EDÉNYRENDEZÉSEK

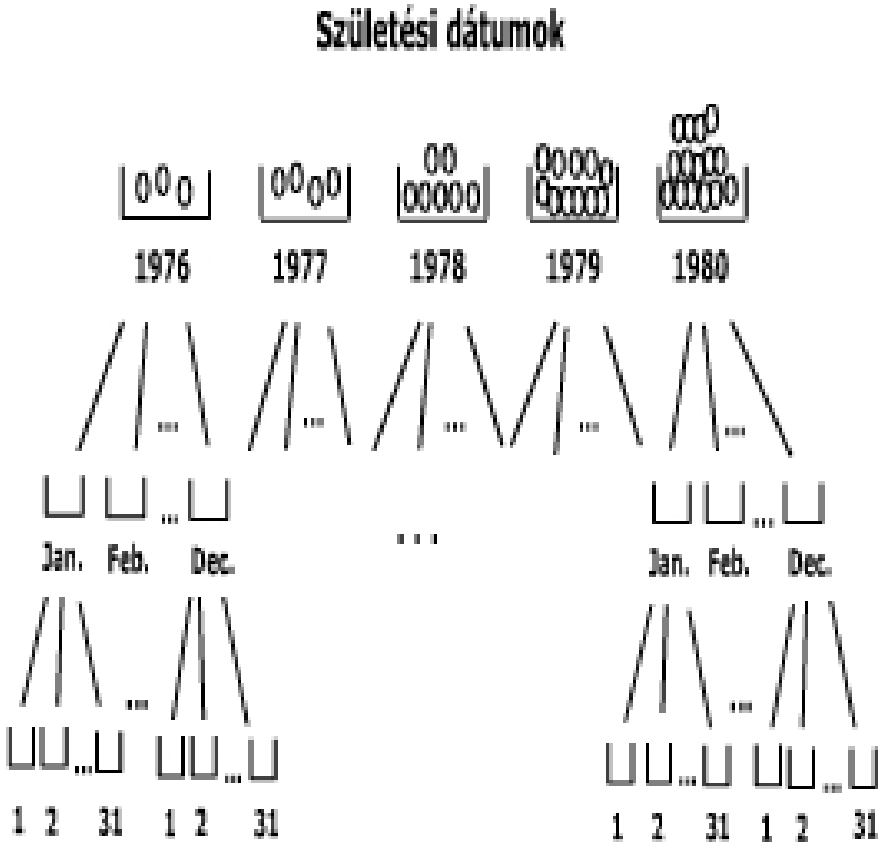
Eddig az összehasonlításos rendezésekről volt szó. Az edényrendezés abban különbözik az előbbiektől, hogy nem hasonlítjuk össze a rendezendő elemeket, hanem a rendezés során az egyes elemeket az értéküknek megfelelő „edények”-be rakjuk szét.

1)
Induljunk ki egy példából. Egy évfolyam hallgatói felírják egy-egy kis lapra a születési dátumaikat, pl. így: 1980.30.12. Szeretnénk sorbarendezni ezeket az adatokat. Ezek speciális adatok: mindhárom mező csak adott tartományból vehet értékeket:

- az évek mondjuk 1980 és 1976 között helyezkednek el,
- 12 hónap van és
- 31 nap jöhet szóba.

Első megoldás:

1. Rakjuk szét a születési dátumokat az évek edényeibe.
2. Rakjuk tovább a dátumokat a megfelelő hónapok edényeibe.
3. Rakjuk végül a dátumokat a napoknak megfelelő edénybe.
4. Vegyük ki a születési dátumokat sorban az edényekből és alakítsunk belőlük sorozatot.



