15. A VERSENYRENDEZÉS

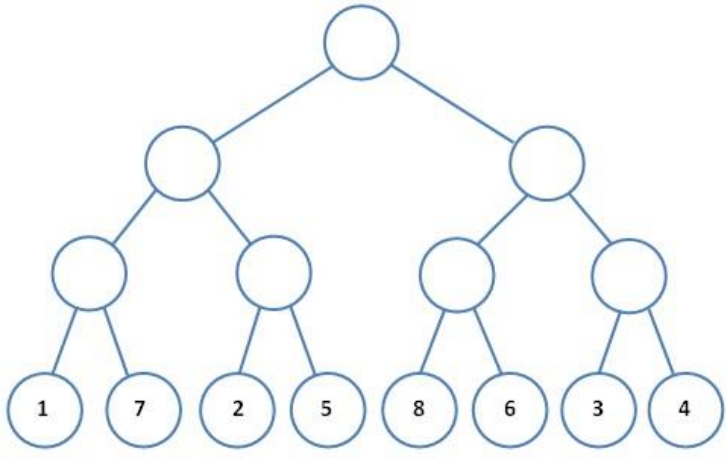
A *versenyrendezés* (*tournament sort*) a *maximum-kiválasztó* rendezések közé tartozik, ugyanis az elemek közül újra és újra kiválasztja (eltávolítja és kiírja) a legnagyobbat. Az eljárás így, a kiírás sorrendjében, nagyság szerint csökkenő rendezést valósít meg. A maximum kiválasztásban jól ismert gyakorlati háttérre támaszkodik: a *kieséses sportversenyek* lebonyolítási rendjét követve határozza meg az elemek között a „győztest”.

A módszert $n = 2^k$ elemszámra ismertetjük, ahogyan egy teniszverseny játéktáblájára is 2-*hatvány* számú versenyzőt írnak fel. Ahogyan a versenyen is megoldják, hogy ha a résztvevők száma nem ennyi, úgy a versenyrendezést is ki lehet terjeszteni tetszőleges n input méretre. A kiterjesztésre azonban nincs gyakorlati igény, mert egy másik, ehhez nagyon hasonló rendező eljárás, a kupacrendezés több szempontból kedvezőbb tulajdonságú, így előnyt élvez a gyakorlati felhasználásban.

A versenyrendezést több szempontból is érdemes megismerni. Egyszerűsége könnyen érthetővé teszi, amihez hozzájárul a kieséses sportverseny, mint szemléletes háttér is. Az eljárás jól reprezentálja mind a maximum-kiválasztó, mind az $n \log n$-es műveletigényű rendezéseket. Az algoritmus és a használt adatszerkezet, a versenyfa (tournament) alkalmas arra, hogy látványos különbséget tegyünk az *absztrakt adatszerkezet* (teljes bináris fa) és az (tömbös) *ábrázolás* szintje között.

15.1. A versenyrendezés módszere

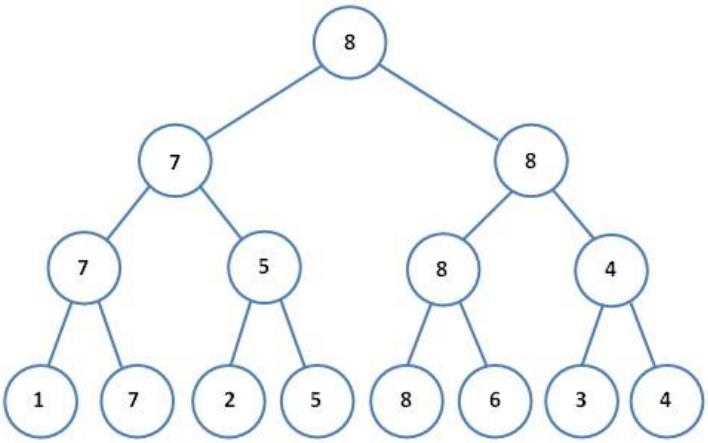
A versenyrendezés a *versenyfa* (tournament) adatszerkezetet használja. A versenyfa olyan *teljes bináris fa*, amelynek a *leveleiben* helyezkednek el a *rendezendő elemek*. A 15.1. ábrán egy kitöltött levelekkel rendelkező, de a belső pontjaiban még kitöltetlen versenyfát láthatunk.



