Pete vaje APS2: krovni izrek in Karacubov algoritem

1 Krovni izrek

Krovni izrek uporabljamo za reševanje določenih rekurenčnih enačb za časovno zahtevnost, ki lahko nastanejo pri ocenjevanju časovne zahtevnosti algoritmov z rekurzivnim razcepom (»deli in vladaj«). Če nalogo velikosti n rešimo tako, da rešimo a nalog velikosti n/b, ki so enake oblike kot izhodiščna naloga, za razbijanje naloge na podnaloge in sestavljanje rešitev podnalog v rešitev izhodiščne naloge pa porabimo $\Theta(n^d)$ časa, potem za celotno rešitev potrebujemo

$$T(n) = aT(n/b) + \Theta(n^d)$$

časa.

Formulo zapišimo kot $T(n) = kn^d + aT(n/b)$ in jo vstavljajmo samo vase:[1]

$$\begin{aligned}
T(n) &= kn^d + aT(n/b) \\
&= kn^d + a(k(n/b)^d + aT(n/b^2)) \\
&= kn^d + ak(n/b)^d + a^2T(n/b^2) \\
&= kn^d + ak(n/b)^d + a^2(k(n/b^2)^d + aT(n/b^3)) \\
&= kn^d + ak(n/b)^d + a^2k(n/b^2)^d + a^3T(n/b^3) \\
&= kn^d + ak(n/b)^d + a^2k(n/b^2)^d + a^3k(n/b^3)^d + a^4T(n/b^4) \\
&= \dots \\
&= \underbrace{kn^d + ak(n/b)^d + a^2k(n/b^2)^d + a^3k(n/b^3)^d + a^4k(n/b^4)^d + \dots}_{\log_b n \text{ členov}} \\
&= kn^d \left(\underbrace{1 + \frac{a}{b^d} + \left(\frac{a}{b^d}\right)^2 + \left(\frac{a}{b^d}\right)^3 + \left(\frac{a}{b^d}\right)^4 + \dots}_{\log_b n \text{ členov}} \right). \qquad (1)
\end{aligned}$$

[1] Pravilneje bi bilo pisati $k_1 n^d aT(n/b) \leq T(n) \leq k_2 n^d + aT(n/b)$, zanemarili pa bomo še nekaj drugih podrobnosti. Natančen dokaz je v Cormenu.

Zakaj je število členov (približno) enako $\log_b n$? Zato, ker se zaporedje $T(n)$, $T(n/b)$, $T(n/b^2)$, $T(n/b^3)$, ... zaključi, ko imenovalec doseže ali preseže števec.

Sedaj imamo tri možnosti:

(1) $\underline{a < b^d}$. V tem primeru lahko izraz v oklepaju aproksimiramo z geometrijsko vrsto. Vemo, da je $1 + q + q^2 + q^3 + \ldots = 1/(1-q)$, če je $q < 1$. V našem primeru je $q = a/b^d$, zato dobimo

$$T(n) \leq kn^d \frac{1}{1 - a/b^d} = kn^d \frac{b^d}{b^d - a}.$$

Ker sta k in $b^d/(b^d - a)$ konstanti, lahko zapišemo

$$T(n) = O(n^d).$$

(2) $\underline{a = b^d}$. V tem primeru so vsi členi v oklepajih v (1) enaki 1, zato dobimo

$$T(n) = kn^d \log_b n = O(n^d \log n).$$

(3) $\underline{a > b^d}$. V tem primeru je izraz v oklepajih v (1) vsota končnega geometrijskega zaporedja. Spomnimo se, da je $1 + q + q^2 + \ldots + q^{m-1} = (q^m - 1)/(q - 1)$. Pri nas velja $q = a/b^d$ in $m = \log_b n$:

$$T(n) = kn^d \frac{\left(\frac{a}{b^d}\right)^{\log_b n} - 1}{\frac{a}{b^d} - 1}.$$

Če vse konstante združimo v novo konstanto K in »zanemarimo« odštevanec, lahko zapišemo

$$\begin{aligned}
T(n) &\leq Kn^d \left(\frac{a}{b^d}\right)^{\log_b n} \\
&= Kn^d \frac{a^{\log_b n}}{b^{d \log_b n}} \\
&= Kn^d \frac{a^{\log_b n}}{(b^{\log_b n})^d} \\
&= Kn^d \frac{a^{\log_b n}}{n^d} \\
&= Ka^{\log_b n} \\
&= K(n^{\log_n a})^{\log b_n} \\
&= Kn^{\log_b n \log_n a} \\
&= Kn^{\log_b a} \\
&= O(n^{\log_b a}).
\end{aligned}$$

Velja torej:

$$T(n) = \begin{cases} O(n^d), & \text{če } a < b^d; \\ O(n^d \log n), & \text{če } a = b^d; \\ O(n^{\log_b a}), & \text{če } a > b^d. \end{cases}$$

Izkaže se, da asimptotične meje niso samo zgornje, ampak tudi spodnje, zato lahko vse $O(.)$ zamenjamo s $\Theta(.)$.

