Algoritam pismenog logaritmiranja

Domagoj Oreški[1]

Predgovor i uvod

Najranije službene naznake svijesti ljudi o postojanju logaritama datiraju još s početka 17. stoljeća, iz radova škotskog matematičara Josha Napiera (rad objavljen 1614. g.) i švicarskog matematičara Josta Bürgija (rad objavljen 1620. g.), iako su logaritamske tablice postojale i prije toga. Funkcija kao koncept još nije osuvremenjena sve do kraja istog stoljeća, stoga su se logaritmi u ulozi funkcija pronašli tek nešto kasnije. Njihova je primarna svrha traženje eksponenta u eksponencijalnim funkcijama pa su logaritamske funkcije, kao takve, inverzne funkcije eksponencijalnih funkcija.

U školama smo učili osnovna logaritamska pravila i kako drugačije (najčešće pomoću potencija) zapisati argumente logaritma, te s ostalim pravilima presložiti oblik jednadžbe kako bismo mogli ustvrditi rješenje zadanog logaritma.

Uvijek nam je bilo napomenuto da postoje i logaritmi čije će rješenje biti iracionalan broj, te da je zapravo velika većina logaritama iracionalnog rješenja; svi logaritmi koji se rade u školama namješteni su tako da rješenje uvijek bude racionalan broj, i to radi vježbanja primjene i svojstava logaritama. Za evaluaciju logaritamskih jednadžbi čije bi rješenje bio iracionalan broj koristili smo ili kalkulator, ili unaprijed pripremljene logaritamske tablice. No, odakle su ljudi koji su programirali kalkulatore ili popisivali vrijednosti u logaritamske tablice dobivali te rezultate?

Prve poznate tablice logaritama potječu iz 16. stoljeća, ali prve pregledne i u praktičnom računu uporabljive s početka su 17. stoljeća. Vrijednosti su dobivane kojekakvim računskim trikovima; jedna od njih je tzv. *radix* metoda pomoću korjenovanja. Primjerice, za $a=1, b=10$ imamo $c=\sqrt{ab}=3.16$, pa je $\log(3.16)=0.5$; slijedi da je $d=\sqrt{bc}=5.62$, pa je $\log 5.62=0.75$, i na taj je način popunjavana tablica. Izračun logaritama pomoću beskonačnih redova također je poznat od kraja 17. stoljeća, a Taylorov red, poznat i koristan i dan danas, služio je za popunjavanje logaritamskih tablica od početka 18. stoljeća.

No, postoji li ikakav algoritam koji bi se mogao sprovoditi isključivo pomoću četiriju osnovnih računskih operacija (zbrajanja, oduzimanja, množenja i dijeljenja)

[1]Domagoj Oreški, student Fakulteta organizacije i informatike iz Varaždina

samo pomoću papira i olovke (eventualno i kalkulatora, ali takvog koji na sebi sadrži samo četiri osnovne računaske operacije), odnosno takav da se vrijednost logaritma s iracionalnim rješenjem može računati do željene preciznosti bez uporabe algoritama koje koriste računala, već nekog jednostavnijeg, praktičnijeg, primijenjivog prosječnoj osobi bez računala?

Odgovor na to pitanje jest: **da, postoji**, ali budući da nije intuitivno jasan, pozvat ćemo se na neka poznata svojstva logaritama, preurediti izgled argumenta koji se logaritmira po zadanoj bazi, te ćemo na primjeru dokazati njegovu valjanost.

Što mi želimo? Primarno i jasno, želimo se baviti isključivo racionalnim brojevima jer jedino s takvima može postojati računica kod ručnih algoritama. Drugo, kako postupno aproksimirati vrijednost zadanog logaritma s iracionalnim rješenjem, a da se ne izgubi dosljednost, tj. da sa sigurnošću znamo da naša aproksimacija teži upravo pravoj vrijednosti zadanog logaritma? Treće, koja logaritamska pravila koristiti, te kako ih serijski primjenjivati da aproksimacija bude sve bolja – a opet da sadrži samo osnovne računske operacije koje svi znamo računati pismeno već od nižih razreda osnovne škole?

