



28.

Rendezési algoritmusok

A rendezési algoritmusok használhatósága nagy mértékben függ a halmazok rendezettségétől. Ehhez nem árt tisztában lenni a *ZV_09tetel*-ben található rendezési relációkkal.

Rendezési feladat: egy n számból (általánosabban: adatból) álló (a_1, a_2, \dots, a_n) sorozat elemeinek egy olyan $\langle a'_1, a'_2, \dots, a'_n \rangle$ permutációjának előállítását, amelyre $a'_1 \leq a'_2 \leq \dots \leq a'_n$ (\leq jel itt csupán a rendezési reláció irányát jelöli, nem azonos a kisebb-egyenlő jellel !)

Rekord: a rendezendő adatokat tartalmazó összetett adategyüttes; általában n elemű tömbként vagy listaként ábrázolják.

A RENDEZÉSI ALGORITMUSOK CSOPORTOSÍTÁSA

Helyben rendező algoritmusok: A bemeneti adatokon kívül csak állandó számú változóra van szükség a rendezéshez.

- cserélő („buborék”) rendezés
- Shell rendezés
- minimum-/maximumkiválasztásos rendezés
- beszűrő rendezés
- kupacrendezés
- gyorsrendezés

Összehasonlító rendezések: az elemek sorrendjét azok összehasonlításával állapítja meg. Modellje egy *döntési fával* ábrázolható, ami egy adott méretű tömb rendezése során történt összehasonlításokat szemlélteti. A *döntési fa magassága* (a gyökerét valamely levéllel összekötő leghosszabb út hosszúsága) megadja a rendezési algoritmus legrosszabb esetben elvégzett összehasonlításainak számát. Egy n elemet rendező döntési fa magassága $\Omega(n \lg n)$.

- az itt tárgyalt helyben rendező algoritmusok
- összefésülő rendezés

Lineáris időben rendező algoritmusok: a rendezendő n elemet képesek $O(n)$ idő alatt rendezni

- leszámpláló rendezés
- számjegyes rendezés
- edényrendezés

Cserélő (buborék) rendezés: párosával összehasonlítja az elemeket, s ha nem a megfelelő sorrendben követik egymást, akkor felcserélődnek. Addig ismétlődik az eljárás, amíg további cserére nincs szükség. Nevét onnan kapta, hogy az elemek, mint a buborékok a helyükre bugyognak.

Időigénye: $\Theta(n^2)$.

Shell rendezés (rendezés fogyó növekménnyel): A buborékrendezés cserélési fázisát gyorsítja meg azzal, hogy egymástól távol lévő elemeket cserél fel, s a cserélendő elemek közti távolságot fokozatosan csökkenti, egészen 1-ig. Mire a növekmény 1 lesz, az elemek a helyükre kerültek a beszűrő rendezés algoritmusának alapján.

Időigénye: $O(n^{1.2})$.

Minimum-/maximumkiválasztásos rendezés: először a legkisebb/legnagyobb elemet határozzuk meg, majd a következő legkisebbet/legnagyobbat s így tovább. Tehát az aktuális első elemet a mögötte lévők közül csak a legkisebbel/legnagyobbval cseréljük ki.

Időigénye: $\Theta(n^2)$.

Beszűrő rendezés: a kártyalapok rendezéséhez hasonló az algoritmus. Üres bal kézzel kezdünk, a lapok színükkel lefelé az asztalon vannak. Egy lapot felvesszünk az asztalról és elhelyezzük a bal kezünkben a megfelelő helyen. A hely megtalálásához összehasonlítjuk a felvett lapot a kezünkben lévőkkel egyesével sorban jobbról balra. Hatékony kisszámú elem rendezésére.

Időigénye: $O(n^2)$.

Összefésülő rendezés: „Oszd meg és uralkodj!” elv alapján kisebb részekre osztjuk fel a problémát, amit rekurzívan megoldhatjuk.

