Összefésüléses rendezés

Csima Judit
BME SZIT
csima@cs.bme.hu

2019. október 1.

Összefésüléses rendezés

A múlt órán láttuk a bináris keresést, mely azon az oszd meg és uralkodj nevű elven alapult, hogy úgy oldunk meg egy feladatot (a bináris keresés esetén az s érték megkeresését az $A[0 : n-1]$ tömbben, hogy visszavezetjük a feladatot egy legfeljebb fele akkora méretű, hasonló kérdésre: egy összehasonlítás után vagy a tömb első vagy a második felében folytatjuk az eljárást (amennyiben az összehasonlítással nem találtuk meg s-et). Ez a stratégia azért vezet gyorsan eredményre, mert minden lépésben vagy megtaláljuk az elemet vagy feleződik a tömb mérete vagyis gyorsan elérünk az alapesethez, amikor egy üres tömbben keresünk valamit (és persze nem találjuk).

Ez az oszd meg és uralkodj elv fog megjelenni a most tárgyalandó új rendezőalgoritmusban, az összefésüléses rendezésben.

Eddig tanult rendezőalgoritmusok összefoglalása

A félév során eddig három rendezőalgoritmust tanultunk:

- Kiválasztásos rendezés: ennek lépésszámáról láttuk, hogy $O(n^2)$ egy n méretű tömbön, sőt az is igaz (ezt nem láttuk, de meggondolható), hogy az ismételt minimumkiválasztások közben összesen $n-1+n-2+\ldots+2+1 = \frac{n(n-1)}{2} \geq 1/4n^2$ összehasonlítás történik bármelyik n méretű inputon futtatjuk az eljárást vagyis ennek az algoritmusnak a lépésszáma nagyságrendileg n^2-es.

- Buborékrendezés: ennek lépésszámáról is beláttuk, hogy $O(n^2)$ egy n méretű tömbön, sőt az is igaz (ezt nem láttuk, de meggondolható), hogy összesen $n-1+n-2+\ldots+2+1 = \frac{n(n-1)}{2} \geq 1/4n^2$ összehasonlítás biztosan történik bármelyik n méretű inputon is futtatjuk az eljárást (mégha szerencsés esetben cseréből jóval kevesebb is van) vagyis ennek az algoritmusnak a lépésszáma is nagyságrendileg n^2-es.

- Beszúrásos rendezés: erről is láttuk, hogy lépésszáma $O(n^2)$ és az is igaz, hogy a fordítva rendezett $n, n-1, \ldots, 2, 1$ tömbön összesen $n-1+n-2+\ldots+2+1 = \frac{n(n-1)}{2} \geq 1/4n^2$ csere biztosan történik vagyis ennek az algoritmusnak a lépésszáma is nagyságrendileg n^2-es.

A most tárgyalásra kerülő rendező algoritmus, az összefésüléses rendezés ennél gyorsabb lesz, azt fogjuk róla megmutatni, hogy lépésszáma $O(n \log n)$.

Az összefésüléses rendezés elve

Az érdekes eset az lesz, ha a rendezendő tömb legalább 2 elemet tartalmaz. Ekkor a következőt fogjuk csinálni:

1. Rendezzük a tömb első felét (majd mindjárt megbeszéljük, hogy hogyan)
2. Rendezzük a tömb második felét (az első feléhez hasonlóan)
3. Az így kapott két rendezett tömbből előállítunk egyetlen rendezett tömböt, ami így már az input összes elemét fogja rendezetten tartalmazni.

Először foglalkozzunk a 3. lépéssel, azaz azzal, hogy hogyan lehet két rendezett tömb elemeiből egyetlen rendezett tömböt létrehozni.

Az összefésülés eljárás elve

Ennek az eljárásnak két inputja van, egy k elemű rendezett $B[0:k-1]$ és egy ℓ elemű rendezett $C[0:\ell-1]$ tömb, az eljárás ezeknek az elemeiből fog előállítani egy $k+\ell$ hosszú $D[0:k+\ell-1]$ tömböt.

Nézzük meg az eljárás működését először egy példán.