Konstrukcija algoritma

Standardni oblik logaritma

Kako bismo mogli pismeno logaritmirati, moramo se dogovoriti za neki zadani oblik logaritma. Jedini oblik s kojim ćemo moći izvoditi algoritam je logaritam oblika

$$\log_a b, \quad a>1, \quad b>1.$$

Takav oblik logaritma nazivat ćemo **standardnim oblikom**. Svaki se logaritam može svesti na njemu odgovarajući standardni oblik, ovisno o tome trebamo li mijenjati bazu ili argument logaritma:

$$\log_a b = -\log_{\frac{1}{a}} b, \quad a<1, \quad b>1;$$

$$\log_a b = -\log_a \frac{1}{b}, \quad a>1, \quad b<1;$$

$$\log_a b = \log_{\frac{1}{a}} \frac{1}{b}, \quad a<1, \quad b<1.$$

Dakle, dokle god su i baza i argument logaritma realni brojevi veći od 1, moguće je nad njima izvoditi algoritam pismenog logaritmiranja koji nadolazi. Mi ćemo, radi jednostavnosti demonstracije i pojašnjenja, koristiti prirodne brojeve i za bazu, i za argument logaritma, no dokaz će vrijediti općenito za bilo koje realne brojeve strogo veće od 1.

Referentna logaritamska pravila potrebna za naš algoritam pismenog logaritmiranja

Najbitnije pravilo, na kojemu će počivati cijeli algoritam, sljedeća je ekvivalencija:

$$\log_a a^n \leq \log_a b < \log_a a^{n+1} \Leftrightarrow n \leq \log_a b < n+1.$$

Kako smo obrazložili u prethodnom poglavlju, svako se računanje logaritma može svesti na računanje logaritma zapisanog u standardnom obliku. Dakle, u nastavku pretpostavljamo da računamo logaritme oblika

$$\log_a b, \quad a > 1, \quad b > 1.$$

Pa, što nam je prvo učiniti? Najbolje da prvo zapišemo argument logaritma najbolje što možemo pomoću cjelobrojne potencije pripadne baze:

$$b = ca^n, \quad n \in \mathbb{N}_0, \quad c \in \mathbb{R}, \quad 1 \leq c < a.$$

Varijablu c nazivat ćemo **česticom argumenta**. Ona nam predstavlja realni faktor strogo manji od baze koji nam nedostaje u zapisu argumenta logaritma samo pomoću najvećeg mogućeg cjelobrojnog eksponenta s bazom jednakom bazi logaritma. Iz prvog navedenog pravila i definicije čestice c, ovdje uviđamo da vrijedi

$$n \leq \log_a b = \log_a ca^n < n+1.$$

Formalno, za sada smo aproksimirali vrijednost zadanog logaritma s preciznošću od 1.

Bez navođenja/korištenja nekih drugih logaritamskih pravila, najbolja aproksimacija koju možemo dobiti jest upravo ova – cjelobrojna. To nikako nije dovoljno. Želimo imati rekurzivni algoritam koji će nam ovisno o duljini njegova izvođenja dati logaritam do željenog reda veličine preciznosti.

Sljedeće pravilo logaritama koje nam treba i koje će nam generirati naš pismeni algoritam jest

$$\log_a b = \frac{1}{2}\log_a b^2.$$

Kada god argument logaritma kvadriramo, moramo čitav logaritam pomnožiti jednom polovinom. Budući da bazu logaritma nismo dirali, lako možemo novi argument ponovo zapisati kao umnožak cjelobrojne potencije njegove baze i čestice argumenta, tj.

$$\log_a b = \log_a \left(ca^n\right) = \frac{1}{2}\log_a \left(ca^n\right)^2 = \frac{1}{2}\log_a \left(c^2 a^{2n}\right).$$