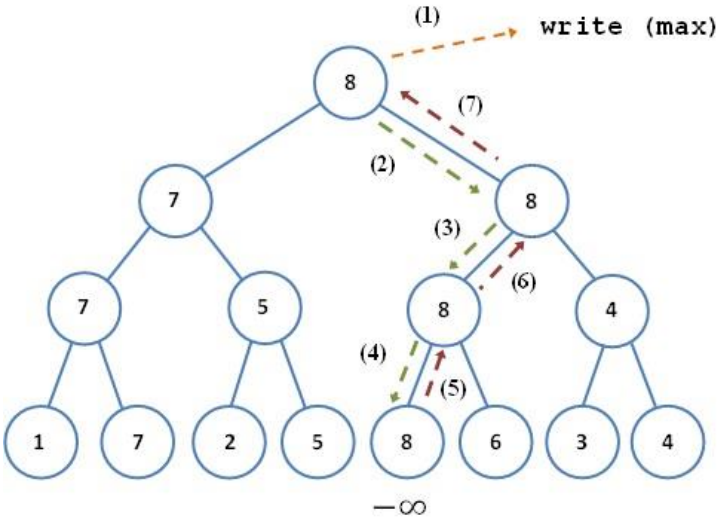
15.1. ábra. Versenyfa, leveleiben a rendezendő számokkal

A fa *belső pontjait* szintenként úgy töltjük ki, ahogyan egy kieséses verseny halad előre: minden belső pontban a gyerekei közül a *nagyobbnak* az értéke kerül (mint egy mérkőzés nyertese). Végül, a maximum (az abszolút győztes) a fa *gyökerébe* kerül. A kitöltött versenyfát a 15.2. ábra szemlélteti, $n = 8$ rendezendő szám esetére.

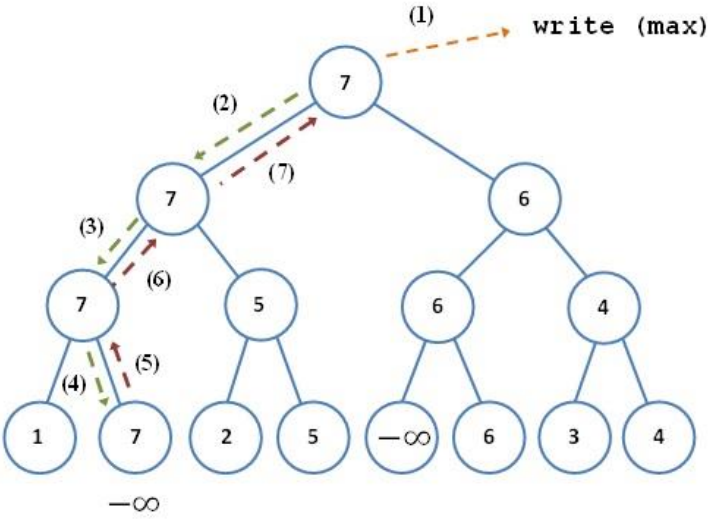
A rendezés egy speciális első menetet, majd azután még $(n - 1)$ egyszerűbb iterációs lépést hajt végre. Az *első menetben* – a versenyfa imént leírt kitöltésével – kiválasztjuk a *legnagyobb elemet*, amely „felkerül” a fa gyökerébe. Vegyük hozzá ehhez még a legnagyobb elem kiírását is, a rendezés kimenetére. Ezt már a következő 15.3. ábra tünteti fel.



15.2. ábra. Kitöltött versenyfa



15.3. ábra. Az első „újrajátszás” a győztes ágán



15.4. ábra. A második „újrajátszás” a következő győztes ágán

A versenyfa kitöltését követően kerül sor $(n-1)$ egyszerűbb menetre, a *következő legnagyobb elem kiválasztására*. Példánkban ez $8-1=7$ iterációt jelent, amelyek közül az első kettő lépéseit mutatja be a 15.3. és a 15.4. ábra.

Miután a gyökérben megjelenő legnagyobb elemet kiírtuk, „lefelé” haladva *megkeressük* azt a *levelet,* amelyben eredetileg ez az érték helyet foglal. A levélben található értéket $(-\infty)$-re cseréljük, azaz egy *abszolút vesztest* teszünk a helyére. Utána „felfelé” haladva ezen az ágon „*újrajátsszuk a mérkőzéseket*”. Ezeket a lépéseket sorszámozott formában láthatjuk a 15.3. ábrán.

