Zložitosť

Obsah:

- cieľ a zámer vyjadrovania zložitosti
- O-notácia
- očakávaná zložitosť od veľkosti vstupu
- príklady na notáciu

Keď riešime nejaké problémy, môže sa stať, že vymyslím viacero riešení. Ako však povedať, ktoré je lepšie a vôbec, čo to znamená byť lepším? O tom všetkom sa porozprávame práve v tejto kapitole.

Dôležitým aspektom je rýchlosť daného algoritmu. To znamená, ako rýchlo sa pre nejaký vstup doráta k výsledku. Nechceme predsa čakať na výsledok niekoľko hodín, ak sa dá vyrátať v priebehu sekúnd. Poďme sa teda zamýšľať nad takzvanou `časovou zložitosťou` algoritmov.

Hneď na začiatku sa však stretávame s množstvom problémov. Prvým je to, že nechceme porovnávať skutočný čas výpočtu. Ten je totiž príliš ovplyvnený počítačom, na ktorom daný program spúšťame. Nečakáme predsa, že na počítači našej babky bude bežať rovnaký algoritmus takisto rýchlo ako na počítači v Googli. Musíme sa teda snažiť oddeliť časovú zložitosť algoritmu od výpočtu na skutočnom počítači. Bolo by teda fajn, ak sme vedeli odhadnúť rýchlosť programu z obyčajného pohľadu naň.

Našťastie to však vôbec nie je také ťažké. Skúste si to sami. Tu vidíte tri programy, ktoré všetky rátajú tú istú vec – súčet prvých n čísiel.

Myslím, že aj vám je na prvý pohľad jasné, ktorý z týchto programov je najrýchlejší a naopak, ktorý je najhorší. Vynásobiť dve čísla je totiž určite jednoduchšie a rýchlejšie, ako pričítavať po jednotkách k výsledku.

Keď sa pozrieme na druhý a tretí program, vieme veľmi ľahko povedať, koľko sčítaní, ktorý z nich potrebuje. Druhý spraví n sčítaní, tretí spraví $n(n-1)/2$ sčítaní. A dá sa predpokladať, že sčítanie trvá na konkrétnom počítači vždy rovnako dlho. Zrazu teda vieme bez ohľadu na rýchlosť konkrétneho počítača povedať, ktorý je rýchlejší. Jeden totiž urobí o dosť viac sčítaní ako druhý. Operáciu sčítanie potom nazvime `elementárnou operáciou` – to znamená, že táto operácia trvá počítaču krátky čas, ktorý je naviac nemenný. Aj keď je to prekvapivé, počítaču trvá rovnako dlho vyrátať $1 + 1$ ako $1574123 + 478413$.

Nezastavme sa však len pri sčítaní. Je aj viac operácií, ktoré vie počítač vykonávať veľmi rýchlo, a teda môžu byť považované za elementárne. Väčšinou do nich patria všetky aritmetické operácie ako sčítanie, odčítanie, násobenie, delenie, zvyšok po delení, všetky logické operácie – and, or, xor ... Ďalej tam patrí aj príkaz priradenia, alebo náhľad do poľa či inicializovanie jednej premennej, načítanie alebo výpis jedného znaku.

Reálne je to bohužiaľ trošku komplikovanejšie, napríklad zvyšok po delení je náročnejší ako bitový and. Ale nám stačí takáto abstrakcia. Pri veľkých vstupoch to málokedy hrá hlavnú úlohu a dodáva to oveľa lepšiu možnosť sa vyjadrovať. A o tom, ako je to v skutočnosti, sa môžeme pobaviť neskôr.

Vieme teda už počítať elementárne operácie programov. Zrazu je nám úplne jasné, prečo je prvý program najrýchlejší. Stačí mu predsa spraviť jednu jedinú operáciu. A aj u zvyšných dvoch je jasné, ktorý je lepší. Tu však problémy nekončia. Predstavme si, že máme dva programy, ktoré na vstupe dostanú jedno číslo n a niečo s ním rátajú. Prvý urobí $20n$ operácií, druhý n^2. Ktorý z nich je rýchlejší? Pre $n < 20$ je to program druhý, potom už vždy vyhrá program prvý. Intuitívne je teda prvý program lepší, veď na väčšine vstupov je rýchlejší. My si však túto intuíciu poriadne definujeme.

Asymptotická zložitosť

Uvedomme si, že počet elementárnych operácií programu sa dá vyjadriť ako funkcia od premenných na vstupe. Ak máme na vstupe číslo n, bude to funkcia tohto čísla. A my chceme nejak porovnávať tieto funkcie. Dobrá miera na porovnávanie je schopnosť rásť. Čím rýchlejšie funkcia rastie, tým viac operácií bude treba. Takisto nás poväčšine nezaujímajú malé vstupy, ale také, na ktorých zrátanie treba milióny operácií. A samozrejme, nechceme byť zbytočne presný. Však 1000000 a 1000042 je skoro to isté. Hore dole 42 operácií, také niečo môžeme zanedbať.

