16. tétel

Adatszerkezetek és algoritmusok vizsga

Frissült: 2013. január 28.

Külső rendezés

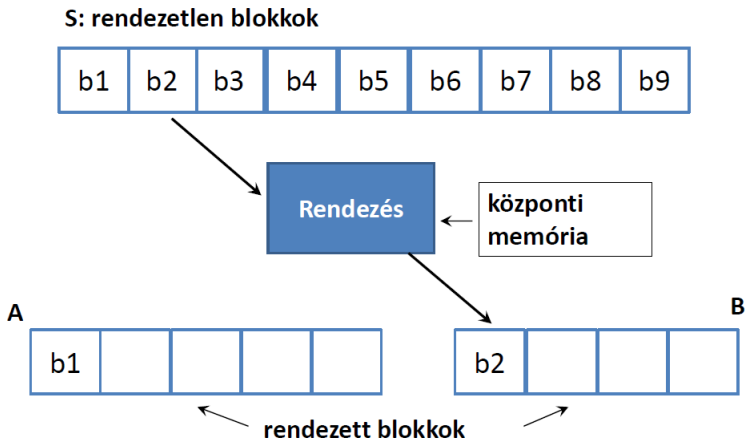
Az eddig látott rendezéseknél feltettük, hogy az adatok a központi memóriában vannak. Ennek megfelelt, hogy a hatékonyságot az összehasonlítások számában mértük. Ha az adatok háttértárban vannak, akkor a futási idő döntő részét az I/O utasítások teszik ki. (Az I/O egysége az 1 blokk, ami k x 512 byte valamely kis k-val, pl. 1024 v. 2048 byte. Ezt lapnak is nevezik.) A háttértár lehet szalag v. lemez (ennek különféle változatai). A hatékonyságot a szükséges blokk I/O-k számában mérjük. Külső rendezésre igazából csak az összefésüléses rendezés (MergeSort) alkalmas.

Az összefésüléses rendezés külső tárakon

Adott egy S szekvenciális input fájl, amely n blokkból áll, minden blokkban adott számú rekorddal. (Pl. 1 blokk = 1024 byte és ezen 3 rekord foglal helyet.) A blokkok tartalma rendezetlen. Az összefésülést iteratív módon végezzük, úgy, hogy az egyes „menetek" végén egyre nagyobb darabok, vagyis egyre több szomszédos blokk lesz rendezett. Az összefésülést menetenként váltakozva az A, B, illetve a C,D fájlokba végezzük, végül a teljesen rendezett eredményt S-be írjuk. A közbülső menetekben azért lesz 2 output fájl, mert az összefuttatás eredményét az „egyet ide, egyet oda" elv alapján írjuk ki.

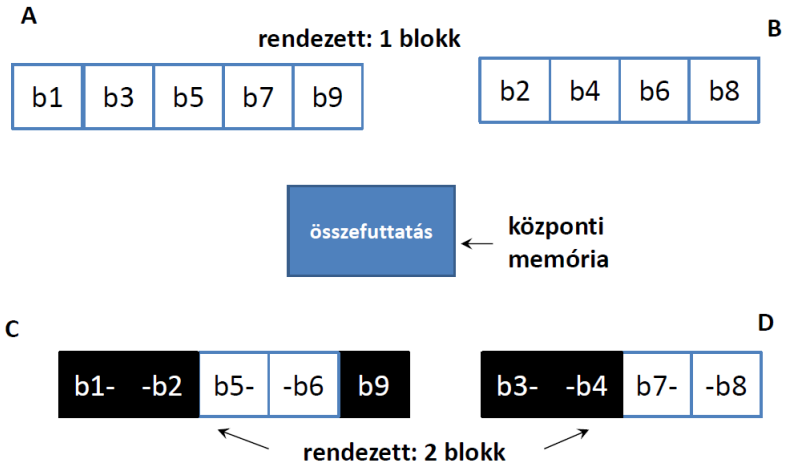
Menete

1. beolvassuk S rendezetlen blokkjait, valamilyen belső rendezővel rendezzük, majd kiírjuk felváltva A-ba, illetve B-be. Itt még nem volt összefésülés.

