10. Szimultán kiválasztások

Ebben a fejezetben két újabb alsókorlát-elemzés következik. Mindkettőben egy szimultán algoritmus műveletigényére adunk alsó korlátot. Pontosabban, egy-egy feladat megoldásához minimálisan szükséges lépésszámot határozzuk meg. Ezek az ötletes és szép meggondolásokat hozzá tartoznak az algoritmuselmélet színes világához, egyébként is, az algoritmusok lépésszámára ritkán állapítható meg alsókorlát, így ezen a téren minden eredményt „meg kell becsülni”.

10.1 Szimultán minimum-maximum kiválasztás

Specifikáljuk azt a feladatot, amely egy s sorozat minimális és maximális elemének a meghatározását tűzi ki célul.

Feladat. Valamely $s \in X^*$ sorozat minimális és maximális elemének megkeresése.

Deklaráció: $s, s' \in X^*, \;\; x, y \in X$

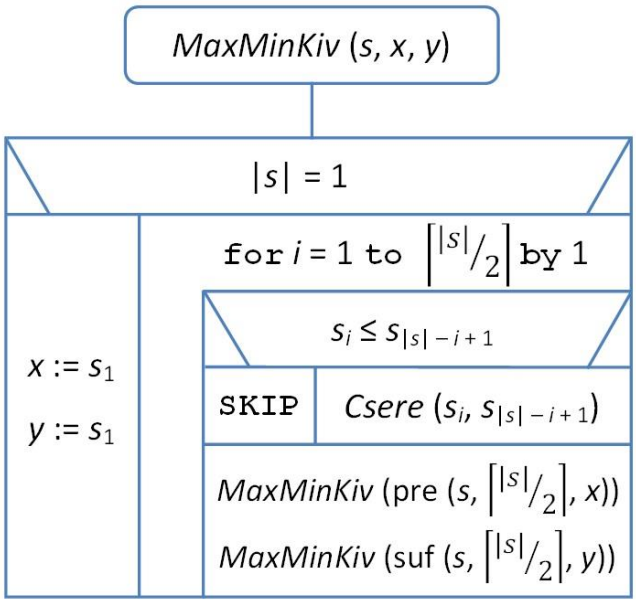
Előfeltétel: $s = s' \wedge s \neq \varepsilon$

Utófeltétel: $s = Perm(s')$ és x, y az s' maximális, illetve minimális eleme

Természetes ötlet, hogy válasszuk ki a sorozat maximumát, aztán a többi elem közül a minimumot. Ennek az algoritmusnak a műveleti igénye az összehasonlítások számával mérve $n - 1 + n - 2 = 2n - 3$. Látszólag nem lehet jobb algoritmust készíteni, hiszen a maximum kiválasztáshoz n elem esetén mindig szükséges legalább $n - 1$ összehasonlítás, majd a maradék $n - 1$ elem közül legalább $n - 2$ összehasonlítással tudjuk csak kiválasztani a minimumot.

Hatékonyabb szimultán minimum-maximum kiválasztó algoritmushoz juthatunk úgy, ha bizonyos összehasonlítások eredményét (lehetőleg minél többet) a maximum és a minimum kiválasztása során egyaránt felhasználunk.

Ezt a gondolatot felhasználva a következő $MaxMinKiv$ nevű absztrakt algoritmust készíthetjük (lásd: 10.1. ábra).



10.1. ábra. A *MaxMinKiv* algoritmus

Párokba rendezzük s elemeit, és minden páron elvégezzük az összehasonlítást (ha n páratlan, akkor ebből egy elem kimarad). Ezek után a győztesek közül – hozzávéve az esetleg kimaradó elemet – kiválasztjuk a legnagyobbat, és hasonlóan járunk el a vesztesekkel is, a minimum megtalálására.