Az újrajátszás eredményét már a következő 15.4. ábra tünteti fel. A *második legnagyobb* elem megjelent a gyökérben. Ez által azt is megtudjuk, hogy „ki nyert volna, ha a győztes nem indult volna a versenyen”. Ezen az ábrán bejelöltük a második legnagyobb elem levélszintű helyének a megkeresését és az ezt követő újrajátszás útvonalát.

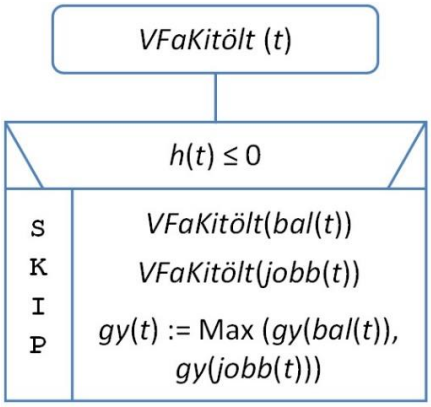
Így haladunk tovább, minden menetben a következő legnagyobb elemet megkeresve és kiírva. Mivel az elemeket csökkenő sorrendben vettük ki, az eljárás alkalmas a *rendezés* megvalósítására.

15.2. Rekurzív algoritmus az absztrakt adatszerkezet szintjén

Először *ADS szinten, rekurzív* módon írjuk le az algoritmust, mert az jobban illeszkedik az eddigi szemléletes meggondolásokhoz, könnyebben tudjuk algoritmus formájában rögzíteni a bináris fán leírt működést. (Később leírjuk azt a változatot is, amely a versenyfa tömbben tárolt változatán működik.)

15.2.1. A versenyfa kezdeti kitöltése

Tételezzük fel, hogy a t fa *levelei* már tartalmazzák a *rendezendő adatsorozatot.* Ahhoz, hogy ezt a teljes bináris fát versenyfává tegyük, először töltsük ki a *baloldali részfáját*, majd a *jobboldalit* is. Ha már a bal és jobb részfa egyaránt kitöltött versenyfa, akkor mindössze annyi a dolgunk, hogy a *fa gyökerébe* beírjuk a bal és jobb részfa gyökereiben lévő értékek *maximumát.* A *rekurzióból* való „*kijáratot*” az biztosítja, hogy ha a t fa magassága 0, azaz *egyelemű* fáról van szó, akkor már nem kell tennünk semmit. A versenyfa kitöltésének algoritmusát a 15.5. ábrán adtuk meg.



15.5. ábra. A versenyfa kezdeti kitöltése (rekurzív)

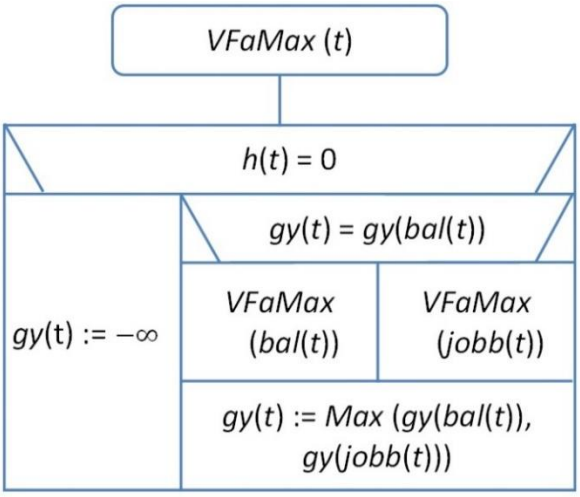
15.2.2. A következő legnagyobb elem kiválasztása

A következő maximum kiválasztása érdekében a mérkőzéseket *újrajátsszuk* a *győztes ágán*. Ezt is *rekurzív* módon hajtjuk végre. Ha t az egy elemű fa, akkor azt jelenti, hogy a rekurzív hívások láncolatában elérkeztünk az aktuális „győztes” *eredeti* helyéhez. *Állítsuk át* annak értékét $(-\infty)$-re!