Sada općenito ne znamo je li naša novonastala čestica c^2 još uvijek čestica po definiciji ($<a$) ili je možda jednaka ili veća bazi. Znamo da novonastala čestica ne može biti veća ili jednaka a^2 jer je početna čestica bila strogo manja od a. Ukoliko je još uvijek čestica, ništa ne diramo. Ukoliko je veća ili jednaka bazi, nju dijelimo bazom, a eksponentu baze a dodajemo novu jedinicu u eksponent. Dakle, imamo dva

slučaja – kada novonastala čestica ostaje čestica, te kada novonastala čestica postaje veća od baze:

$$\log_a b = \frac{1}{2}\log_a (c^2 a^{2n}), \quad 1 \le c^2 < a \qquad \Rightarrow \qquad \frac{2n}{2} \le \log_a b < \frac{2n+1}{2}$$

$$\log_a b = \frac{1}{2}\log_a (c^2 a^{2n}), \quad a \le c^2 < a^2 \qquad \Rightarrow \qquad \frac{2n+1}{2} \le \log_a b < \frac{2n+2}{2}.$$

Novonastala čestica, nakon što smo je podijelili bazom logaritma ukoliko je ispala veća ili jednaka bazi logaritma, jednostavnom nam je usporedbom s bazom logaritma odala aproksimaciju zadanog logaritma s preciznošću od $\frac{1}{2}$. Bazu logaritma nismo dirali, ni potenciju u argumentu s bazom jednakom bazi logaritma i cjelobrojnim eksponentom. Jedino smo provjerili je li čestica nakon njenog kvadriranja postala veća ili jednaka bazi logaritma. Ukoliko jest, tu vrijednost prepišemo faktoru s potencijom. Ukoliko ne, sve ostaje isto.

Kontinuiran postupak

Dakle, jednostavnim smo kvadriranjem argumenta te pripisivanjem odgovarajućeg faktora od jedne polovine ispred čitavog logaritma dobili vrijednost našeg logaritma s većom preciznošću. Promatrali smo kako se čestica ponaša, a ako postane prevelika (veća ili jednaka bazi logaritma), dijelimo ju bazom kako bi njena vrijednost opet bila veća od 1 a strogo manja od baze logaritma, te u eksponent kod potencije gdje je baza jednaka bazi logaritma dodamo jedinicu. Time nam je čitava vrijednost logaritma aproksimirana na vrijednost jednaku eksponentu te potencije u argumentu logaritma gdje je baza jednaka bazi logaritma, puta razlomak od jedne polovine ispred čitavog logaritma.

Možemo postaviti iduće pitanje: može li se taj postupak ponoviti, pa da opet udvostručimo preciznost naše aproksimacije logaritma, tj. na preciznost od jedne četvrtine? Naravno da možemo budući da vrijedi sljedeće:

$$\log_a b = \frac{1}{2}\log_a b^2 = \frac{1}{4}\log_a b^4 = \frac{1}{8}\log_a b^8 = \frac{1}{16}\log_a b^{16} = \ldots = \frac{1}{2^n}\log_a b^{2^n}$$

Kako bismo izbjegli grananje eksponenta u potenciji u argumentu, stavit ćemo u njega neku cjelobrojnu varijablu m, te samo argumentirati je li se ona morala povećati za jedan u pojedinom koraku ili ne. A kako bismo razlikovali pojedine čestice u pojedinim koracima, pridodat ćemo im odgovarajuće indekse s rednim brojevima koraka, tj.

$$\log_a b = \log_a \left(c_0 a^{m_0}\right) = \frac{1}{2}\log_a \left(c_1 a^{m_1}\right) = \frac{1}{4}\log_a \left(c_2 a^{m_2}\right) = \ldots = \frac{1}{2^n}\log_a \left(c_n a^{m_n}\right),$$

$$1 \le c_n < a, \qquad m_n \in \left\{2m_{n-1}, 2m_{n-1}+1\right\}.$$

Gledajući taj algoritam korak po korak (tj. nakon svakog kvadriranja i pripisivanja ispred logaritma novog faktora od jedne polovine), uvijek biramo hoćemo li dodati našoj aproksimaciji idući razlomak reda veličine 2^{-n}, a to upravo odgovara konceptu decimalnog zapisa brojeva u binarnom brojevnom sustavu.

