Oszd meg és uralkodj

Hajnal Péter

Bolyai Intézet, TTIK, SZTE, Szeged

2021. ősz

Az alapötlet

• Az „oszd meg és uralkodj” algoritmus tervezési séma a matematikai rekurzió egy algoritmuselméleti megvalósítása. A két séma alapelve azonban lényegesen különböző, más-más helyzetben alkalmazhatók.

• Az oszd meg és uralkodj alapú algoritmusok az inputot két közel egyenlő nagyságú részre osztják (esetleg több/k részre melyek méretei az eredeti méret legfeljebb α-szorosai, ahol $\alpha < 1$ rögzített paraméter).

• A részekre is értelmes a feladat. A két „fél-feladatra” megoldjuk a problémát és a két fél megoldásából kiszámítjuk az eredeti kérdésre adandó választ.

• Ismét példákon keresztül érthetjük meg a módszert.

Sorting

Rendezés (angolul sorting): Adottak $x_1, x_2, \ldots, x_n$ számok. Rendezzük sorba ezeket. Azaz keressünk egy π permutációját az $[n] = \{1, 2, \ldots, n\}$ indexhalmaznak, amelyre $x_{\pi 1} \leq x_{\pi 2} \leq \ldots \leq x_{\pi n}$.

Összehasonlítás alapú algoritmusok

• Az input számaira gondolhatunk mint egy számjegysorozat.

• Mi azonban most egy egyszerűsített, de természetes feltevéssel élünk: Számainkról információt csak egy-egy összehasonlításával szerezhetünk.

• Az első összehasonlítás után ennek eredménye függvényében választjuk meg következő összehasonlításunkat és így tovább, míg elegendő informaációnk lesz a rendezett sorrend bejelentéséhez.

• Az algoritmus költsége egy adott inputon az elvégzett összehasonlítások száma.

Oszd meg

- Ha $n = 2$, akkor egy összehasonlítással megoldható a probléma.

- Ha nagyobb inputtal dolgozunk akkor vágjuk két körülbelül egyforma nagyságú számsorra az inputot, rendezzük a két fél inputot, majd a két rendezett félből számoljuk ki a teljes rendezett sort, azaz az outputot.

Az összefésülés

• Az oszd meg és uralkodj elv lényegi része, hogy két rendezett sorozatot hogy lehet „összefésülni”.

• A probléma könnyen megoldható. Legyen adott két rendezett számsorozat (egy k és egy ℓ hosszú).

• Az alapötlet, hogy a két legkisebb szám összehasonlítása után tudni fogjuk a teljes rendezett sorozat legkisebb elemét. „Egy elemet a helyére raktunk.” A maradék/helyére nem tett $k + \ell - 1$ szám is két rendezett sorban van.

• Iterálhatjuk az ötletet, egy összehasonlítás árán egy szám újra a helyére kerül.

• Legfeljebb $k + \ell - 1$ összehasonlítás után lesz $k + \ell - 1$ számunk, amely helyét (legkisebb, második legkisebb, . . . meghatároztuk. Ezzel a teljes rendezett sorunk meglesz.

Összefésülés: egy megjegyzés

Megjegyezzük, hogy esetleg kevesebb összehasonlítás is elvezethet a végeredményhez. Ha az inputban a rendezett k számunk a teljes számhalmaz k legkisebb eleme, akkor k összehasonlítással kiderítjük ezt.

Az analízis kezdete

- Legyen $S(n)$ az az összehasonlítás szám, ami minden szám n-es esetén elegendő a teljes rendezett sorrend kialakításához.

- Tudjuk, hogy $S(1) = 0$ és $S(2) = 1$.

Az analízis folytatása

• Az összefésülés összehasonlításainak száma (másképpen az összefésülés költsége) az $S(n)$-es teljes költségben felülről becsülhető $n-1$-gyel.

• Maradt a két „fél” rendezésének költsége, ami a majdnem tökéletes félbevágás esetén legfeljebb

$$S\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + S\left(\left\lceil \frac{n}{2} \right\rceil\right).$$

• Összefoglalva: $n \geq 3$ esetén

$$S(n) \leq S\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + S\left(\left\lceil \frac{n}{2} \right\rceil\right) + n - 1.$$

Az analízis vége

- Ebből teljes indukcióval könnyen igazolható a következő állítás.

Lemma

$$S(n) \leq n \log_2 n.$$

- Gyakorlaton láttuk, hogy ebben a számítási modellben ez az optimális nagyságrend.
- Tehát a jó séma szerint tervezve algoritmusunkat „uralkodunk”.

Szünet

Az alapkérdés

Szorzás: Adott két n-jegyű szám $x_1x_2 \dots x_n$ és $y_1y_2 \dots y_n$ (azaz az x_i és y_i összetevői az inputnak számjegyek). Számoljuk ki a szorzatukat.

Miért?

- Miért pont a szorzást vizsgáljuk?
- Az összeadás, kivonás egyszerűbb művelet. Ezek azonban nagyon jól számolhatók az általános iskolában megtanult módszerekkel.
- Két n-jegyű szám összeadása/kivonása $\mathcal{O}(n)$ számjegy művelettel kiszámítható a klasszikus módszerekkel.
- Nyilván minden számjegy elolvasása szükséges a végeredmény bejelentéséhez. A nagyságrend optimális.
- Hasonló a helyzet a kettővel való szorzás, illetve a kettővel való osztás esetén.
- Az általános iskolai, naív algoritmus $\mathcal{O}(n)$ számjegy művelettel kiszámolja a végeredményt és csak konstans faktorral való javítás lehetséges.

A probléma

• Két n-jegyű szám szorzására megtanított algoritmus $\mathcal{O}(n^2)$ számjegy művelettel számítható ki.