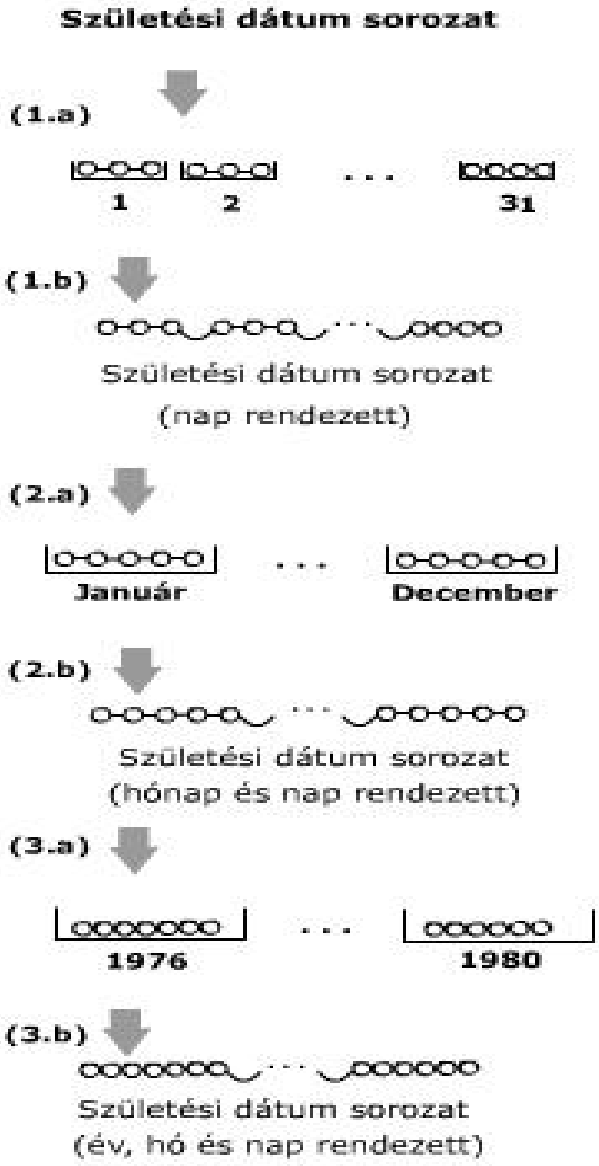
Edény-hierarchia: 5 év, 5*12=60 hónap,
kb. 60*30=1800 nap

A sorozat rendezett lesz, mert az edényeket a rendezettségüknek megfelelő sorrendben érintjük, továbbá minden edényben egy féle születési dátum lehet (0,1 vagy több példányban).

Az eljárás hatékony (lineáris), mert 4-szer mentünk végig az adatok sorozatán, azonbansok edényt (memóriát) igényel. Ez még dinamikus helyfoglalás után is igaz: az edények száma ≈ arányos az adatok számával, amíg el nem éri a maximális edényszámot.

Második megoldás:

Kicsit talán meglepő, de rendezhető a sorozat úgy, hogy az egyes mezők szerint hátulról előre haladva alkalmazzuk az edényekbe való szétrakást és az összefűzés műveletetit.



1. a) Rakjuk szét a születési dátumokat a napok edényeibe úgy, hogy az eredeti beérkezési sorrendet megtartjuk az edényben.
1. b) Fűzzük össze az 1., 2., ... 31. nap edényeiben lévő sorozatokat egyetlen sorozattá.
2. a) Rakjuk szét a sorozatot a hónapok edényeibe, 12 részsorozat formájában.
2. b) konkatenáljuk ismét az edényekben található sorozatokat egyetlen sorozattá.
3. a) Végül az évek edényeibe fűzzük szét a sorozatot.
3. b) Fűzzük össze az évek edényeinek tartalmát egyetlen sorozattá.

Gondoljuk meg, hogy így éppen a kívánt rendezés alakul ki a sorozatban.

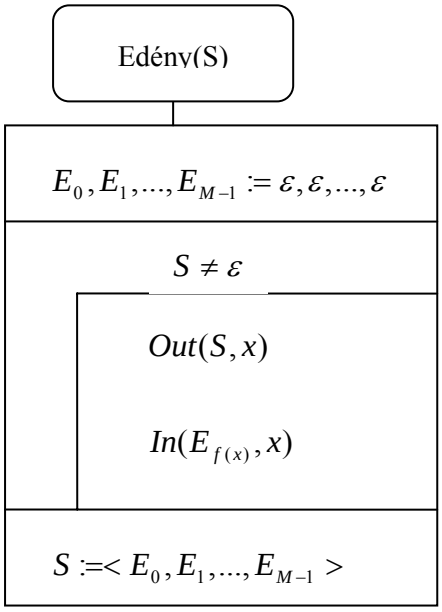
Legjobb, ha végig követjük a lépések hatását pl. a következő sorozatban:

1980.05.16.	1980.05.01.	1977.10.21.	1976.10.21.	1977.05.21.