Konverzija u praktični algoritam

Kod konstrukcije algoritma primijetili smo jednu vrlo povoljnu stvar: uvijek smo kvadrirali argument i ni na koji drugi način ga nismo mijenjali, te smo uvijek gledali isključivo česticu argumenta – je li postala veća ili jednaka bazi našeg logaritma. I to je sve što smo morali raditi. Dva elementa u bilo kojem pismenom postupku gotovo su optimalna, minimalistička razina radnji.

Sada možemo prethodno konstruirani algoritam pokazati na identičan način na jednom konkretnom primjeru logaritma s iracionalnom vrijednošću, te ćemo tada i taj algoritam prikazati s ekvivalentnim algoritmom koji neće sadržavati matematičke koncepte, jednakosti i pravila, već točno onoliko minimalno potrebno u svakom koraku da točno znamo gdje smo, tj. da ga optimiziramo za praktično pismeno računanje.

Odredimo vrijednost logaritma log 500

Vrijednost zadanog logaritma aproksimirat ćemo korak po korak i paralelno pisati njegovu trenutnu aproksimaciju sve do preciznosti od $\frac{1}{16}$.

$\log_{10} 500 = \log_{10}(5\cdot 10^2) \quad \Rightarrow \quad 2 < \log_{10} 500 < 3$

$\frac{1}{2}\log_{10}\left(5\cdot 10^2\right)^2 = \frac{1}{2}\log_{10}\left(25\cdot 10^4\right)$ [$c \geq a, 25 \geq 10$; čestica mora biti manja od baze]

$\frac{1}{2}\log_{10}\left(25\cdot 10^4\right) = \frac{1}{2}\log_{10}\left(2.5\cdot 10^5\right) \quad \Rightarrow \quad \frac{5}{2} < \log_{10} 500 < \frac{6}{2}$

$\frac{1}{4}\log_{10}\left(2.5\cdot 10^5\right)^2 = \frac{1}{4}\log_{10}\left(6.25\cdot 10^{10}\right) \quad \Rightarrow \quad \frac{10}{4} < \log_{10} 500 < \frac{11}{4}$

$\frac{1}{8}\log_{10}\left(6.25\cdot 10^{10}\right)^2 = \frac{1}{8}\log_{10}\left(39.0625\cdot 10^{20}\right)$ [$c \geq a$; čestica prevelika]

$\frac{1}{8}\log_{10}\left(39.0625\cdot 10^{20}\right) = \frac{1}{8}\log_{10}\left(3.90625\cdot 10^{21}\right) \quad \Rightarrow \quad \frac{21}{8} < \log_{10} 500 < \frac{22}{8}$

$\frac{1}{16}\log_{10}\left(3.90625\cdot 10^{21}\right)^2 = \frac{1}{16}\log_{10}\left(15.2587890625\cdot 10^{42}\right)$ [čestica prevelika]

$\frac{1}{16}\log_{10}\left(15.2587890625\cdot 10^{42}\right) = \frac{1}{16}\log_{10}\left(1.52587890625\cdot 10^{43}\right) \quad \Rightarrow$

$\Rightarrow \quad \frac{43}{16} < \log_{10} 500 < \frac{44}{16}.$

Aproksimirali smo vrijednost logaritma log 500 našim algoritmom na vrijednost između $\frac{43}{16}$ i $\frac{44}{16}$, tj. na vrijednost između 2.6875 i 2.75. To je poprilično dobra aproksimacija jer prava vrijednost našeg logaritma s prve četiri decimale iznosi 2.6989.

Optimizacija za pismeni račun

Promotrimo li prethodnu računicu malo bolje, vidjet ćemo da smo svaki put zapravo kvadrirali samo trenutnu vrijednost čestice, a ukoliko bi ona premašila vrijednost baze, podijelili smo je bazom kako bi se vratila na svojstvo čestice. Red veličine preciznosti može se bilježiti korak po korak u decimalnom obliku broja u binarnom brojevnom sustavu. Cjelobrojna vrijednost logaritma dobije se poprilično lako: konstantno dijelimo početni argument bazom tako dugo dok njegova vrijednost ne postane strogo manja od baze. Broj dijeljenja označavat će cjelobrojnu vrijednost logaritma.

