Niekoľko slov o efektivite algoritmov

Úlohu na zostrojenie algoritmu riešiaceho daný problém možno riešiť spravidla mnohými rôznymi spôsobmi. Aby sme si z viacerých správne fungujúcich algoritmov mohli vybrať ten najlepší, potrebujeme mať nejaké kritérium, podľa ktorého by sme mohli algoritmy porovnávať. Algoritmy je možné hodnotiť z rôznych hľadísk, napríklad podľa toho, aké ľahké je pre nás takýto algoritmus napísať.

Kritériá, ktoré sa obyčajne používajú pri hodnotení kvality algoritmov, sa nazývajú časová a pamäťová zložitosť (alebo tiež efektivita) algoritmov.

Časovú zložitosť algoritmu môžeme definovať ako závislosť časových nárokov algoritmu na veľkosti konkrétneho riešeného problému alebo konkrétnych vstupných údajov. Neudávame ju však v žiadnych bežných časových jednotkách, pretože skutočná doba výpočtu programu realizujúceho navrhnutý algoritmus závisí tiež na množstve čisto technických parametrov, ako je napríklad rýchlosť procesora použitého počítača, zatiaľ čo my chceme charakterizovať rýchlosť práce algoritmu samého.

Budeme preto časovú zložitosť vyjadrovať počtom elementárnych operácií (aritmetických, logických, porovnaní apod.),

ktoré budú vykonané pri výpočte algoritmu nad zvolenými vstupnými údajmi. Časová zložitosť algoritmu A je teda funkcia t_A, ktorá každej hodnote N, charakterizujúcej veľkosť spracovávaných údajov, priraďuje hodnotu $t_A(N)$, čo je počet operácií, ktoré algoritmus A vykoná pri spracovaní údajov veľkosti N.

Je potrebné rozlišovať časovú zložitosť algoritmu v najhoršom prípade a časovú zložitosť v priemernom prípade. **Časová zložitosť v najhoršom prípade** je funkcia udávajúca pre každú hodnotu N ako najdlhšie môže trvať výpočet algoritmu s ľubovoľnými údajmi veľkosti N. Naproti tomu **časová zložitosť v priemernom prípade** nám hovorí, ako dlho bude algoritmu v priemere trvať spracovanie vstupných údajov veľkosti N. Nie je to teda už horný odhad, ale priemerná očakávaná doba výpočtu pre údaje veľkosti N, získaná na základe úvah o všetkých možných konkrétnych hodnotách vstupných údajov veľkosti N.

Pamäťovú zložitosť algoritmu definujeme obdobne ako závislosť pamäťových nárokov algoritmu na veľkosti riešeného problému. Vyjadrujeme ju ako počet premenných, resp. pamäťových miest, ktoré algoritmus pri výpočte so zvolenými vstupnými údajmi potrebuje. Pamäťová zložitosť algoritmu A je teda opäť istou funkciou m_A, ktorá veľkosti vstupných údajov veľkosti N priraďuje potrebný počet pamäťových miest $m_A(N)$, ktoré bude algoritmus A pri výpočte s týmito údajmi využívať.

Algoritmus je predpis na riešenie celej triedy navzájom podobných úloh, líšiacich sa zadaním vstupných údajov. Veľkosťou vstupných údajov rozumieme počet vstupujúcich čísel alebo znakov, ktoré algoritmus spracuje. Veľkosť vstupných údajov však väčšinou nezávisí na jednotlivých hodnotách týchto čísel alebo znakov. Tak napríklad rýchlosť algoritmu na utriedenie N čísel podľa veľkosti je daná práve týmto počtom triedených čísel N, ale obyčajne už nezávisí na tom, aké čísla vlastne triedime.