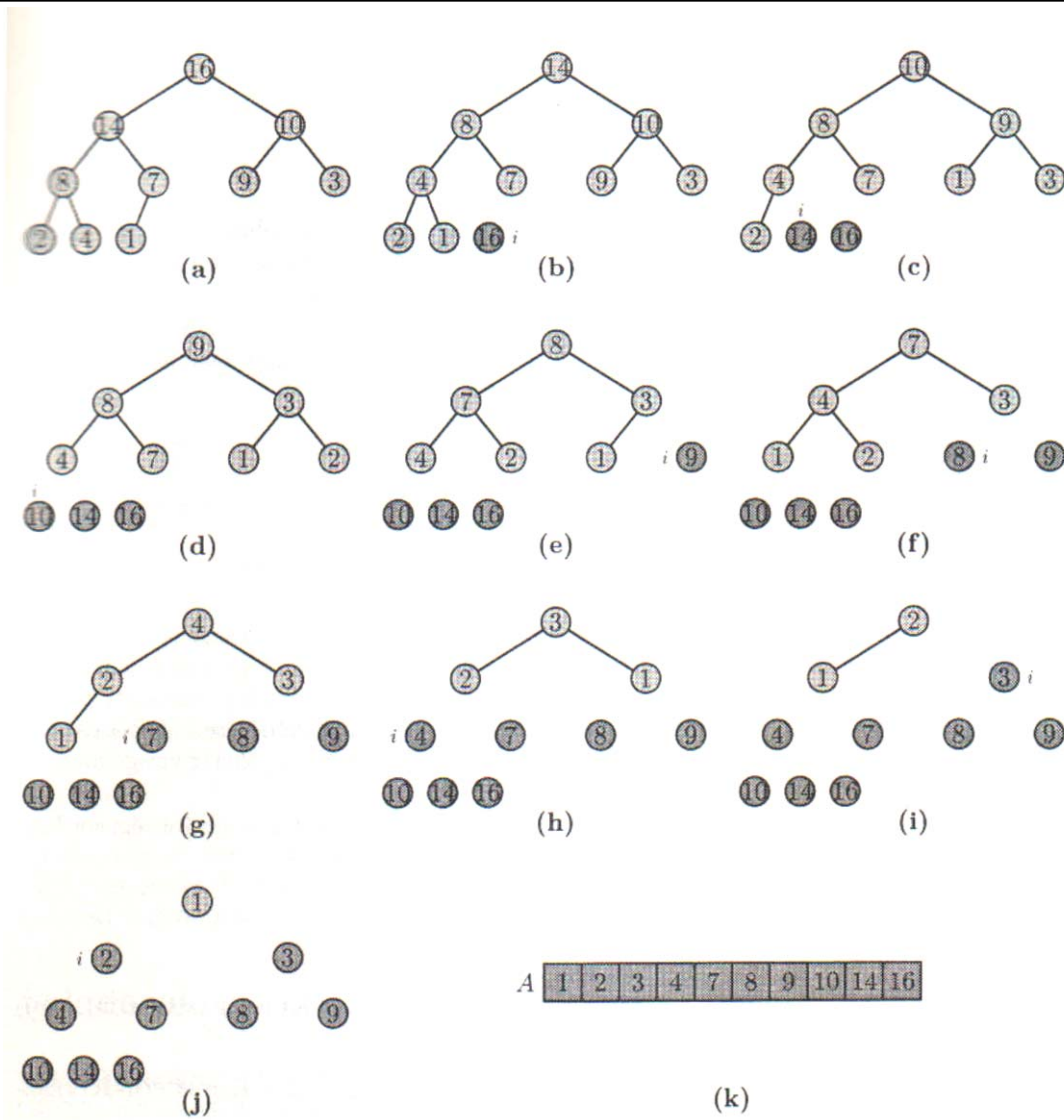
- 1) **felosztás:** a rendezendő tömböt $n/2$ részre osztjuk
- 2) **uralkodás:** rekurzív összefésüléses módon mindkettőt rendezzük (az 1 elemű már rendezettnek tekintendő)
- 3) **egyesítés:** a két rendezett résztömböt összefésüljük.

Időigénye: $O(n \lg n)$.