Egy magasabb (nagyobb méretű) fában az újrajátszás rekurzív szemlélettel úgy fogalmazható meg, hogy

(i) ha a gyökérelem (azaz a győztes) a baloldalról származik, akkor a baloldali részfa egy ágát játsszuk újra, ha a jobboldalról, akkor a jobboldaliban tesszük ugyanezt;

(ii) miután a megfelelő oldalon elvégeztük az újrajátszást, újból ki kell számítanunk a fa gyökérelemének értékét, ami a következő maximummal lesz egyenlő.

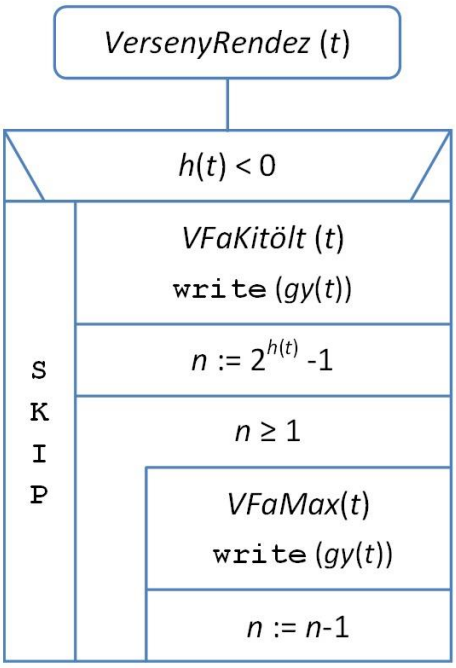


15.6. ábra. A győztes ág újrajátszása (rekurzív)

A rekurzív algoritmus a 15.6. ábrán látható. A rekurzív hívások láncolatának végrehajtása azt eredményezi, hogy az egymásba ágyazott *hívások* során az eljárás balra-jobbra tájékozódva eljut a maximális elem *levélszintű* helyéhez és azt $(-\infty)$-re állítja. Majd a hívásokból való *visszatérések* során történik a tulajdonképpeni *újrajátszás* a győztes ágán, amely elvezet a *következő legnagyobb elem* kiválasztásához a gyökérben.

15.2.3. A versenyrendezés rekurzív algoritmusa

A versenyrendezés a két létrehozott rekurzív eljárás felhasználásával, a bevezetőben leírt módon működik. Az algoritmust a 15.7. ábrán adtuk meg.



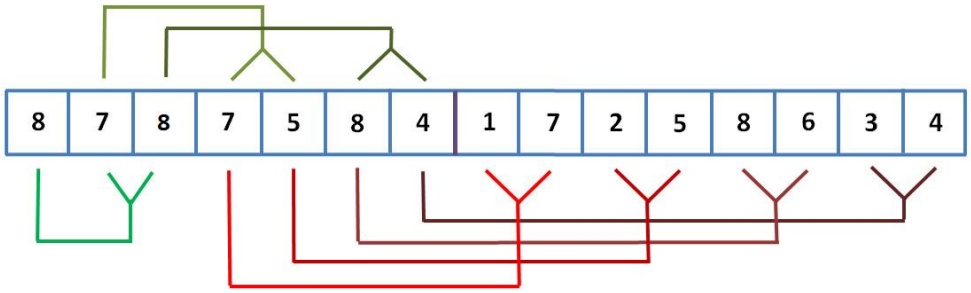
15.7. ábra. A versenyrendezés rekurzív megvalósítása

Tegyük fel, hogy a t fa *levelei* tartalmazzák a *rendezendő* sorozatunkat. *Egyetlen elem* rendezése nem kíván tevékenységet. *Több elem* esetén először *kitöltjük* a t rendezőfát, majd *kiírjuk* a maximumát. Ezután $(n-1)$-szer végre kell hajtanunk az a legnagyobb elem (a győztes) ágán az *újrajátszását*, amivel megvalósítjuk az egyre *csökkenő maximumok* egymás utáni kiválasztását, vagyis magát a *rendezést*.

15.3. Iteratív megvalósítás a reprezentáció szintjén

A teljes bináris fa egyszerű és helytakarékos megvalósítása az, ha *tömbben* tároljuk. Az $n = 2^k$ levelű fának $(2n-1)$ pontja van, tehát egy $(2n-1)$ hosszú tömbben tárolható. A tömb elemei felelnek meg a fa csúcsainak.