Általános leírás

2)
Az edényekbe rakás és az edények összefűzése: Edény(S) eljárás.

Legyen K a sorozat elemeinek típusértékhalmaza: $S \in K^*$.
Legyen q egy hasító függvény (hash függvény): $\varphi \to [0...M-1]$.
Legyenek $E_0, E_1, ..., E_{M-1}$ edények, amelyek éppen olyan sorozatok, mint S. Az egyes edényekben megmarad az S-beli elemek ottani (S-beli) relatív sorrendje.
Az eljárás annyit végez, hogy végigolvassa S-et és az elemeket a φ szerinti edényekbe teszi, majd az edények tartalmát konkatenálja.



A φ hasító függvény egy fontos tulajdonsága: ha $\varphi(k_1) < \varphi(k_2)$, akkor $k_1 < k_2$.

= lorem

= hiext

konkatenáció

Ahhoz, hogy Edény(s) az egyszeri szétrakaással rendezzen, elégséges a következő feltételek valamelyike:

a) Az edényekben legfeljebb 1 elem van: $|E_i| \leq 1 \quad (i = 0,1,...,M-1)$.
b) Az egyes edényekben csak azonos elemek vannak: E_i mint halmaz legfeljebb 1-elemű $(i = 0,...,M-1)$.
c) Az egyes edények rendezettek.

Ha az a) b)és c) feltételek valamelyike teljesül, akkor tökéletes edényrendezésről beszélünk. Ennek műveletigénye:

$$T_{Edény}(n) = O(n)$$

ahol n = |S|. A ciklusban végig kell olvasni az S sorozat elemeit, a konkatenáció pedig vagy szintén minden elem végigolvasásával jár, vagy pedig M pointerművelettel megvalósítható listák esetén.

A tökéletes edényrendezés lineáris műveletigényű algoritmus.
Erős feltételeknek kell teljesülniük ehhez.

Példa: Sok embert kell magasság szerint sorbaállítani. Megéri minden egyes szóba jövő tastmagasság (cm-ben) számolva egy-egy edényt létrehozni. Teljesül a b)!

Bevezető példánkban egyetlen szétrakással nem lehtne rendezzni.
Az első megoldás alapján jutunk el az utórendezéses edényrendezéshez: a szétrakás után és az összefűzés előtt rendezzük az $E_0, E_1, ..., E_{M-1}$ edényeket, méghozzá edényrendezéssel.

A második megoldás alapján jutunk el az előrendezéses edényrendezéshez: a szétrakás előtt csaknem teljesen rendezzük a sorozatot, mégpedig edényrendezéssel.

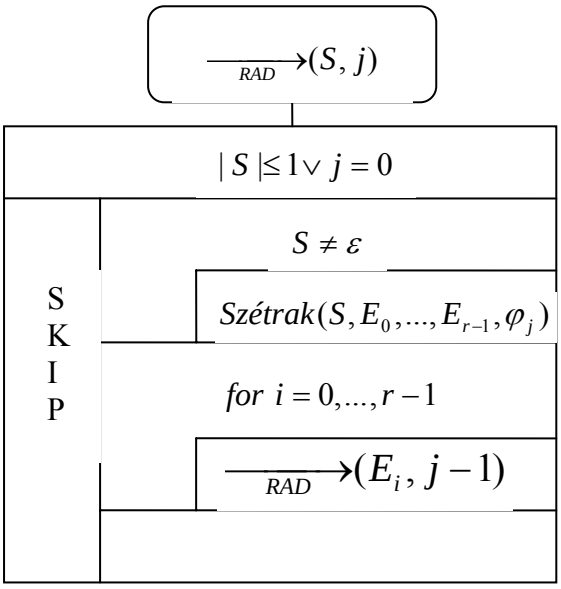
Ha az edényrendezéseket olyan számokra fogalmazzuk meg, amelyek
- r alapú számrendszerben vannak felírva és
- d számú számjegyet tartalmazzank,

akkor az úgynevezett RADIX rendezéseket kapjuk.