Ak máme dva rôzne algoritmy riešiace daný problém, ktorý z nich je lepší? Ak vezmeme za základ nášho hodnotenia algoritmov časové hľadisko, je lepší ten z nich, ktorý je rýchlejší, tzn. ktorý má menšiu časovú zložitosť. Nie je pritom príliš dôležité, ktorý z algoritmov má menšiu časové nároky pre malé hodnoty veľkosti vstupných údajov N, pretože malé údaje bude takmer vždy možné algoritmicky spracovať v rozumnom čase. Podstatné je, ako sa chová funkcia vyjadrujúca časovú zložitosť algoritmov pre rastúce hodnoty N. Hovoríme preto niekedy aj o **asymptotickej časovej zložitosti**. Časová zložitosť algoritmu býva obyčajne rastúcou funkciou. Z dvoch algoritmov budeme teda za časovo výhodnejší a efektívnejší považovať ten, ktorého funkcia časovej zložitosti s rastúcou hodnotou N rastie pomalšie.

Pritom je ale možné a bežne sa stáva, že pre niektoré malé vstupné údaje pracuje algoritmus s väčšou asymptotickou časovou zložitosťou rýchlejšie. Keby mal napríklad algoritmus A1 časovú zložitosť $t_{A1}=N^2$ a algoritmus A2 s časovou zložitosťou $t_{A2}=10.N$, má iste A1 väčšiu asymptotickú časovú zložitosť než A2 (funkcia t_{A1}

rastie rýchlejšie než t_{A2} pre rastúce N), ale pre N<10 je $t_{A1}(N)<t_{A2}(N)$ a teda A1 pracuje nad malými vstupnými údajmi rýchlejšie než A2. Pri praktickom riešení úloh na počítači je preto potrebné pri voľbe vhodných algoritmov zvažovať aj to, aké veľké vstupné údaje bude program spracovávať. Ak na veľkosť očakávaných vstupných údajov nie je vopred dané žiadne obmedzenie, zvolíme samozrejme algoritmus s menšou asymptotickou časovou zložitosťou.

Pri určovaní asymptotickej časovej zložitosti algoritmov stačí charakterizovať iba rýchlosť rastu príslušnej funkcie. Presný počet elementárnych operácií, ktoré je potrebné uskutočniť pri spracovaní vstupných údajov veľkosti N, nie je tak dôležitý a nie je zvyčajne ľahké "spočítať všetky drobné". Ak máme napríklad algoritmus A s časovou zložitosťou danou funkciou $t_A(N)=3N^2+2n-4$, hovoríme, že algoritmus A má kvadratickú časovú zložitosť $O(N^2)$. Algoritmy s časovou zložitosťou O(N) sa nazývajú lineárne, so zložitosťou $O(N^3)$ kubické, so zložitosťou $O(c^N)$ exponenciálne. Zatiaľ čo hodnota polynómu (lineárneho, kvadratického, kubického, ...) hoci aj vyššieho stupňa býva pre bežné N ešte prijateľná a príslušný algoritmus s takouto časovou zložitosťou býva z hľadiska doby trvania výpočtu použiteľný, rýchlosť rastu exponenciálnej funkcie je tak veľká, že exponenciálne algoritmy možno použiť len pre veľmi malé N. Pri návrhu algoritmu sa preto snažíme vyhýbať exponenciálnym algoritmom všade, kde je to možné.

I keď sa obmedzíme iba na časovú a pamäťovú zložitosť algoritmov pri hľadaní toho najlepšieho, môžeme sa dostať do ťažkej situácie, pretože tieto dve kritériá často stoja proti sebe. Stáva sa totiž, že k zrýchleniu výpočtu musíme použiť nejakú pomocnú dátovú štruktúru slúžiacu k uchovávaniu vopred spočítaných a pripravených hodnôt. Najrýchlejší možný algoritmus riešiaci úlohu potom nie je optimálny z hľadiska pamäťových nárokov a naopak algoritmus s najmenšou pamäťovou zložitosťou zase nemusí byť najrýchlejší.

V súčasnej dobe sa obyčajne dáva prednosť časovému hľadisku a hľadajú sa čo najrýchlejšie algoritmy, a to i za cenu potreby dodatočnej pamäte.