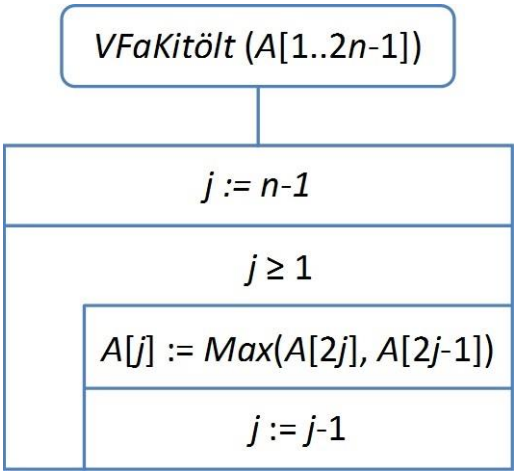
A tömb i-edik elemének *balgyereke* a $2i$-edik elem, míg *jobbgyereke* a $(2i+1)$ indexű helyen található. Ha $i \neq 1$, akkor az i-edik elem *szülője* az $\left\lfloor \frac{i}{2} \right\rfloor$ indexszel érhető el. A példánkban szereplő *versenyfa tömbös tárolását* és a *szülő-gyerek kapcsolatokat* a 15.7. ábra szemlélteti.



15.8. ábra. Versenyfa tárolása tömbben

15.3.1. A versenyfa kezdeti kitöltése

Ha feltételezzük, hogy az A tömb már *tartalmazza* az n számú *rendezendő elemet* az n-edik pozíciótól kezdve a $(2n-1)$-edik helyig, akkor a tömb *megelőző* elemei, amelyek a fa *belső pontjait* reprezentálják, egyszerű iterációval kitölthetők. A $(2n-1)$-edik tömbelemtől kezdve az első elemig jobbról balra lépegetünk, és minden pozícióba beírjuk a (bináris fában értelmezett) bal- és a jobbgyerek értékei közül a *nagyobbat*. A fa $A[1]$-es *gyökerébe*, a *legnagyobb* érték kerül. A *versenyfa kitöltésének* algoritmusát a 15.9. ábrán adtuk meg.



15.9. ábra. Az versenyfa kitöltése (iteratív)

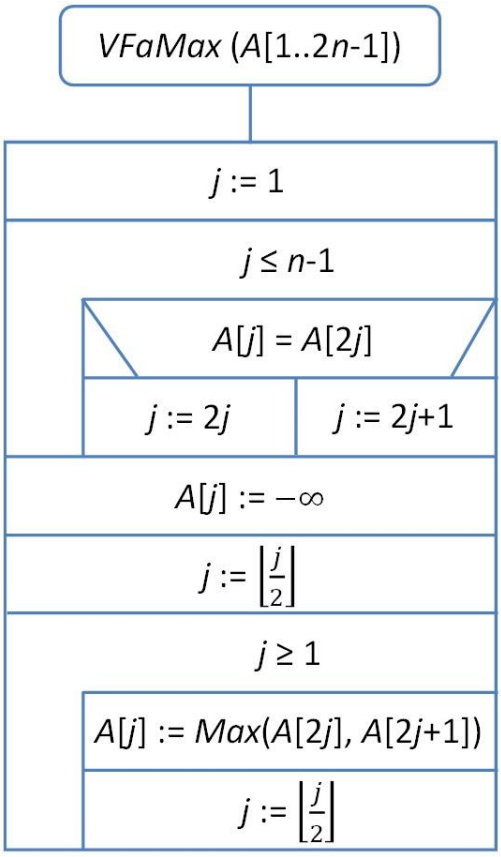
15.3.2. A következő legnagyobb elem kiválasztása

A *következő maximum* meghatározásának céljából *két* nagyobb *iteratív* módon szervezett *lépést* hajtunk végre.

Az *első* ciklikusan szervezett tevékenység során végighaladunk a gyökértől kezdve az aktuális győztest tartalmazó levélig. Ezután a megtalált levél értékét $(-\infty)$-re állítjuk.

Az eljárás *második* iteratív lépéssorozatában visszafelé haladunk, a most átírt elemtől végig a gyökérig, és minden érintett tömbelem értékét a (bináris fában értelmezett) gyerekei maximumára frissítjük.