3)
Utórendezéses edényrendezés, RADIX „előre" ($\xrightarrow[RAD]{}$)

Az általános utórendezéses változat olyan, mint a bevezető példánk első megoldása. Ha speciálisan r alapú számokra írjuk fel, akkor $\xrightarrow[RAD]{}$ eljárást kapjuk.

$e = e_d e_{d-1} ... e_1$

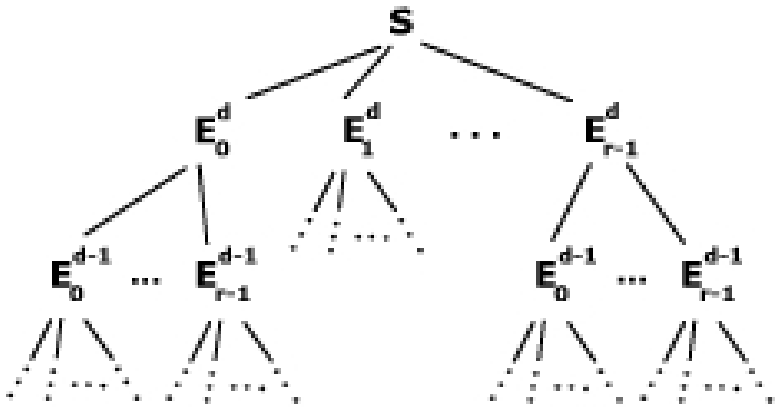


Rendez S-et a j-edik jegy szerint, majd további j-1, ..., 1 jegyek szerint is.

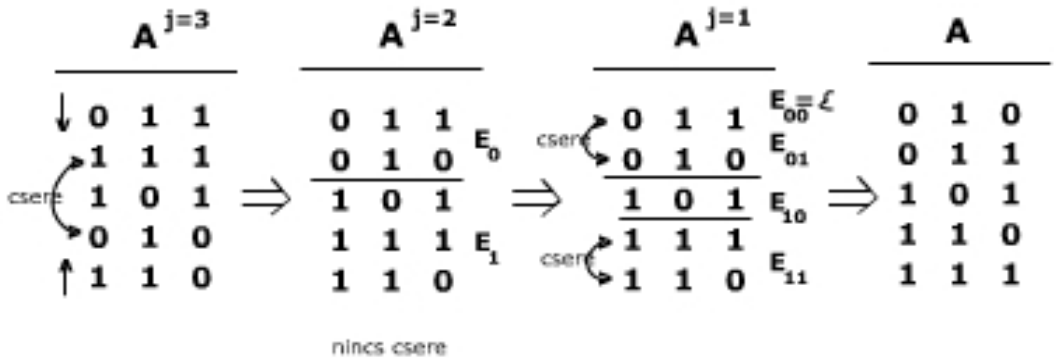
Külső hívás: $\xrightarrow[RAD]{}$(S, d)

A rekurzív hívás minden szintjén lokálisan létrejönnek az $E_0, ..., E_{r-1}$ edények. Így egy fastruktúrájú edényrendszer alakul ki.

Hány edény kell?
A rekurzióban lokális változók újabb és újabb példányai foglalódnakle; max. $r + r^2 + ... + r^d$ edény jön létre.
Valójában gyakran jóval kevesebb elég, hiszen sok olyan edény lehet, amely üres lenne.



Speciálisan d hosszú <u>bináris számokra</u> (r = 2) egyetlen tömbben megoldható a rendezés. Ekkor minden szinten két edény kell: E_0 *és* E_1. Egy edény egy tömbrésztlet. Ezt további két edényre osztjuk úgy, hogy két mutatóval megyünk szemmbe és cseréljük a számokat, ha kell, egészen addig, amíg a két mutató át nem lépi egymást.
Pl.:



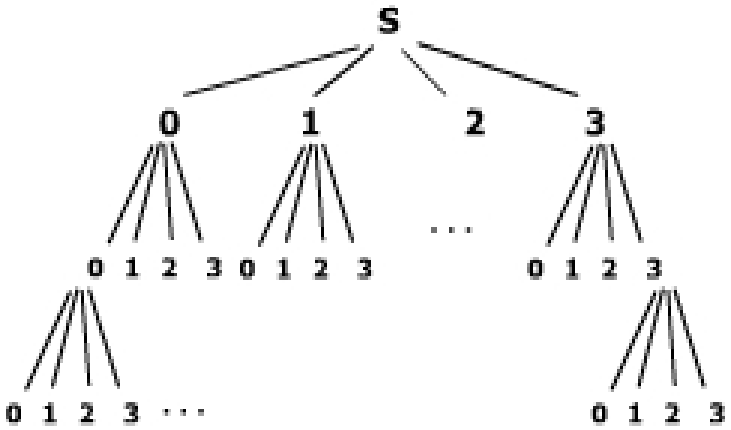
A $\xrightarrow[RAD]{}$ eljárás lineáris: d*|S|+|S| műveletet (mozgatást) igényel: +szétrakások, összefűzések.

Bináris esetben legfeljebb d*|S| = d*n elem mozgatást végzünk.

Példa:

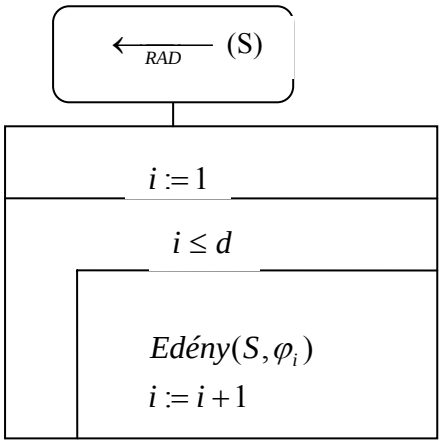
d = 3
r = 4

S = 331
230
023
121
202
312
123
003
013



4)
Elrendezéses edényrendezés, RADIX „visszafelé” ($\xleftarrow[RAD]{}$)

Az általános előrendezéses változat úgy működik, mint a bevezető példa második megoldása. Ha speciálisan d hosszú r alapú számokra írjuk fel, akkor a $\xleftarrow[RAD]{}$ eljárást kapjuk.



Az eljárás az $e = e_d e_{d-1} ... e_1$ számot jobbról balra indulva, az alacsony helyiértékek felől indulva pozíciónként szétrakja edényekbe, majd összefűzi az edények tartalmát. Az i-edik pozíción a φ_i hasító függvényt alkalmazzuk: $\varphi_i(e) = e_i$.

Az 1, 2, ..., i-edik pozíciókon végrehajtott szétrakás és összefűzés után S „i-rendezett” lesz. Ezt úgy definiáljuk, hogy ha $x = x_d x_{d-1} ... x_2 x_1$ és $y = y_d y_{d-1} ... y_2 y_1$ és $\le i$ jelöli az i-rendezettségét , akkor $x <_0 y$ minden x, y-ra

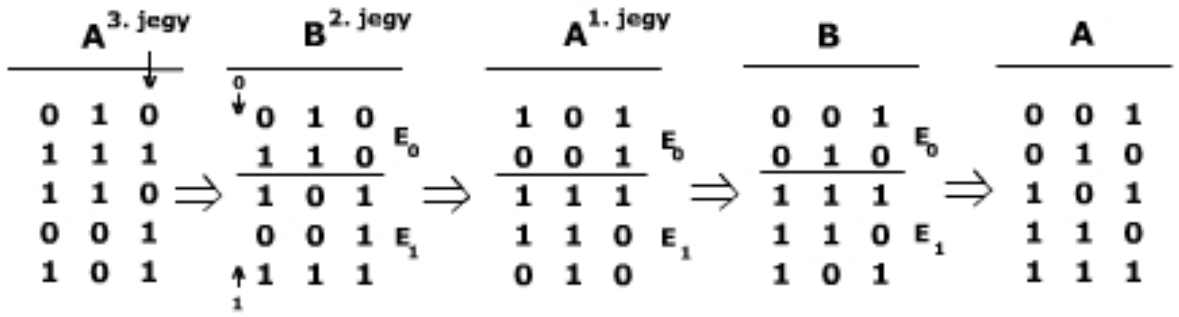
$x \le_i y \Leftrightarrow x_i < y_i$ vagy $x_i = y_i$ és $x \le_{i-1} y$ $(i > 0)$

Ekkor a d-rendezés a közönséges rendezés.