Kupacrendezés: egyesíti a *beszűrő* és az *összefésülő rendezés* előnyeit, ugyanis gyors futási idővel ($O(n \cdot \lg n)$) helyben rendez. Az algoritmus a kupac adatszerkezetet használja, ám emellett az elsőbbségi sorok rendezéséhez is hatékony. A kupacról mint adatstruktúráról a 25. Záróvizsga tételben olvashatunk. A rendezési algoritmus a következő (szemléltetés az ábrán):

```

KUPACRENDEZES(A)
0   M = kupac-méret[A]
1   KUPACOT_ÉPÍT(A)           // kupaccá alakít egy A[1..n] tömböt; n = hossz[A]
2   for i=hossz[A] downto 2
3       do csere(A[1], A[i]) // mert a max elem a gyökér, és a szülők nagyobbak a gyereknél
4           M=M-1           // az n-dik elemet kizárva a kupacból csökkenti a méretet eggyel
5           KUPACOL(A, 1)   // helyreállítja a kupatulajdonságot az A[1..n-1] tömbön
    
```



7.4. ábra. A KUPACRENDEZÉS működése. (a) A KUPACOT-ÉPÍT eljárás által felépített kupac adatszerkezet. (b)-(j) az adatszerkezet a KUPACOL egy-egy futása után (algoritmus 5. sor). Az ábra mutatja *i* változó pillanatnyi értékét. A kupacban csak a világos körben lévő elemek maradtak. (k) Az eredményül kapott rendezett A tömb.

Gyorsrendezés: szintén az „Oszd meg és uralkodj!” elven alapul; gyakorlatban hasznos, mert jó az átlagos futási ideje; emellett virtuális memória környezetben is jól működik és helyben rendez. Veremfoglalása elég nagy, de kisebb tömböknél nem foglal túl sok memóriát.

- 1) **felosztás:** a rendezendő $A[p..r]$ tömböt két részre osztjuk úgy, hogy $A[p..q]$ minden eleme kisebb legyen, mint $A[q+1..r]$ bármely eleme. Ezt összehasonlítgatással és cserékkel érjük el.
- 2) **uralkodás:** rekurzív módon mindkettőt rendezzük (az 1 elemű már rendezettnek tekintendő)
- 3) **egyesítés:** nincs rá szükség, mert a tömböt a saját helyén rendeztük.

Időigénye (legrosszabb esetben): $\Theta(n^2)$.

Időigénye (átlagos esetben): $\Theta(n \lg n)$ – és a képletben szereplő állandók meglehetősen kicsik

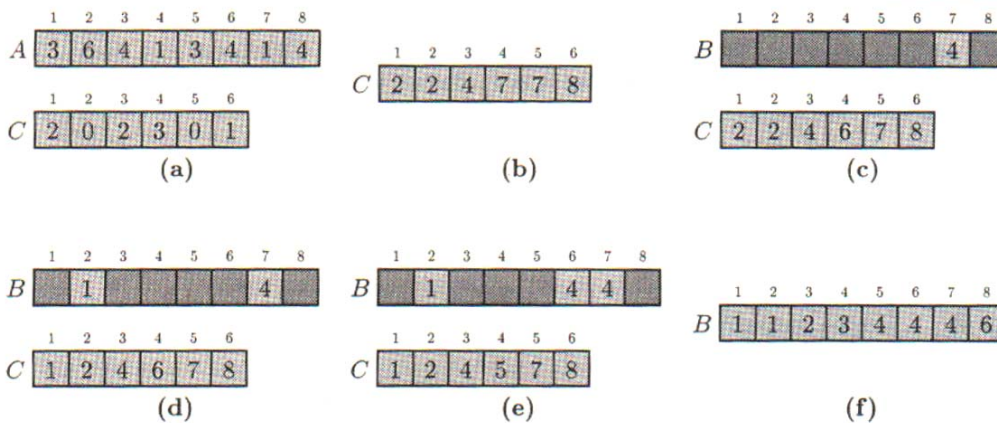
Az alábbiakban összehasonlítgatás helyett más módszerekkel igyekszünk lefaragni az $\Omega(n \lg n)$ alsó korlátból.

Leszámláló rendezés (binsort, ládarendezés): feltételezi, hogy az n bemeneti elem mindegyike 1 és $k \in \mathbf{Z}$ közötti egész szám. Minden elemnél meghatározza a nála kisebb elemek számát (így a vizsgált elem közvetlenül a saját pozíciójába kerülhet a kimeneti tömbben – kivétel, ha a tömb egyforma elemeket is tartalmaz). Egy n hosszúságú A tömbbe rakjuk az elemeket, majd ezeket egy C segéd tömb segítségével sorrendbe rendezzük a B kimeneti tömbbe. *Stabil eljárás*, mert az azonos értékűek sorrendje megegyezik a rendezett tömbben is.

Időigény: $O(k + n)$, de gyakorlatban akkor érdemes ezt használni, ha $k = O(n)$, mert ekkor a futási idő $O(n)$.

