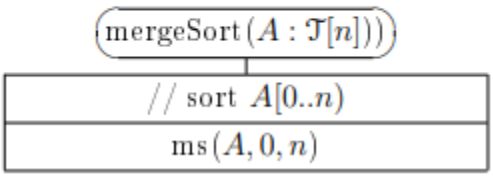
Összefésülő algoritmus

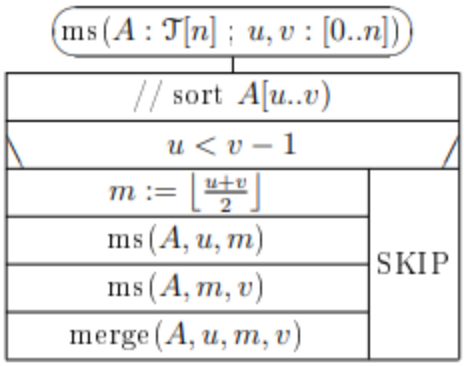
Feladat

Szemléltesse az összefésülő rendezés (mergesort) működését a tanult módon egy tetszőleges 10 elemű, egész számokat tartalmazó tömbön. A bemeneti tömb legyen véletlenszerű, nem állhat csupa azonos elemből, és nem lehet eleve rendezett. Mutassa be, hogyan alakítja ki és hogyan fésüli össze az algoritmus a résztömböket. Az egyes összefésülés meneteknél adja meg mit-mivel hasonlítana össze az algoritmus. Az utolsó összefésülő menetet (5-5 rendezett elem összefésülését) részletesen is szemléltesse. Adja meg összesen hány kulcs-összehasonlítás történt és vesse össze a kapott összehasonlítás számot a maximum kiválasztásos rendezéssel, ott hány összehasonlítás történ volna.
A megoldást digitális formában (pdf, docx, pptx) kell kidolgozni és beadni, papírra írt és lefotózott kidolgozást nem fogadunk el! A bemeneti tömböknek mindenkinél különbözőnek kell lennie! A megoldás legyen pontos, jól áttekinthető és egyértelmű!

Elemzés

kezdő függvény:





Az ms függvény rekurzív módon hívja meg önmagát, tehát az ms(A,u,m) ágon megyünk, egészen addig, amíg már csak 1 elemek nem maradtak a vizsgálathoz(u<v-1).

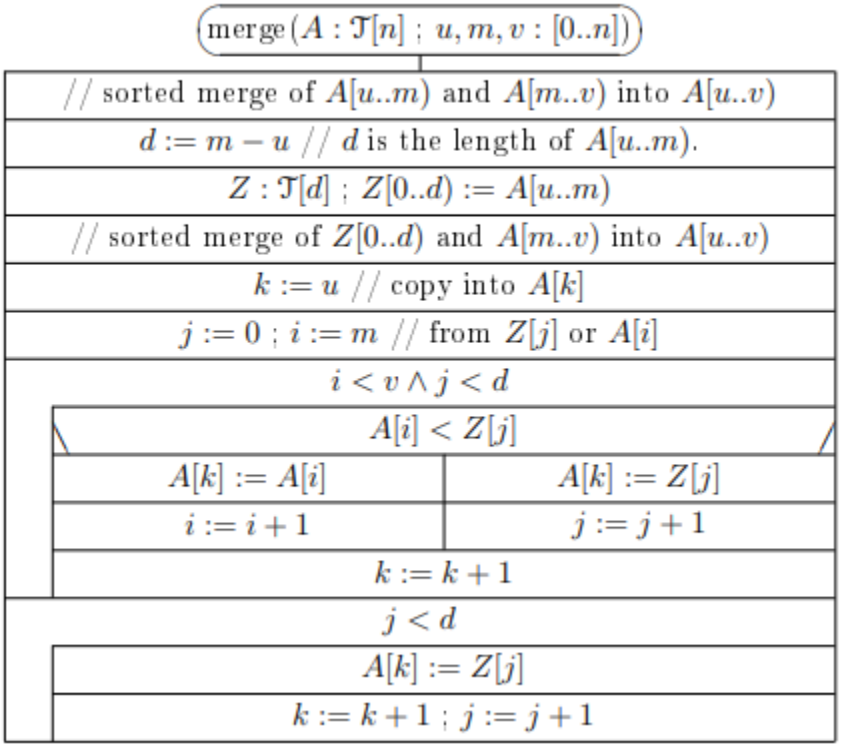
A tömb elemei : *10, 9, 8, 7, 20, 1, 14, 2, 1, 0*

„Kettévágjuk[1]" a tömböt: 1. *10, 9, 8, 7, 20,* 2. *1, 14, 2, 1, 0*

„Kettévágjuk" az 1. részt: 1 : *10, 9,* 2. *8, 7, 20,*

„Kettévágjuk" az 1. részt: : 1 : *10,* 2. *9,*

Ezután jön az összefésülés a már szétszedett elemekre.

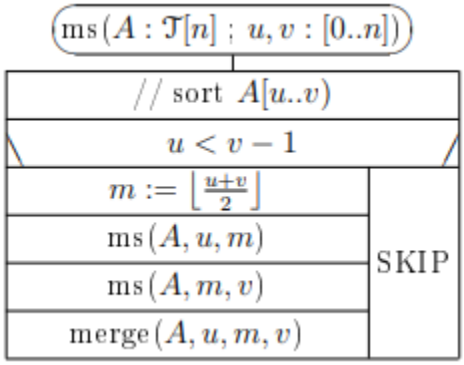


A[2] értéke merge() végén*: 9, 10,*

[1] Valójában itt nem kettévágás történik, csak a vizsgálandó tartományt szűkítjük, de a tömbünk változatlan marad.

[2] Természetesen az A nem csak ebből a két elemből áll, a többi elemet itt nem soroltam fel, ők változatlanul szerepelnek a tömbben, csak a vizsgálatban érintett elemek jelennek meg a következőkben.

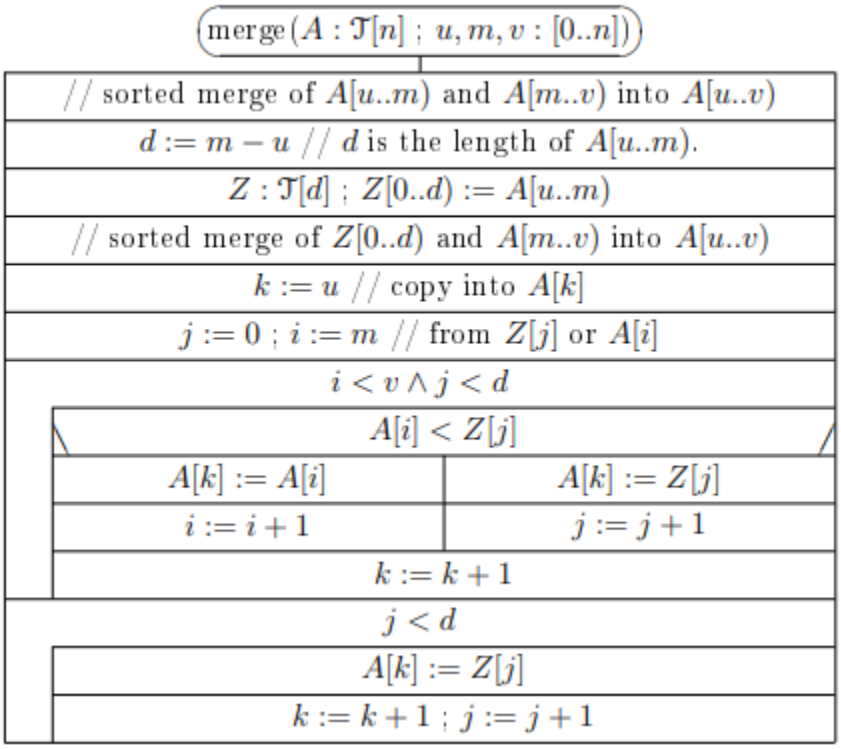
Ezután folytatóduk az ms rekúrzív függvény a 2. részekkel (ms(A,m,v))



„Kettévágjuk” az 2. részt: 1 : *8* 2. *7, 20,*

„Kettévágjuk” az 2. részt: 1 : *7,* 2. *20,*

Megint 1-1 elem maradt, indulhat az összefésülés.



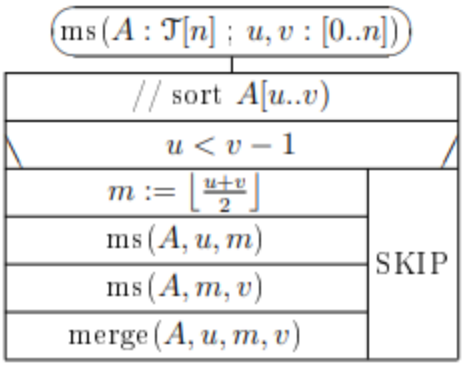
1. hívás A értéke merge() végén: *7, 20,*

Mivel ebben az ágban nem maradt több szétszedni való, ezért indulhat ezekre is az összefésülés.

2. hívás **A** értéke merge() végén: *7, 8, 20,*
3. hívás **A** értéke merge() végén *: 7, 8, 9, 10, 20,*

Ezután a második csoport következik: *1, 14, 2, 1, 0*

ugyanúgy addig „szeleteljük”, amíg 1-1 elemet nem kapunk.



„Kettévágjuk” az 2. részt: 1 : *1, 14* 2. *2, 1, 0,*

Kettévágjuk” az 1. részt: 1 : *1* 2. *,14*

Jöhet az 1. rész összefésülése:

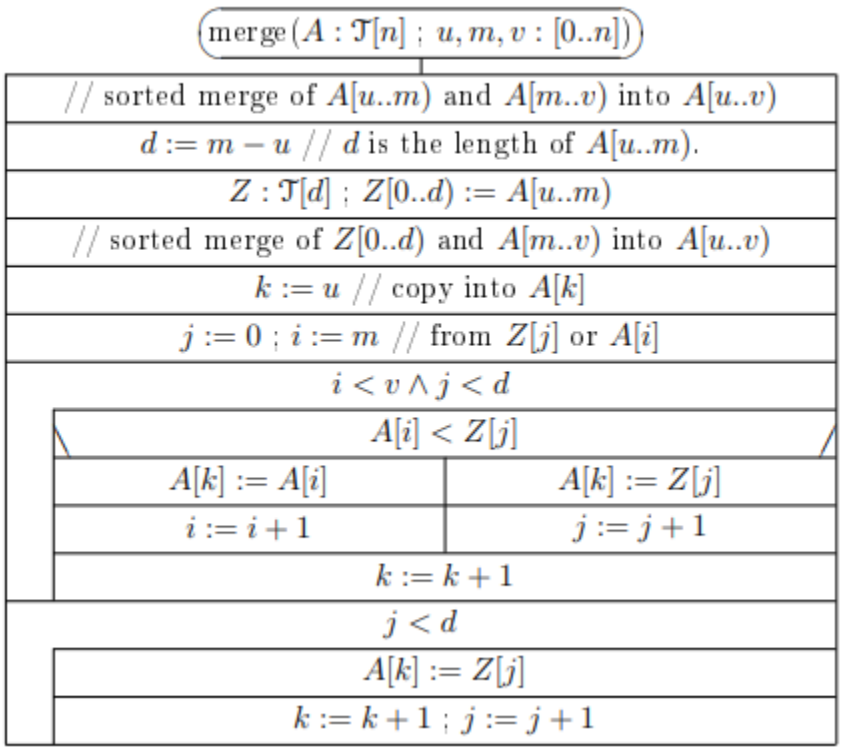
4. hívás **A** értéke merge() végén: *1, 14,*

„Kettévágjuk” az 2. részt: 1 : *2* 2. *1, 0,*

„Kettévágjuk” az 2. részt: 1 : *1* 2. *0,*

Végére értünk a tömbnek, nincs tovább szétbontani való csoport, ezután jön az összefésülés:

5. hívás **A** értéke merge() végén: *: 0, 1,*
6. hívás **A** értéke merge() végén *: 0, 1, 2,*
7. hívás **A** értéke merge() végén *: 0, 1, 1, 2, 14,*
8. hívásra pedig előáll a rendezett tömb : *0, 1, 1, 2, 7, 8, 9, 10, 14, 20,*
   Utolsó merge() metódus hívás részletezése:



Itt látható, hogy a z megkapja a A tömbnek az első 5 elemét. Ez tulajdonképpen a legelső szétválasztásnál kapott két csoport külön-külön rendezett tömbje.

Tehát z értéke: *7, 8, 9, 10, 20,*

A értéke ebben a pillanatban: *7, 8, 9, 10, 20, 0, 1, 1, 2, 14,*

A vizsgálat során az utolsó 5 db-ot fogjuk nézni, és cseréljük fel az első 5 elem valamelyikével, amennyiben szükséges.

Első elemünk a 0, őt fogjuk vizsgálni, hogy ez kisebb e a Z-nek az első elemével. Mivel kisebb, ezért kicseréljük, és tovább lépünk az **A** tömbben.

z értéke: *7, 8, 9, 10, 20,*

A értéke*: 0, 8, 9, 10, 20, 0, 1, 1, 2, 14,*

Mivel még se a z sem az A tömbön nem értünk végig folytatódik a vizsgálat. A következőkben az 1-et hasonlítjuk össze a 7-el. Ez kisebbnek minősül, tehát mehet az A tömb második helyére, és tovább lépünk az a tömbben. Így fog kinézni a vizsgálat után a tömbünk:

z értéke: *7, 8, 9, 10, 20,*

A értéke*: 0, 1, 9, 10, 20, 0, 1, 1, 2, 14,*

Ezután megint az 1-et hasonlítjuk össze a 7-el, megint kisebb, bekerül a harmadik helyre, és tovább lépünk a tömbben.

z értéke *7, 8, 9, 10, 20,*

A értéke*: 0, 1, 1, 10, 20, 0, 1, 1, 2, 14,*

Ezután megint az 2-et hasonlítjuk össze a 7-el, megint kisebb, bekerül a negyedik helyre, és tovább lépünk a tömbben.

z értéke *7, 8, 9, 10, 20,*

A értéke*: 0, 1, 1, 2, 20, 0, 1, 1, 2, 14,*

A 14-et hasonlítjuk össze a 7-el, mivel kisebb, mint 14, ezért elfoglalja az ötödik helyet.

z értéke *7, 8, 9, 10, 20,*

A értéke*: : 0, 1, 1, 2, 7, 0, 1, 1, 2, 14,*

Most a 14-et hasonlítjuk össze a 8-al, hasonlóan az előzőhöz a 8 bekerül az A tömb hatodik helyére, és tovább lépünk a z tömbben.

z értéke *7, 8, 9, 10, 20,*

A értéke*: 0, 1, 1, 2, 7, 8, 1, 1, 2, 14,*

A 14-et hasonlítjuk össze a 9-al, ezért a 9 bekerül az A tömb hetedik helyére, és továbblépünk a z tömbben.

z értéke *7, 8, 9, 10, 20,*

A értéke*: 0, 1, 1, 2, 7, 8, 9, 1, 2, 14,*

A 14-et hasonlítjuk össze a 10-al, ezért a 10 bekerül az A tömb nyolcadik helyére, és továbblépünk a z tömbben.

z értéke *7, 8, 9, 10, 20,*

A értéke*: 0, 1, 1, 2, 7, 8, 9, 10, 2, 14,*

A 14-et hasonlítjuk össze a 20-al, mivel a 14 kisebb, ezért bekerül az A tömb kilencedik helyére, és az A továbblépünk a tömbben.

z értéke *7, 8, 9, 10, 20,*

A értéke*: 0, 1, 1, 2, 7, 8, 9, 10, 14, 14,*

Itt véget értünk az A tömbön, ezért a ciklusból kilépünk. Látható még, hogy nem vagyunk készen, hisz még a 20-as elemnek is a helyére kell kerülnie.

Mivel a z tömbnél az ötödik helyen állunk, az A tömbben pedig a kilencedik helyre tettük a legutolsó számot, ezért az a következő ciklus csak egyszer fordul le, és a 10. helyre bekerül a 20-as szám. Így előállt a rendezett tömbünk.

Összehasonlítás

Összesen **30 db** összehasonlítás történt, 25 az első ciklusban, tehát 25 vizsgálat volt, ahol megnéztük melyik a nagyobb, majd a kimaradt elemeket 5 alkalommal vizsgáltuk, hogy hova kerüljenek a tömbben.

A maximum keresés lényege, hogy először kiválasztjuk a minimum elemet az intervallumból, majd az intervallum elejére tesszük. **44 db** (9+8+7+6+5+4+3+2) összehasonlítást kell végeznünk ahhoz, hogy minden elem a helyére kerüljön, tehát jóval többet, mint összehasonlító rendezés esetén.