2 Karacubov algoritem

Karacubov algoritem uporabljamo za množenje velikih naravnih števil. Predpostavili bomo, da so števila desetiška, vendar pa bi algoritem zlahka prilagodili tudi za druge številske osnove.

Če sta a in b enomestni števili, lahko njun zmnožek izračunamo v času $\Theta(1)$. V nasprotnem primeru pa predpostavimo, da sta obe števili n-mestni, kjer je n sodo število; če to ni res, ju pač dopolnimo z vodilnimi ničlami. Števili zapišimo takole:

$$a = 10^{n/2} a_1 + a_0,$$
$$b = 10^{n/2} b_1 + b_0.$$

Je tak zapis že dovolj, da običajno časovno zahtevnost množenja ($\Theta(n^2)$) zmanjšamo? Poglejmo:

$$ab = 10^n a_1 b_1 + 10^{n/2}(a_1 b_0 + a_0 b_1) + a_0 b_0$$

Problem množenja n-mestnih števil torej razstavimo na štiri ($a = 4$) probleme velikosti $n/2$ ($b = 2$), za združevanje rezultatov podproblemov pa potrebujemo čas $\Theta(n)$ ($d = 1$); množenje s potenco števila 10 seveda ni »pravo« množenje, saj gre zgolj za zamikanje števk. Ker je $a > b^d$, je po krovnem izreku $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 4}) = \Theta(n^2)$. Šment!

No, k sreči se izkaže, da lahko s preprosto matematično telovadbo eno množenje prihranimo. Uvedimo

$$c_2 = a_1 b_1,$$
$$c_0 = a_0 b_0,$$
$$c_1 = (a_1 + a_0)(b_1 + b_0) - c_2 - c_0.$$

Ni težko pokazati, da velja

$$ab = 10^n c_2 + 10^{n/2} c_1 + c_0.$$

Ker za izračun koeficientov c_0, c_1 in c_2 potrebujemo zgolj tri množenja, smo problem množenja n-mestnih števil to pot razstavili na tri ($a = 3$) probleme velikosti $n/2$ ($b = 2$), za združevanje rezultatov podproblemov pa še vedno

potrebujemo $\Theta(n)$ časa ($d = 1$). Vnovič je $a > b^d$, zato po krovnem izreku dobimo $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 3}) = O(n^{1.59})$. Jupiii!

Za lažje razumevanje algoritma zmnožimo števili $a = 123$ in $b = 45$ ($n = 4$). Velja $a_1 = 1$, $a_0 = 23$, $b_1 = 0$ in $b_0 = 45$. Izračunamo $c_2 = a_1 b_1 = 1 \cdot 0 = 0$, $c_0 = a_0 b_0 = 23 \cdot 45 = 1035$ ($c'_2 = 2 \cdot 4 = 8$, $c'_0 = 3 \cdot 5 = 15$, $c'_1 = 5 \cdot 9 - 8 - 15 = 22$, $a_0 b_0 = 100c'_2 + 10c'_1 + c'_0 = 800 + 220 + 15 = 1035$) in $c_1 = (a_1 + a_0)(b_1 + b_0) - c_2 - c_0 = 24 \cdot 45 - 0 - 1035 = 45$ (seveda tudi $24 \cdot 45$ izračunamo rekurzivno). Iskani zmnožek je torej $ab = 10^4 c_2 + 10^2 c_1 + c_0 = 10^4 \cdot 0 + 100 \cdot 45 + 1035 = 5535$.

Kako pa se Karacuba obnese v praksi? V skladu s pričakovanji, a le pod pogojem, da rekurzijo na primerni točki odrežemo in zmnožke »dovolj majhnih« (največ M-mestnih) števil izračunamo po klasičnem kvadratnem postopku. Pri moji implementaciji se je splačalo vzeti $M = 64$. S povečevanjem M se čedalje bolj čutijo negativni vplivi kvadrata, z zmanjševanjem pa stroški, povezani z rekurzivnim razcepom.