9.2. Leszámláló rendezés

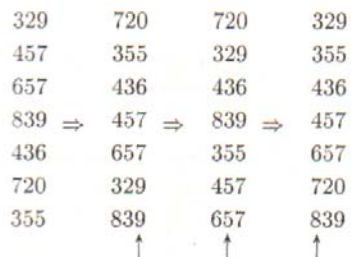
147



9.2. ábra. A LESZÁMLÁLÓ-RENDEZÉS működése egy olyan $A[1..8]$ bemeneti tömbön, amelynek minden eleme olyan pozitív egész, amelyik kisebb vagy egyenlő mint $k = 6$. (a) Az A tömb és a C segéd tömb a 4. sor után. (b) A C tömb a 7. sor után. (c)-(e) A B kimeneti tömb és a C segéd tömb a 9–11. sorokban levő ciklus első, második illetve harmadik iterációs lépése után. A B tömbnek csak a világos elemei kaptak értéket. (f) A végső, rendezett B kimeneti tömb.

Számjegyes rendezés: a leszámláló rendezés kiterjesztése; korábban lyukkártyarendező berendezésekben használták, de alkalmazható azonos d hosszúságú szavak, ugyanannyi d számjegyből álló számok, dátumok esetén is. Az egy karakteren előforduló jegyek, jelek számát jelölje k . Stabil módszer szerint számjegyenként rendezzük a számokat. Elsőként a legkevesbé értékes számjeggyel kezdjük, majd a második legkevesbé értékessel, végül eljutunk a legértékesebb jegyig. A módszer stabilitása itt nagyon fontos, mert csak így érhető el a megfelelő számjegysorrend egy számon belül.

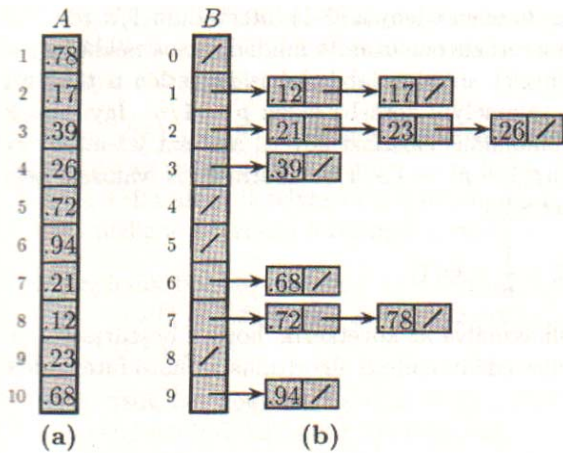
Időigény: $\Theta(d(n + k))$, ám ha d konstans és $k = O(n)$, akkor lineáris idejű a rendezés.



9.3. ábra. A számjegyes rendezés működése 7 darab 3 számjegyű számot tartalmazó lista esetén. Az első oszlop a bemenet. A többi oszlop a listát mutatja, az egyre értékesebb számjegyek alapján történő rendezések után. A függőleges nyilak azt a számjegypozíciót mutatják, amelyiken az előző listát rendeztük és így az adott listát megkaptuk.

Edényrendezés: Feltételezzük, hogy a bemenet a $[0,1)$ intervallumon egyenletes eloszlású számok sorozata. Ezt az intervallumot felosztjuk n egyenlő részre (ezek lesznek az edények). Az edényekbe szétosztjuk az elemeket, mindegyik edényben egy listát kezelve. Az azonos edényben lévőket beszűrösös módszer szerint rendezzük, majd a listákat a végén összefűzzük az elsővel kezdve.

Időigény: $\Theta(n)$



9.4. ábra. Az EDÉNY-RENDEZÉS működése. (a) Az $A[1..10]$ bemeneti tömb. (b) A rendezett listák (edények) $B[0..9]$ tömbje az algoritmus 5. sora után. Az i -edik edény az $[i/10, (i + 1)/10)$ intervallumhoz tartozó értékeket tartalmazza. A rendezett kimenet a $B[0], B[1], \dots, B[9]$ listák sorban történő összefűzéséből áll elő.

Topologikus rendezés: Ez is ide tartozik, bár eléggé speciális feltételeket szab a rendezendő halmazra. Nem árt megemlíteni. Lásd *ZV_29tetel*.