Dakle, uzastopnim dijeljenjem argumenta bazom te brojenjem koliko smo puta dijelili argument dobivamo cjelobrojnu vrijednost logaritma, a prethodno provedenim postupkom dobivamo decimalu po decimalu u binarnom zapisu. Kada dođemo do željene preciznosti, s postupkom stanemo te pretvorimo dobiveni binarni zapis decimala u željeni (vjerojatno ćemo preferirati dekadski). Ukratko, otkrili smo kako pismeno logaritmirati.

Kako bismo prikazali optimalni ekvivalentni postupak, ponovo ćemo odrediti vrijednost od log 500.

Prvo odredimo cjelobrojni dio:

$500 : 10 = 50, 50 : 10 = 5 < 10 \Rightarrow$ dva dijeljenja, cjelobrojni je dio 2, a početna je čestica naš ostatak 5. Krenimo na decimale.

$5^2 = 25 \geq 10 \Rightarrow$ zapisujemo binarnu decimalu 1, tj. $2.(1)_{(2)}$, a česticu dijelimo bazom jer je prevelika.

$25 : 10 = 2.5$

$2.5^2 = 6.25 < 10 \Rightarrow$ zapisujemo binarnu decimalu 0, tj. $2.(10)_{(2)}$, a česticu ne diramo jer je još uvijek manja od baze.

$6.25^2 = 39.0625 \geq 10 \Rightarrow 2.(101)_{(2)}$

$39.0625 : 10 = 3.90625$

$3.90625^2 = 15.2587890625 \geq 10 \Rightarrow 2.(1011)_{(2)}$

$15.2587890625 : 10 = 1.52587890625$

$1.52587890625^2 \approx 2.32830643654 < 10 \Rightarrow 2.(10110)_{(2)}$

$2.3280643654^2 \approx 5.42101086243 < 10 \Rightarrow\ 2.(101100)_{(2)}$

$5.42101086243^2 \approx 29.3873587706 \geq 10 \Rightarrow\ 2.(1011001)_{(2)}$

$29.3873587706 : 10 = 2.93873587706$

$2.93873587706^2 \approx 8.63616855512 < 10 \Rightarrow\ 2.(10110010)_{(2)}$

$8.63616855512^2 \approx 74.5834073124 \geq 10 \Rightarrow\ 2.(101100101)_{(2)}$

Možemo stati u kojem god trenu želimo; vidimo da se računica postupno otežava zbog sve većeg broja decimala ako bismo htjeli ići unedogled. No, za očekivane potrebe neće nam nikad trebati više od 6 dekadskih decimala vrijednosti logaritma, tako da i s aproksimacijom od 11 decimala na svakom koraku (kao što smo ovdje radili) nećemo početi pogrešno aproksimirati dokle god nam koraci ne odu do vrlo velikog broja, a mi ćemo obično stati već nakon najviše 10-ak koraka.

Zadnje što moramo napraviti jest, naravno, pretvoriti dobiveni broj u potpunosti u dekadski. To činimo na standardni način:

$$2.(101100101)_{(2)} = 2 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-7} + 2^{-9} = 2 + \frac{1}{2} + \frac{1}{8} + \frac{1}{16} + \frac{1}{128} + \frac{1}{512} =$$

$$2.697265625$$

U retrospekciji, pogreška iznosi samo $2.69897 - 2.69726 = 0.0017$, što je vrlo prihvatljivo nakon relativno kratkog računa aproksimacije.

Logaritmi s racionalnim rješenjima

Isti se postupak može provesti, naravno, nad bilo kojim logaritmom standardnog oblika koji je objašnjen na početku ovog članka.

Neki od tih logaritama imat će, dakako, i racionalna rješenja, a za njih nema smisla unedogled računati redoslijed decimala. Ono što razlikuje racionalne brojeve od iracionalnih je periodični ili parcijalno periodični zapis decimala ili postojanje posljednje decimale (što je opet parcijalna periodičnost, gdje periodu čine sve nule nakon posljednje neperiodične znamenke). Bez obzira bile decimale u nama dobro poznatom i istreniranom decimalnom obliku ili u bilo kojoj drugoj bazi (pa tako i binarnoj), periodičnost znamenaka znači racionalnost.