• Az egyik input szám minden számjegyét a másik input szám minden számjegyével megszorozzuk, majd összegzéssel jön ki a szorzat. A szükséges számjegy műveletek száma négyzetes lesz. Ebben van javítási lehetőség.

• Az 1960-as évek elején Kolmogorov (az évszázad rendkívül jelentős hatású matematikusa) szemináriumán sejtette, hogy a több évszázados algoritmus nem javítható.

• Egy munkatársa, Karatsuba cáfolta meg ezt a sejtést. Karatsuba egy algoritmust adott, amely hatékonyabb a szokásos szorzási algoritmusnál. Az alábbiakban ezt ismertetjük.

A naív „osztás”

• Az oszd meg és uralkodj elve azt mondja, hogy írjuk fel az x, illetve y számot mint

$$x_{eleje} \cdot 10^{\nu} + x_{vége}, \qquad \textit{illetve} \qquad y_{eleje} \cdot 10^{\nu} + y_{vége}.$$

• x és y szorzata

$$x_{eleje}y_{eleje} \cdot 10^{2\nu} + (x_{vége}y_{eleje} + (x_{eleje}y_{vége}) \cdot 10^{\nu} + x_{vége}y_{vége}.$$

• Ez jó félbevágás esetén négy feleakkora (az inputhoz képest, a számjegyek számában mérünk) számpár szorzatával számolható ki.

• Fontos látni, hogy 10^{ν} és $10^{2\nu}$ tényezőkkel való szorzás csupán nullákkal bővíti a szorzott számot. Számolási igénye nincs.

Az analízis eleje

• Az összefésülő rendezéshez hasonlóan meghatározható, hány számjegy művelet szükséges ha fenti formulán alapuló, oszd meg és uralkodj elven tervezett algoritmussal dolgozunk.

• Bevezethető az $M(n)$ függvény, ami az algoritmus maximális számjegy-művelet igényét jelöli az n-jegyű számok között (ez a minimális műveletszám, amit két n-jegyű szám szorzásakor a számok ismerete nélkül garantálhatunk algoritmusunk futásának szükségletére).

Az analízis vége

- Felírhatjuk, hogy

$$M(n) \leq 4 \cdot M\left(\frac{n}{2}\right) + \mathcal{O}(n),$$

ahol a $\mathcal{O}$ jelölés csak a lustaság miatt szerepel.

- Az érdeklődő hallgató egy pontos konstanst kiszámolhat, vagy némi gondolkozás után meggyőződhet, hogy $100 \cdot n$-et írva igaz állítást kap.

Csalódás

• A becslés „kifejtése” után sajnos kiderül, hogy algoritmusunk nem lesz hatékonyabb mint a naív szorzás, $\mathcal{O}(n^2)$ számjegy-művelet szükséges hozzá. Egyelőre nem „uralkodunk”.

Az első ötlet

• Karatsuba fontos észrevétele, hogy egy n-jegyű szám négyzetének kiszámolása lényegében ugyanolyan bonyolultságú mint a szorzás.

• Valóban a

$$x \cdot y = \frac{(x+y)^2 - x^2 - y^2}{2}$$

képlet alapján három négyzetreemelés és lineáris számjegyműveleteket igénylő „mellékszámolások" kiadják a szorzatot.

• Tehát ha a négyzetreemelést hatékonyabban el tudjuk végezni mint négyzetes sok számjegyművelet, akkor a szorzást is.

• A továbbiakban a négyzetreemelést vizsgáljuk. Legyen $N(n)$ az alábbiakban megtervezett algoritmus számjegy-művelet igénye n-jegyű számok esetén.

Az első ötlet folytatása

- A négyetreemelésnél is alkalmazható az oszd meg és uralkodj elv. Lássuk.
- Az

$$(x_{eleje} \cdot 10^{\nu} + x_{vége})^2 = x_{eleje}^2 \cdot 10^{2\nu} + x_{vége}^2 + 2x_{eleje}x_{vége} \cdot 10^{\nu}$$

képlete „ígéretesebb”: két négyzetreemelés és egy szorzás szükséges, de feleakkora számokon!

A második ötlet

• A kétféle művelet keveredésére (szorzás, négyzetreemelés) kiküszöbölésére már megvan az ötletünk, áttérhetünk egy csak négyzetreemelést használó képletre:

$$(x_{eleje} \cdot 10^{\nu} + x_{vége})^2 = x_{eleje}^2 \cdot 10^{2\nu} + x_{vége}^2 + ((x_{eleje} + x_{vége})^2 - x_{eleje}^2 + x_{vége}^2) \cdot 10^{\nu}$$

Az analízis

- Ha ezt a képletet alkalmazzuk, akkor három négyzetreemelés és egyszerű mellékszámolások kellenek.
- Kapjuk, hogy

$$N(n) \leq 3N\left(\frac{n}{2}\right) + \mathcal{O}(n).$$

- Ennek megoldása

$$N(2^k) = \mathcal{O}(3^k).$$

Intuitíven felezzük az inputot és ennek ára háromszorozodás. 2^k esetén k felezés van, a költségoldalon k darab háromszorozódás történik.

- A pontos bizonyítás teljes indukció lehet.
- Ebből adódik, hogy

$$N(n) = \mathcal{O}(n^{\log_2 3}).$$

- Az általános iskolás szorzási algoritmus nem optimális.

Végső megjegyzés

• Megjegyezzük, hogy Karatsuba algoritmus távol van az optimálistól.

• Az előadás idején van bejelentett $\mathcal{O}(n \log n)$-es algoritmus. Ez még nem jelent meg referált folyóiratban.

Vége van!

Köszönöm a figyelmet!