Legyenek $B=[1,2,5,8]$ és $C=[3,4,10,12,13]$. Ekkor a kilenc elemű, kezdetben üres D tömböt a következőképpen töltjük fel elölről hátrafelé haladva:

A D tömb első eleme biztosan a B tömb és a C tömb első elemei közül kerül ki,mert minden más elem ezeknél nagyobb, ezért ezeket hasonlítjuk össze és mivel $B[0]=1<C[0]=3$, így $B[0]=1$-et rakjuk $D[0]$-ba.

Mivel B-ből már az első elemet kivettük (úgy gondolunk rá, hogy átléptük, vele már nem kell foglalkozni, ő már a helyére került D-ben), most a B maradék elemei közül a következő legkisebbet (azaz a következő elemet, azaz $B]1]=2$-t) hasonlítom össze C első még nem átlépett elemével, vagyis $C[0]=3$-mal. Mivel $B[1]=2<C[0]=3$, így $B[1]=2$-t rakjuk $D[1]$-ba.

Mivel B-ből már az első két elemet átléptük, most a B maradék elemei közül következő legkisebbet $B[2]=5$-öt hasonlítom össze C első még nem átlépett elemével, vagyis $C[0]=3$-mal. Mivel $B[2]=5>C[0]=3$, így $C[0]=3$-at rakjuk D következő üres cellájába, azaz $D[2]$-be.

Az eljárás elve tehát a következő lesz: nyilvántartom minden lépésben, hogy kik a legkisebb, még nem átlépett elemek a B és C tömbökben és ezen kettő közül a kisebbet teszem be D következő cellájába. A fenti példán az előző három lépés után a D tömbben már benne van $1,2,3$ és ezután a $5>4$ összehasonlítás miatt bekerül a 4, majd az $5<10$ összehasonlítás miatt bekerül az 5, majd a $8<10$ összehasonlítás miatt bekerül a 8.

Ekkor az a helyzet áll elő, hogy a B tömb minden elemét D-be raktuk már, így további összehasonlítások nélkül C maradék elemeit sorban D-be tesszük.

Az összefésülés eljárás pszeudokódja

Az eljárás inputjai a k elemű rendezett $B[0:k-1]$ és az ℓ elemű rendezett $C[0:\ell-1]$ tömb, outputja pedig az ezeknek az elemeiből álló $k+\ell$ hosszú $D[0:k+\ell-1]$ tömb. Tegyük fel, hogy a két tömbben szereplő elemek mind különbözőek.

```
D:= (k+l) hosszú üres tömb
i: = 0     // ez mutatja, hogy hol tartunk a B tömbben
j: = 0     // ez mutatja, hogy hol tartunk a C tömbben
t: = 0     // ez mutatja, hogy hol tartunk a D tömbben

ciklus amíg (i < k és j < l):     // még van elem mindkét tömbben
    ha B[i] < C[j]:        // B-ból jön a kisebb érték
        D[t]: = B[i]
        i := i+1
        t := t+1
    egyébként:                 // C-ból jön a kisebb érték
        D[t] := C[j]
        j := j+1
        t := t+1
    elágazás vége
ciklus vége

ha i == k:                       // a B tömb fogyott el
    ciklus amíg j < l:     // amíg van elem C-ben
        D[t] := C[j]
        j := j+1
        t := t+1
    ciklus vége
egyébként:                //  a C tömb fogyott el
    ciklus amíg i < k:  // amíg van elem B-ben
        D[t]: = B[i]
        i := i+1
        t := t+1
    ciklus vége
elágazás vége

return D
```

Az összefésülés eljárás helyessége és lépésszáma

Az eljárás helyessége abból következik, hogy minden lépésben igazak a következők:

- a D tömb mindig rendezetten tartalmazza az elemeket
- a D tömb minden eleme kisebb, mint bármelyik B-ben vagy C-ben levő, még nem átlépett elem
- a még fel nem dolgozott elemek közül a legkisebb vagy a $B[i]$ vagy a $C[j]$ elem lehet csak (a B és C tömbök rendezettsége miatt), így ezen két elem összehasonlításával mindig a még nem kiírt legkisebb elem kerül helyesen a D tömb következő cellájába.

Az eljárás végén az első pont miatt D rendezett lesz és ekkor már minden elemet tartalmaz.

Az eljárás lépésszámát a pszeudokód megvizsgálásával tudjuk megállapítani. Az elején van konstans sok lépés, azután jön egy ciklus, majd egy elágazás, ezeknek a lépésszámát külön becsüljük meg.