Uzmimo tako, primjerice, $\log_9 2187$. Njegova vrijednost iznosi 3.5. Krenimo s našim postupkom pismenog logaritmiranja te pokušajmo ustvrditi u kojem trenu valja stati:

$2187 : 9 = 243, 243 : 9 = 27, 27 : 9 = 3 < 9 \Rightarrow$ tri dijeljenja, cijeli dio rješenja iznosi 3.

$3^2 = 9 \geq 9 \Rightarrow\ 3.(1)_{(2)}$

$9 : 9 = 1 \Rightarrow$ čestica nam je postala jednaka jedinici, što znači da kvadriranje na nju

više nema utjecaj, a to pak znači da će ostatak našeg decimalnog zapisa biti sve nule. Dakle, postupak je ovdje gotov, a rješenje egzaktno:

$$3.(1)_{(2)} = 3.5$$

Druga je mogućnost da zapis bude periodički. Uzmimo primjer $\log_{32} 512$.

$512 : 32 = 16 < 32 \quad \Rightarrow$ jedno dijeljenje, cijeli dio je 1.

$16^2 = 256 \geq 32 \quad \Rightarrow \quad 1.(1)_{(2)}$ [zapamtimo da nam je ovdje $c_0 = 16$]

$256 : 32 = 8$

$8^2 = 64 \geq 32 \quad \Rightarrow \quad 1.(11)_{(2)}$

$64 : 32 = 2$

$2^2 = 4 < 32 \quad \Rightarrow \quad 1.(110)_{(2)}$

$4^2 = c_4 = 16 = c_0 < 32 \quad \Rightarrow \quad 1.(1100)_{(2)}$

Čestica c_4 jednaka je čestici c_0. Budući da je su one jednake, otkrile su nam periodu. Dakle, čestica c_4 imat će potpuno jednak utjecaj na daljnje znamenke kao što je imala i čestica c_0, te odmah znamo da će nam i čestice c_8, c_{12}, c_{16} ... sve biti jednake 16 jer nam one čine periodu u binarnom zapisu, a ta perioda kreće odmah od prve decimale te glasi onome što smo dobili gore: 1100. Uz malo infinitezimalnog računa (za pretvaranje iz binarnih decimala u dekadsku), tj. izračunavanje iznosa

$$\sum_{k=1}^{\infty} (2^{-(4k-3)} + 2^{-(4k-2)}),$$

dobit ćemo da će nam vrijednost ispasti $\frac{4}{5}$, tj. 0.8. Dakle, $\log_{32} 512 = 1.8$.

Mogli smo uočiti da računanje racionalnih vrijednosti ovim algoritmom nije najpraktičnije pa ćemo se radije držati klasičnih, školskih metoda sređivanja izraza da lakše izvučemo van pravu vrijednost logaritma. No, ukoliko nismo sigurni kakva će nam vrijednost ispasti (racionalna ili iracionalna), definitivno trebamo ići ovim algoritmom. Ako ustvrdimo da postoji perioda, možda je najbolje da u potpunosti odbacimo ovaj algoritam i krenemo „školski" sređivati zadani logaritam. No, ako znamo da nam logaritam nije racionalne vrijednosti, tada je ovaj algoritam najbolji izbor ukoliko se odlučimo poigrati njime bez ikakve pomoći računala, Taylorovog reda ili logaritamskih tablica.

Zaključak

Postoje još mnoge relativno jednostavne, elegantne stvari u matematici koje mogu poslužiti kao primjer da se nešto izuzetno kompleksno (poput aproksimacije logaritama s iracionalnim rješenjem) može dokučiti samo pomoću osnovnih računskih operacija i nekih od najosnovnijih svojstava pripadnih koncepata – u ovome slučaju logaritama. Najbitniji su ideja i eksperimentiranje, a dalje sve ovisi o znanju dolaženja do dokaza ili kontradikcije.