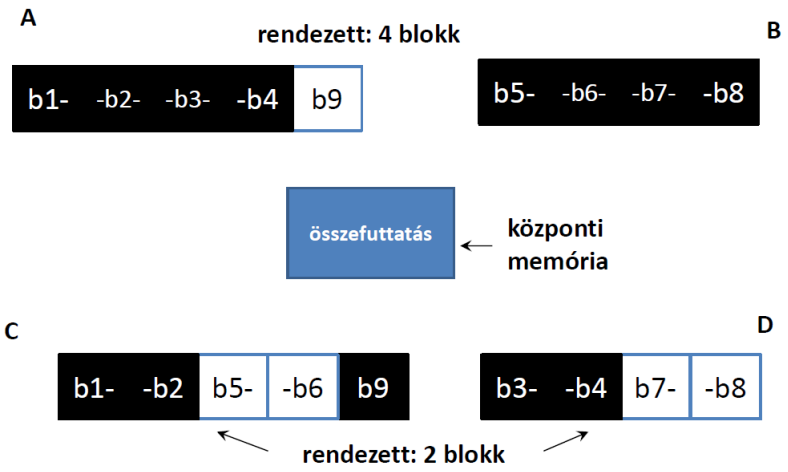


2. Sorban beolvassuk A és B 1-1 blokkját. Ezek rendezettek. Összefésüljük őket és a rendezett két blokkot felváltva C-be, illetve D-be írjuk.

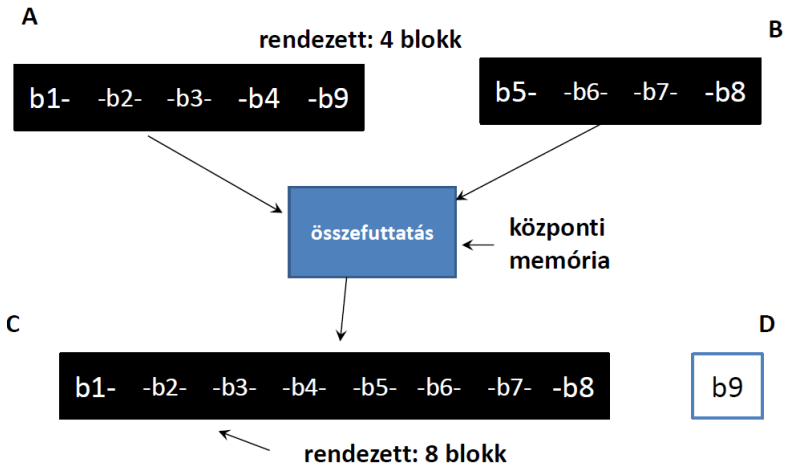
   Az A utolsó blokkjának nincs párja, így azt kiírjuk C-be. A C és D fájlban a rendezett részek hossza 2 blokk, illetve a maradék esetében 1 blokk.



3. C-ből és D-ből olvasunk 2-2 rendezett blokkot, összefésüljük őket és felváltva A-ba és B-be írjuk a rendezett 4 blokkot. A C-beli utolsó töredék blokk változatlanul A végére kerül.

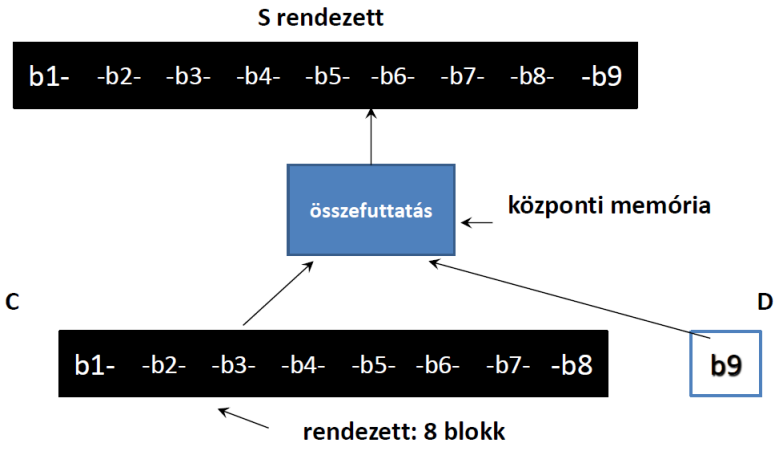


4. C-be kerül A és B 4-4 rendezett blokkja összefésülésének a 8 blokk hosszú rendezett eredménye, D-be pedig a maradék egy blokk.



5. C 8 blokkját és D 1 blokkját összefésüljük S-be.

   Elnevezés: egy k blokkból álló összefüggő rendezett részt k hosszú *futam*nak nevezünk.



Kiegészítő megjegyzések

1. Ebben a példában az S 9. blokkja csaknem végig nem került kapcsolatba más rendezett részekkel, csak az utolsó menetben került összefésülésre. Ha végrehajtjuk a fenti eljárást n=15-re, akkor a páratlan töredék maradék rész mérete így alakul: 1, 3, 7, vagyis a fenti jelenség nem törvényszerű.

2. Ha a központi memória mérete korlátozott és nem képes befogadni az egyre növekvő méretű rendezett részeket (futamokat), akkor ezek összefésülését lehet „pufferelve", akár blokkonként végezni. Ez azért lehetséges, mert az összefuttatás egysége a rekord.

Általában:

- 1. menet eredménye: 1 hosszú futamok
- 2. menet eredménye: 2 hosszú futamok
- 3. menet eredménye: 4 hosszú futamok
- (k-1). menet eredménye: $2^{(k-2)}$ hosszú futamok
- k. (utolsó) menet eredménye: $2^{(k-1)}$ hosszú egyetlen futam = S

(előtte maradék mindig lehet)

Gyorsítási lehetőségek:

- nagyobb kezdő futamokat hozunk létre
- három fájlos külső rendező

Három fájlos külső rendező

Érdekes algoritmushoz jutunk, ha az m-felé fésülésnél nem 2m db fájl-t használunk, hanem csak m+1 db-t.

Ezt m=2 esetére nézzük meg, ami az eredeti eset. Ekkor tehát 3 fájl-t használunk, mondjuk A, B, C-t.

Az 1. menetben szétosztjuk S immár rendezett blokkjait A-ba és B-be.

A második menetben elkezdjük A és B blokkjainak az összefésülését, de most csak 1 fájl tudja fogadni az eredményt, a C fájl. Ezért C-be fésülünk össze egészen addig, amíg A és B egyike ki nem ürül.

Ekkor új menetet kezdünk a két nem üres fájllal s.i.t.

Példa. S n=13 blokk.

A táblázatban az első szám a futamok számát jelenti, zárójelben pedig a futamok hossza áll.

	1.	2.	3.	4.	5.	6.	7.	8.
A	7(1)	1(1)	0	1(5)	0	1(9)	0	1(13)=S
B	6(1)	0	1(3)	0	1(7)	0	1(11)	0
C		6(2)	5(2)	4(2)	3(2)	2(2)	1(2)	0

Látjuk, hogy a második menet végén képződött 6 db 2 hosszú futam csak egyesével tud elfogyni – érezhetően sok lépésben.

Ennek oka az, hogy az 1. menet végi két futamszámnak 1 a különbsége: 7-6=1 Hogy tudnánk ezen javítani??

Eszünkbe jut a Fibonacci sorozat: 0,1,1,2,3,5,8,13,....

Ahol két szomszédos elem különbsége – az első néhány tag kivételével – 1-től különböző.

Innen jön a gondolat, hogy az első menetben írjunk annyi futamot A-ba és B-be, mint a Fibonacci sorozat két (alkalmas) szomszédos eleme, és lépkedjünk visszafelé a sorozaton az egyes menetekben.

	1.	2.	3.	4.	5.	6.
A	8(1)	3(1)	0	2(5)	1(5)	0
B	5(1)	0	3(3)	1(3)	0	1(13)=S
C		5(2)	2(2)	0	1(8)	0

Belátható, hogy a 3 fájlos rendező éppen akkor fut le leggyorsabban, ha így járunk el, azaz A-ban és B-ben két szomszédos Fibonacci számnak megfelelő számú kezdő futamot hozunk létre.

Ha N nem Fibonacci szám, akkor vagy levágjuk és félretesszük a felesleget, és a végén még összefésüljük a kialakult eredménnyel, vagy pedig éppen fordítva: (virtuális) kiegészítéssel alkalmasan megnöveljük az input állomány méretét.

Nézzük meg, hány lépésben érjük el F_k és F_{k-1}-ből az 1, 0 számokat úgy, hogy minden menetben egy újabb számot tudunk lefelé lépni:

$\lvert S\rvert \approx F_{k+1} \Rightarrow$	1.	2.	....	(k-1).	k.
	F_k	F_{k-1}	....	1	1
	F_{k-1}	F_{k-2}	....	1	0

Látható, hogy a menetek száma ekkor k, hiszen F_k-től F_1-ig vezet az út a táblázatban. Ki kell fejezni n-et F_{k+1}-gyel, ami az input fájl (közelítő) mérete blokkokban. Jelöljük $N := F_{k+1}$

$N:=F_{k+1}$

$F_0 = 0, F_1 = 1, (F_2 = 1, F_3 = 2, F_4 = 3,\ldots)\ \ F_{k+1} = F_k + F_{k-1}\ (k \geq 1)$

$F_k = 1/\sqrt{5} * [((1+\sqrt{5})/2)^k - ((1-\sqrt{5})/2)^k] \quad \Rightarrow F_k \approx 1/\sqrt{5} * ((1+\sqrt{5})/2)^k$

$1{,}6180 \qquad -0{,}6180$

Az $A = 1 + \sqrt{5}/2$ az aranymetszés aránya, amely kielégíti az $A^2 - A - 1 = 0$ egyenletet.

Ezt átrendezve: $A^2 = A + 1$, amiből teljes indukcióval megmutatható, hogy $k \geq 2$ esetén:

$A^{k-2} \leq F_k \leq A^{k-1}$

Ha $N=F_{k+1}$, akkor $A^{k-1} \leq N$,

$k-1 \leq \log_A N = \log_2 N / \log_2 A \approx 1{,}44 * \log_2 N$

$k \leq 1{,}44 * \log_2 N + 1$