A na toto všetko je vytvorená takzvaná O-notácia a asymptotické určovanie zložitosti. Hneď vás varujem. Nasledujúce odseky budú obsahovať nejaké formálne definície aj niektoré neformálne tvrdenia. Ono je to celé veľmi intuitívne a keď s tým začnete pracovať, rýchlo si na to zvyknete. Môže to byť však zo začiatku trochu mätúce, pokúsim sa vám to však podať čo najlepšie.

Začnime s príkladom. Majme program, ktorý vykoná $f(n) = n^2/2 + 5n + 4$. Prvé pravidlo znie, že konštanty zanedbáme. To, či urobíme operácií n alebo $5n$ je skoro jedno, zdvihne sa to len o konštantu, ktorá nezávisí

od veľkosti vstupu. Zrazu máme teda $f'(n) = n^2 + n$. Druhé pravidlo zase hovorí o tom, že nás zaujíma len najväčší člen funkcie, teda ten, čo najrýchlejšie rastie. V tomto prípade n^2 rastie oveľa rýchlejšie ako n. Tým pádom dostaneme výslednú funkciu $f''(n) = n^2$. A to je náš odhad na to, koľko operácii spraví náš program.

Možno sa to zdá hrubé, koľko vecí sme zanedbali, ale ono to vôbec nie je tak. Tých zanedbaných vecí je najviac konštantne viac ako to čo nám zostalo. A to nám stačí. Samozrejme takýto spôsob škrtania je škaredý a tieto dve pravidlá sú len pomôcky. Potrebujeme jasnú definíciu a tou je už spomínaná O-notácia.

Definícia: Majme dve funkcie f a g. Hovoríme, že $f \in O(g)$, ak existuje n_0 a $c > 0$ také, že pre všetky $n \geq n_0$ platí $|f(n)| \leq c|g(n)|$.

A slovami: Hovoríme, že funkcia f patrí do $O(g)$ (všimnite si, že $O(g)$ je množina všetkých funkcií, pre ktoré platí daná vlastnosť), ak existujú také konštanty n_0 a c, že pre dosť veľké n (lebo $n \geq n_0$) je $c|g(n)|$ (absolútna hodnota) väčšia ako $f(n)$.

Teraz už ľahko určíme, že naozaj naše $f(n) = n^2/2 + 5n + 4 = O(n^2)$ (tento zápis síce nie je úplne korektný, lebo $O(n^2)$ je množina, napriek tomu sa často používa). Stačí, ak si zvolíme $n_0 = 10$ a $c = 2$. Ľahko potom nahliadneme, že $n^2/2 + 5n + 4 \leq 2n^2$, lebo toto je vlastne $5n + 4 \leq 3n^2/2$. Táto nerovnosť pre $n = 10$ platí a ďalej bude už len masívne narastať.

Tiež si všimnite, že napríklad $n^3 \notin O(n^2)$. Ak si zvolíte totiž ľubovoľné c, tak n^3 je väčšie od cn^2 pre všetky $n \geq c$. Naopak však platí, že $n^2 \in O(n^3)$, lebo to vyhovuje definícii pre $c = n_0 = 1$.

Najhorší prípad a dolná hranica

Samozrejme, nikto neočakáva, že hneď pochopíte, čo to vlastne znamená. Postupne sa však s O-notáciou budete stretávať neustále. Práve v nej sa totiž neustále vyjadruje časová aj pamäťová zložitosť všetkých algoritmov. My si ukážeme pár funkcií, s ktorými sa stretnete najčastejšie.

Ešte sme si však nepovedali, na ktorom vstupe rátame zložitosť. Môže sa totiž stať, že niekde bude náš program potrebovať len $O(n)$ operácií a inde zase $O(n^2)$. V tomto prípade samozrejme berieme do úvahy najhorší možný príklad a zložitosť výpočtu na ňom prehlásime za zložitosť riešenia.

Ďalšia vec je, že my informatici sme tak trochu flákači. Vyjadrovanie sa v O-notácii je totiž mierne nepresné. Ako sme totiž ukazovali, tak $n^2 = O(n^3)$. Ak máme teda algoritmus so zložitosťou $f(n)$, nič nám nebráni povedať, že to je vlastne $O(n^3)$. Nás však zaujíma takpovediac najmenšia dolná hranica, v tomto prípade n^2. Takúto zložitosť by ste mali udávať aj všade, kde po vás požadujú zložitosť.