A ciklus magja olyan, hogy minden egyes lefutáskor vagy i vagy j értéke nő és biztosan leáll a ciklus, amikor ezek elérik a tömb végét, i a k-t, j pedig ℓ-et. Ez azt jelenti, hogy a ciklus magja legfeljebb $(k + \ell)$-szer fut le, egy lefutás pedig konstans sok lépésből áll, vagyis a ciklus lépésszáma $O(k + \ell)$.

Az ezt követő elágazás során az első ágon legfeljebb k-szor fut le a ciklus magja, vagyis legfeljebb $(k+\ell)$-szer, a második ágon legfeljebb ℓ-szer fut le a ciklus magja, vagyis legfeljebb $(k+\ell)$-szer, mindkét esetben a mag egy lefutása konstans lépés vagyis az egész elágazás $O(k + \ell)$ lépés.

Vegyük észre, hogyha (mint az összefésüléses rendezésben) egy n méretű tömböt két részre szedünk, a két részt rendezzük és a kapott rendezett tömböket összefésüljük, akkor (mivel a két tömb hosszának összege n) ez az összefésülés $O(n)$ lépést használ.

Azaz a teljes algoritmus futása konstans + $O(k + \ell)$ + $O(k + \ell)$, azaz a következő feladatban megfogalmazott számolási szabályok szerint $O(k + \ell)$.

Feladat Lássa be, hogy ha egy algoritmus lépésszáma konstans + $O(k+\ell)$, akkor az algoritmus lépésszáma $O(k + \ell)$.

Feladat Lássa be, hogy ha egy algoritmus lépésszáma $O(k+\ell)$ + $O(k+\ell)$, akkor az algoritmus lépésszáma $O(k + \ell)$.

Az összefésüléses rendezés

Az összefésüléses rendezés egy n elemű tömbön úgy működik, hogy $n = 1$ esetén magát az inputot adja vissza, hiszen az egy elemű tömb rendezett. Amennyiben $n \geq 2$, akkor pedig úgy fut az n elemű tömbön, hogy először összefésüléses rendezéssel (vagyis ugyanezzel az algoritmussal) rendezi a tömb első és második felét, majd ezeket a már rendezett fél-tömböket a fenti összefésülés eljárással fésüli össze egy rendezett tömbbe.

Az ÖR($A[0 : n - 1]$) (ÖR az összefésüléses rendezés rövidítése ebben a jegyzetben) eljárás kissé pongyola pszeudokódja tehát a következő:

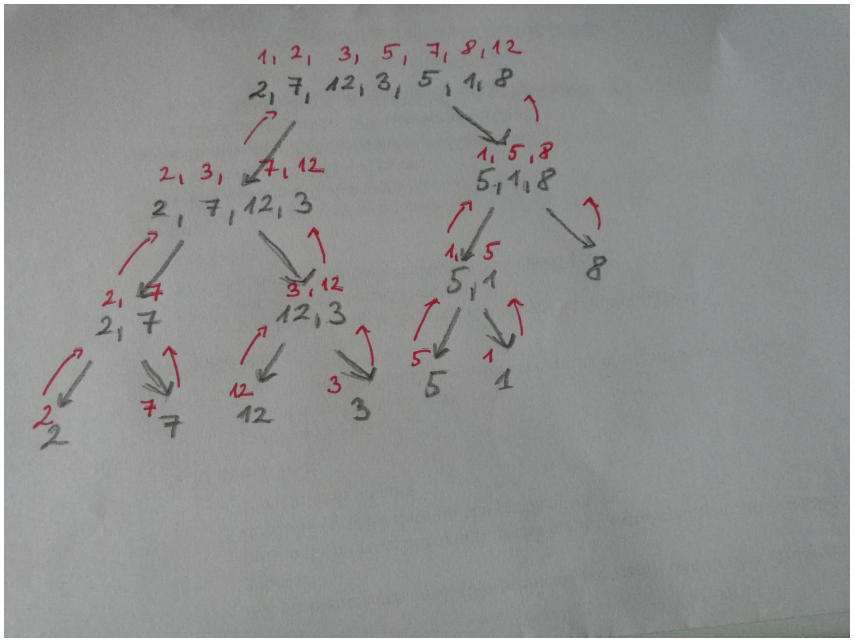
```
Ha n == 1:
    return A
egyébként:
    B: = ÖR(az input tömb első fele)
    C: = ÖR(az input tömb második fele)
    D: = B és C összefésülése a fenti összefésül eljárással
    return D
```