Hatékonyság: 2*d-szer megyünk végig az S sorozaton, így T(n) = O(d*|S|).
Memóriaigény: az eredeti sorozat és r számú edény: (r + 1)*|S|.

Szokásos implementáció: (*Ábra) S fejelemes láncolt lista. Az edényeket egy „fej” és egy „vége”mutató ábrázolja. A szétrakás és az összefűzés az elemek láncolácával megoldható. Az összefűzéskor nem kell az egyes edények részlistáit végigolvasni, hanem egy darabban lehet őket láncolni. A memóriaigény is kisebb: |S| elem + 1 fejelem (+ pointer) + |S| pointer + r db (fejelem + végepointer) (+ pointer).

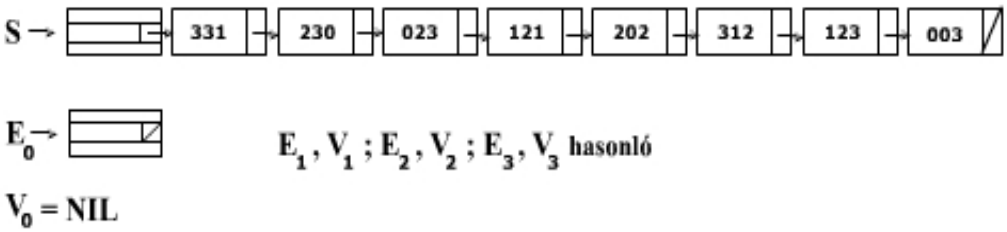
Speciálisan d hosszú bináris számokra (r = 2) egyetlen tömbben megoldható a rendezés. A számokat tartalmazó tömb kezdetben A, a másik B tömbben lesz a két edény, mégpedig szembe fordítva. Azután B-ből pakolunk A-ba, és így tobább... váltakozva:



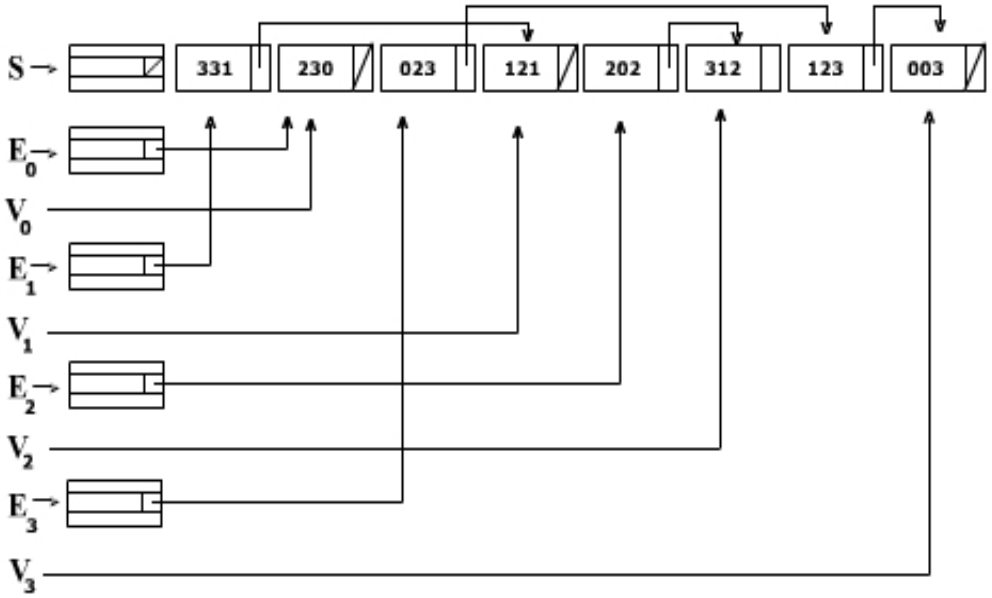
Végül a két szembefordított edényt A-ba pakoljuk folyamatosan.

*Ábra: A listás implementáció

Kezdetben:



Szétrakás:
i=1
(utolsó jegy)



Összefűzés: