10. Versenyrendezés
(Tournament sort)

10.1. A versenyrendezés módszere

Maximum-kiválasztásos rendezés, amelyet $n = 2^k$ számú elemre ismertetünk, de tetszőleges elemszám mellett megvalósítható. Az eljárás a kieséses verseny lebonyolítását veszi alapul. Ezt a rendezést a gyakorlatban nem használják, mégis egyszerűsége és elméleti érdekessége miatt érdemes bevezetni. Mellette szól még az is, hogy jó bevezetésnek tekinthető a kupacrendezéshez (heap sort).

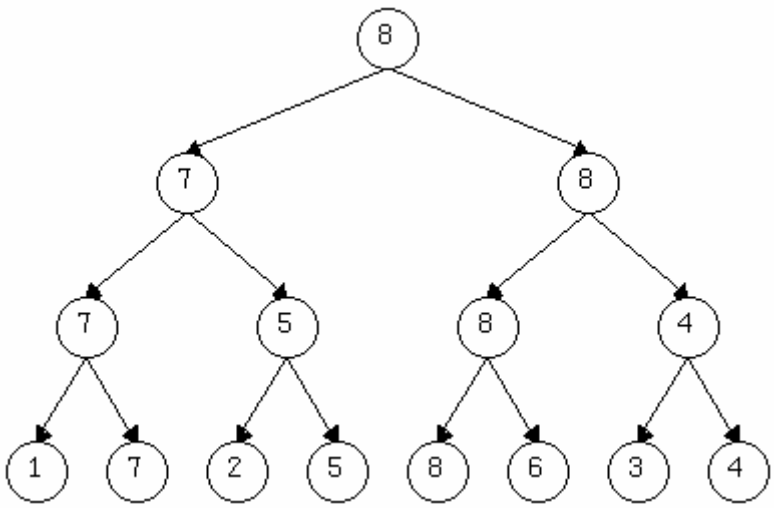
A rendezés egy speciális 1. menetet, majd azután $(n-1)$ menetet hajt végre. Az 1. menetben kiválasztja a legnagyobb elemet (és ki is írja azt), olyan módon, mint ahogy az a kieséses sportversenyeken történik. Az erre felhasznált adatszerkezet a versenyfa (tournament), amely modellezi a verseny tábláját. Az elemek a fa levelein foglalnak helyet, az egyes fordulók győztesei a fa belső pontjainak egy-egy szintjét alkotják. A döntőben nyertes maximum a fa gyökerébe kerül. A maximumot, miután kiírtuk, a kiindulási helyén, a fa egyik levelében $(-\infty)$ értékkel helyettesítjük.

A 2. menetben újra játsszuk azt az ágat, ahová most a $(-\infty)$-t beírtuk. Ez a néhány meccs kiválasztja a második legnagyobb értéket, hiszen az a korábbi menetben ezen az ágon került összehasonlításra a legnagyobb elemmel. A második legnagyobb elemet is kiírjuk, és eredeti helyére szintén $(-\infty)$-t írunk.

Így haladunk tovább, minden menetben a következő maximumot megkeresve és kiírva. Mivel az egyes menetekben kiválasztott legnagyobb elemek csökkenő sorozatot alkotnak, a rendezés kimenetén az eredeti sorozat rendezett formában képződik.

A rendező eljárás felhasználja a dinamikus programozás elvét, mivel feljegyzi egy adott menet maximum kiválasztásának az összehasonlításait.

Példa:

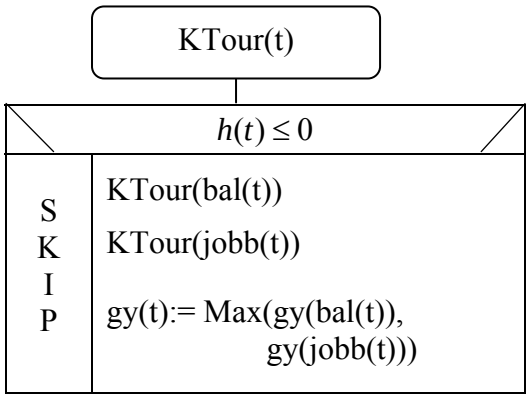


Def.: A tournament olyan teljes bináris fa, amelynek minden belső pontjában a két gyermek nagyobbika foglal helyett.

Először ADS szinten írjuk le az algoritmust, mert a bináris fát szem előtt tartva ez könnyebben megy. Azután majd leírjuk azt a változatot, amely a versenyfa tömbben tárolt változatán működik.

10.2. A versenyrendezés rekuzív algoritmusa (ADS-szint)

Kezdő tournament létrehozása:

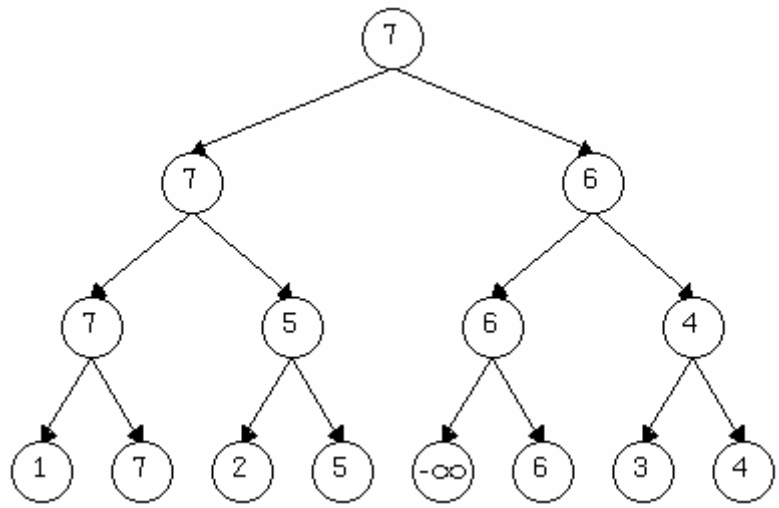


A fa kitöltése: legjobb versenyző /legnagyobb érték meghatározása.

$n = 2^k$ a versenyzők száma

$\ddot{O}_{KT}(n) = n - 1$, ugyanis az egyes szinteken:
$\ddot{O}_{KT}(n) = 1 + 2 + ... + 2^{k-1} = 2^k - 1 = n - 1$

Folytatás: a 2., 3., …, *n*-edik legjobb versenyző kiválasztása.

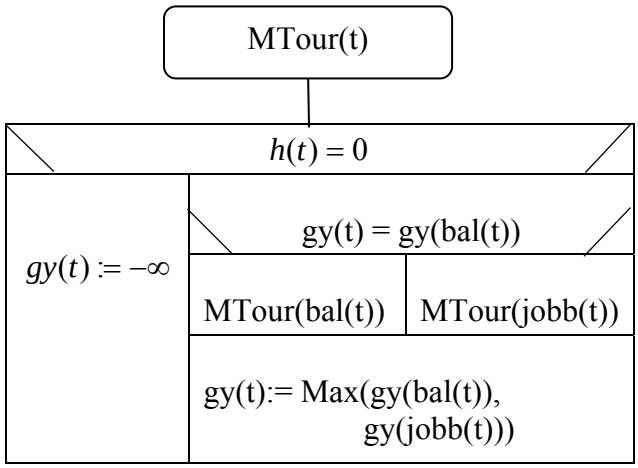


A rajzon az látható, hogy a maximális 8 eredeti helyére $(-\infty)$-t írtunk, majd lejátszottunk ezen az ágon 3 mérkőzést, azaz végrehajtottunk 3 összehasonlítást: $(-\infty,6)$, (6,4) és (7,6).

Az újra játszott ág győztese a 7 a második legnagyobb elem.

Ezt $(n-1)$-szer ismételjük.

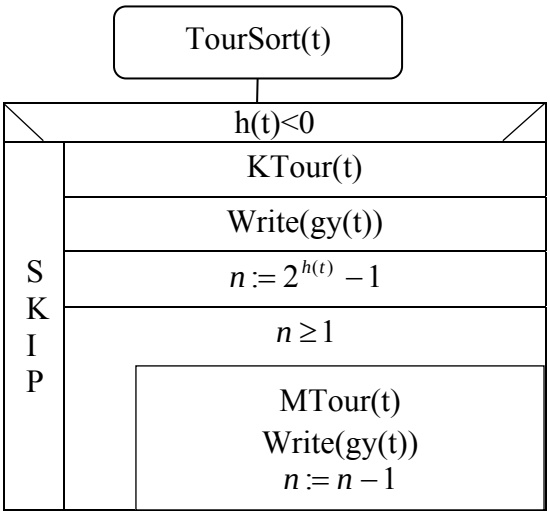
A mérkőzések újra játszása a tournament egy ágán:



A gyökérben lévő elem helyének megkeresése a levelek között, majd azon az ágon a mérkőzések újra játszása.

$\ddot{O}_{MT}(n) = 2k = 2\log_2 n$

A versenyrendezés (tournament sort):



A teljes eljárás összehasonlítás száma:

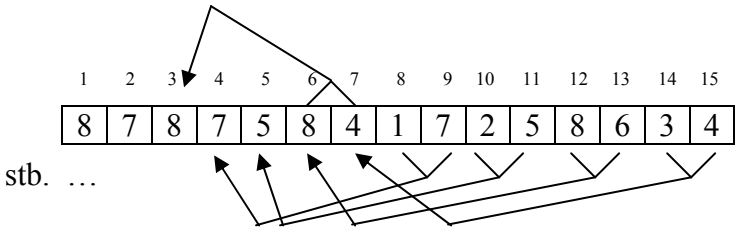
$$\ddot{O}_{TS}(n) = (n-1) + (n-1)2\log_2 n = \\ = (n-1)(2\log_2 n + 1) = \Theta(n \log_2 n)$$

Memória igény: $2n-1$

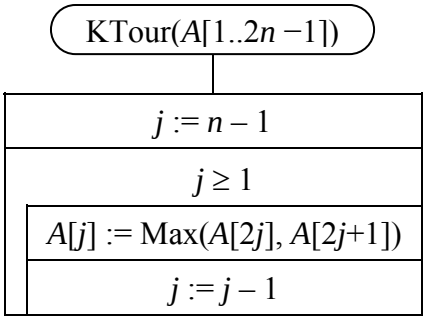
„Hibája”: nem helyben rendez!

10.3. A versenyrendezés iteratív algoritmusa (ábrázolás szintje)

Aritmetika ábrázolás esetén megadjuk a TourSort iteratív változatát. A versenyfát szintfolytonosan elhelyezzük egy tömbben. A rendezendő elemek a fa levelén foglalnak helyet, ezért a $2n-1$ elemszámú tömb utolsó n eleme van kitöltve. A megelőző $n-1$ elemet éppen a Ktour eljárás tölti ki, méghozzá jobbról balra haladva.



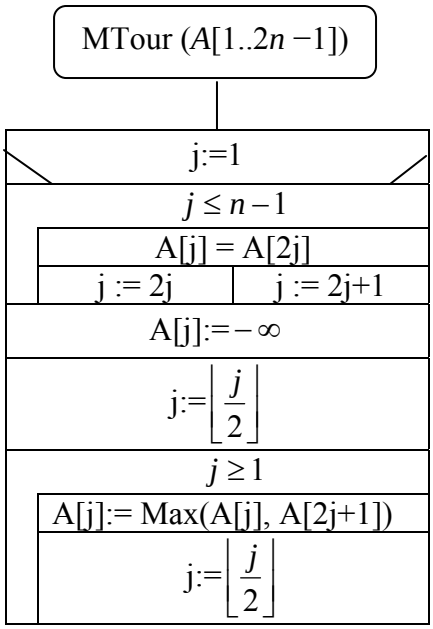
Kezdő tournament létrehozása:



for $j = n-1$ *to* 1 *by* -1

(Használhatunk for-ciklust is!)

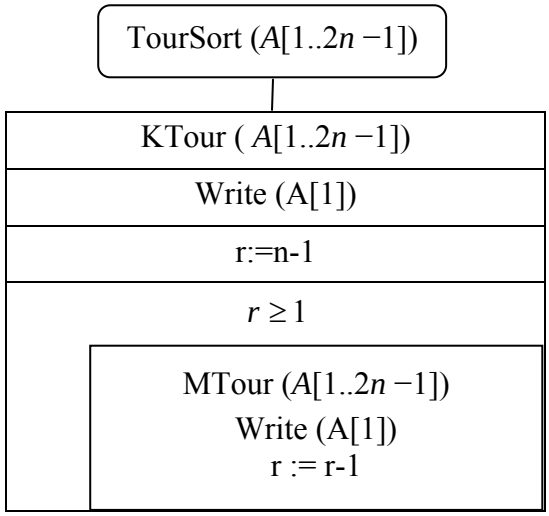
A mérkőzések újra játszása a tournament egy ágán: **(**Q: $n \geq 1$)



A versenyrendezés (tournament sort):

A TourSort eljárás kis módosítással könnyen megírható:

Write(gy(t)) ⟶ Write(A[1])



Megjegyezzük, hogy ha programot írnánk, akkor a KTour és az MTour eljáráshívásokat célszerű lenne paraméter nélkül megírni. Itt inkább csak formai okból szerepel a két eljáráshívásban paraméter, ugyanis amikor előzőleg létrehoztuk az eljárásokat, paraméterként adtuk meg az *A* tömböt.