Nézzük meg egy példán, hogyan fut le az összefésüléses rendezés. Legyen az input tömb $2, 7, 12, 3, 5, 1, 8$ és állapodjunk meg abban, hogy ha páratlan hosszú tömböt kell két részre osztani, akkor az első rész lesz a hosszabb.

```
ÖR(2,7,12,3,5,1,8):

    ÖR(2,7,12,3):

        ÖR(2,7):
            ÖR(2) -> 2
            ÖR(7) -> 7
            összefésülés eljárás 2-re és 7-re -> 2,7
        ÖR(2,7) vége, outputja 2,7

        ÖR(12,3):
            ÖR(12) -> 12
            ÖR(3) -> 3
            összefésül eljárás 12-re és 3-ra -> 3,12
        ÖR(12,3) vége, otputja 3,12

        összefésül eljárás 2,7 és 3,12-re, outputja 2,3,7,12

    ÖR(2,7,12,3) vége, outputja 2,3,7,12

    ÖR(5,1,8):

        ÖR(5,1):
            ÖR(5) -> 5
            ÖR(1) -> 1
            összefésül eljárás 5-re és 1-re, outputja 1,5
        ÖR(5,1) vége

        ÖR(8) -> 8

        összefésül eljárás 1,5-re és 8-ra, outputja 1,5,8

    ÖR(5,1,8) vége, outputja 1,5,8

    össszefésül eljárás 2,3,7,12-re és 1,5,8-ra, outputja 1,2,3,5,7,8,12

ÖR(2,7,12,3,5,1,8) vége, output 1,2,3,5,7,8,12
```

Ilyen részletesen nem fogjuk többször végigkövetni az összefésüléses rendezés algoritmusát, hanem az alábbi ábrán látható módon fogjuk a futást ábrázolni.

A feketével írt tömbökön futtatjuk az ÖR eljárást, ami a pirossal írt tömböket adja vissza, a felfelé mutató nyilak pedig két rendezett tömb összefésülését jelzik.



Az összefésüléses rendezés helyessége és lépésszáma

Az összefésüléses rendezés helyessége n-re vonatkozó indukcióval látható be: Egyrészt $n = 1$-re helyes, mert az input tömböt adja vissza, ami rendezett. Másrészt ha $n \geq 2$, akkor a két féltömb rendezése az indukciós feltevés miatt helyes (hiszen mindkét fél hossza kisebb, mint n), az összefésül eljárásról meg már láttuk, hogy két rendezett tömbből előállítja az elemeiket tartalmazó egyetlen rendezett tömböt.

A lépésszámot $n = 2^k$, azaz 2-hatvány méretű inputra látjuk csak be, az általános eset ehhez hasonló lenne.

A fenti ábrán látható módon ábrázolva az összefésüléses rendezés futását az $n = 2^k$ méretű inputon látjuk, hogy az első szinten egy darab $n = 2^k$ méretű tömbbel dolgozunk, a 2. szinten kettő darab $n/2 = 2^{k-1}$ méretűvel, a 3. szinten négy darab $n/4 = 2^{k-2}$ méretűvel, stb., az utolsón pedig 2^k darab 1 méretű tömbbel. Számoljuk össze szintenként, hogy hány lépésre van szükség az adott szinten található rendezett tömbök előállításához az eggyel lejjebb levő szinteken található résztömbök összefésülése során (érdemi lépések csak az összefésülések során történnek):

- a legfelső szinten 2 darab féltömb összefésülée $O(n)$ lépést igényel, mert a két féltömb együttes hossza n.
- a 2. szinten 2-2 negyedtömböt fésülünk össze két fél-tömbbé, az érintett tömbök összhossza n, így ezen a szinten is $O(n)$ lépésre van szükség
- hasonlóan meggondolható, hogy minden szinten $O(n)$ lépésre van szükségünk
- mivel láttuk, hogy $k = \log n$ szint van, az összes munka $O(n \log n)$.