A $j := \left\lfloor \frac{j}{2} \right\rfloor$ utasítás az jelenti, hogy a j indexű tömbelemről, amely a versenyfa egy csúcsát tárolja, annak *szülőjére* ugrunk. Onnan azután majd „visszanyúlunk” a két gyerekcsúcs tömbbeli helyére, amikor kitöltjük az értékét. A *következő maximum kiválasztásának* algoritmusát a 15.10. ábra tartalmazza.

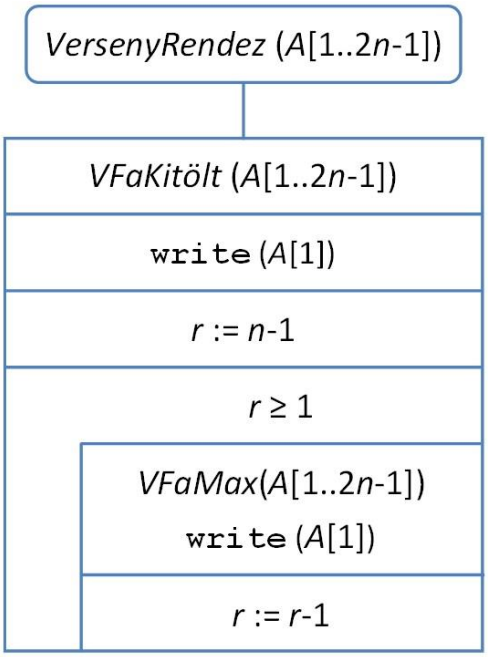


15.10. ábra. A győztes ág újrajátszása (iteratív)

15.3.3. A versenyrendezés iteratív algoritmusa

A versenyrendezés tömbös algoritmusában feltételezzük, hogy az A tömb kezdetben *tartalmazza* az n számú *rendezendő elemet*, az n-edik helytől kezdve a $(2n-1)$ indexű utolsó elemig. A rendezés – a fenti leírásnak megfelelően – a következő nagyobb lépéseket végzi, az imént létrehozott két eljárás felhasználásával, a 15.11. ábrán látható módon.

Először meghívja a *versenyfa kitöltését* végző tömbös eljárást, majd kiírja a tömb első elemét, amely a fa gyökérnek, vagyis a maximális elemnek felel meg. Ezután $(n-1)$-szer végrehajtja a *következő maximum kiválasztását* végző eljárást, amely mindannyiszor „felhozza” a tömb első elemébe az *aktuális győztest*. Ezt mindig az elem *kiírása* követi.



15.11. ábra. A versenyrendezés iteratív algoritmusa

15.4. A versenyrendezés műveletigénye

Az algoritmus *speciális első menete*, a versenyfa kitöltése $(n-1)$ *összehasonlítást* alkalmaz, hiszen a versenyfának $(n-1)$ belső pontja van, és mindegyiket egy-egy összehasonlítás segítségével töltjük ki. A kitöltés ugyanennyi *mozgatást* is igénybe vesz.

A versenyfa, mint teljes bináris fa *magassága* $\log_2 n$, ahol n, a levelek száma, egy 2-*hatvány*. A további menetek $2\log_2 n$ összehasonlítást használnak, mivel *kétszer* kell a fát *teljes magasságában* „átszelni", egyszer a maximális elemhez tartozó *levél* megtalálásához, azután pedig egy ág *újrajátszása* során. Elemmozgatás csak a második, újrajátszási lépéssorozathoz tartozik.

Összefoglalva:

$$Ö(n) = n - 1 + (n-1)\cdot 2\log_2 n = \theta(n\log n)$$

$$M(n) = n - 1 + (n-1)\cdot \log_2 n = \theta(n\log n)$$

Az versenyrendezés egyetlen jelentős hátránya a *tárigénye*. *Nem helyben* rendez, hanem az elemeket tartalmazó n mezőn kívül még $(n-1)$-re van szüksége a fa belső csúcsai számára, és még n-re ahová kiírjuk a rendezett elemeket. A körülbelül *háromszoros helyigény* miatt nem alkalmazzák a gyakorlatban az algoritmust.

Ezt a döntést az a körülmény is megkönnyíti, hogy ugyanebben a „műfajban" ($n\log n$-es maximum-kiválasztó rendezés) hatékony *helyben* dolgozó eljárás, a *kupacrendezés* áll rendelkezésünkre.