvykle počet vstupních dat
(např. mzdová agenda N = počet zaměstnanců, maticové výpočty N = řád matice).

Paměťová složitost - velikost potřebné paměti, chápaná jako funkce rozměru úlohy

Časová složitost - doba potřebná algoritmem k vyřešení úlohy, chápaná jako funkce rozměru úlohy.

Při posuzování složitosti se obvykle pracuje s tzv. asymptotickou složitostí - vyjadřuje chování úlohy pro N ⟶ ∞.

Např. algoritmus zpracovává úlohu rozměru N za čas $a*N^2 + b*N + c$ ⟶ časová složitost = N^2

K vyjádření skutečnosti, že funkce *g* roste asymptoticky tak rychle jako funkce *f*

$$g(N) = O(f(N))$$

Def. Nezáporná funkce g(N) je O(f(N)), existuje-li taková konstanta k>0 taková, že g(N) = k*f(N) pro všechna $N \geq N_0$, $N_0 \geq 0$.

Uvažujeme-li množinu všech legálních vstupů, pak O(f(N)) vyjadřuje v určitém smyslu nejhorší případ.

Dolní mez složitosti → nalezení lepšího algoritmu.
Uzavřené problémy: *horní mez složitosti* = *dolní mez složitosti* ⟶ nelze nalézt efektivnější algoritmus .

Algoritmy se složitostí O(1)

Instrukce programů takovýchto algoritmů se provede pouze jednou s malým počtem opakování bez ohledu na rozměr úlohy. Čas potřebný k provedení algoritmu je konstantní.

Algoritmy se složitostí O(logN)

Čas výpočtu je logaritmickou funkcí vzhledem k rozměrům úlohy. Při *k* násobném zvětšení A se prodlouží doba $log_k N$ krát.
Do této kategorie patří algoritmy, které řeší problém jako transformaci na problém k-krát menšího rozsahu a tento postup se opakuje až do nalezení řešení.
Př. Vyhledávání metodou půlení intervalů.

Algoritmy se složitostí O(N)

Čas výpočtu je lineární funkcí rozměru úlohy ⟶
S každým vstupním elementem se provádí zhruba stejný počet instrukcí. Je to nejmenší možná složitost, pokud požadujeme N výstupních dat.
Př. Některé algoritmy řazení

Algoritmy se složitostí O(N*logN)

Problém se řeší rozkladem na podproblémy, podproblémy se řeší nezávisle a řešení se pak kombinují do výsledného řešení. Nárůst času výpočtu je trochu horší než lineární.
Př. Mergesort (v paměti), algoritmus rychlé Fourierovy transformace (FFT).

Algoritmy se složitostí $O(N^2)$

Čas výpočtu je kvadratickou funkcí rozměru úlohy. Patří sem algoritmy, které zpracovávají všechny možné kombinace vstupních dat. Pro tyto algoritmy je charakteristický dvojnásobný (vnořený) cyklus.
Př. Algoritmy řazení (SelectSort, InsertSort, BubbleSort).

Algoritmy se složitostí $O(N^3)$

Úlohy kubicky závislé na rozměru úlohy, charakteristický pro tyto úlohy je trojnásobný (vnořený) cyklus.
Př. Násobení matic řádu N.

Algoritmy se složitostí $O(2^N)$

Pouze málo algoritmů s touto složitostí má praktické využití. Obvykle jsou to problémy pro které je možné vytvořit strom řešení a žádnou y možností nelze v tomto stromu vyloučit.
Větví-li se strom v každém kroku na *k* možných řešení (každý uzel stromu má stupeň uzlu roven *k*) je složitost rovna $O(k^N)$.