Elemzés. A párok száma $\left\lfloor \frac{n}{2} \right\rfloor$, innen adódik $\left\lfloor \frac{n}{2} \right\rfloor$ összehasonlítás. Az esetleg hozzájuk vett elemmel együtt a nagyok és a kicsik száma egyaránt $\left\lceil \frac{n}{2} \right\rceil$, amiből összesen $2\left(\left\lceil \frac{n}{2} \right\rceil - 1\right)$ összehasonlítás adódik. Könnyen beláthatjuk, hogy ezek összege $\left\lfloor \frac{3(n-1)}{2} \right\rfloor$.

Ha *n* páros, azaz $n = 2m$ esetén, az összeg $3m - 2$ és $\left\lfloor \frac{3(2m-1)}{2} \right\rfloor = 3m - \left\lceil \frac{3}{2} \right\rceil = 3m - 2.$

Ha n páratlan, vagyis $n = 2m + 1$ esetén, az összeg $3m$ és $\left\lfloor \frac{3(2m)}{2} \right\rfloor = 3m.$

A szimultán eljárás által végrehajtott összehasonlítások száma: $\text{Ö}_{MaxMinKiv}(s) = \left\lfloor \frac{3(n-1)}{2} \right\rfloor$.

Azt kaptuk, hogy a $MaxMinKiv$ algoritmus ¾ részére csökkenti az összehasonlítások számát. Felmerül ezek után a kérdés, hogy vajon lehet-e szükséges a lépésszámon tovább javítani. A válasz tagadó, ugyanis fennáll a következő tétel.

Tétel. Legyen $MaxMinKiv$ egy minden bemenetre jól működő összehasonlításos szimultán maximum-minimum kiválasztó algoritmus. Ekkor $M\text{Ö}_{MaxMinKiv}(n) \geq \left\lfloor \frac{3(n-1)}{2} \right\rfloor$.

Más szóval, létezik olyan n hosszúságú $s \in X^*$ sorozat, melyre $MaxMinKiv$-nek a maximum és minimum együttes megtalálásához legalább $\left\lfloor \frac{3(n-1)}{2} \right\rfloor$ összehasonlítást kell végeznie.

Bizonyítás. Az s sorozatot mintegy „menet közben” konstruáljuk meg, a kérdező által feltett kérdések tisztázó hatásának a késleltetése céljából. A válaszoló nem rögzíti tehát előre s elemeit, hanem az előző fejezetben látott „furfangos válaszoló módszerével” eljárva a kérdező, vagyis algoritmus stratégiájához idomulva, próbálja a lehető legtöbb összehasonlításra késztetni a megoldó eljárást.

A kérdező akkor lehet biztos a dolgában, ha talált egy olyan elemet, mely minden összehasonlításban győztes volt (ez a maximum), valamint egy olyat, amelyik mindig vesztes volt (ez a minimum); a többiek lehetnek vegyesen győztesek és vesztesek is.

Az összehasonlításokból adódó „győztes-vesztes” információ nyilvántartására a kérdező egy n elemű táblázatot használ, amelynek *i*-edik elemébe bejegyzi s_i sorozatelem státuszát. A táblázat folyamatosan módosul; a bejegyzések végső állapotából már kiolvasható az elemek viselkedése a kérdések (az algoritmus működése) során.

Az elemek státuszai a következő lehetőségek közül kerülnek ki:

- N: nem volt még rá vonatkozó kérdés;
- L: szerepelt kérdésben és minden összehasonlításban vesztes volt;
- W: szerepelt kérdésben és minden összehasonlításban győztes volt;
- LW: vonatkoztak rá kérdések, és az összehasonlításokban volt győztes és vesztes is.

Az algoritmusnak addig kell kérdeznie, amíg a táblázatban megjelenik $n - 2$ számú (L, W) bejegyzésű, egy L státuszú és egy W státuszú elem. Ez összesen $2 * (n - 1)$ információ-darabka. Egy kérdésre adott válaszból 0, 1 vagy 2 információelem nyerhető ki. Egy kérdés legfeljebb egy győztest és egy vesztest eredményezhet, de ha valamelyikükre már vonatkozik hasonló bejegyzés, akkor ott új információ nem keletkezik.

A válaszoló konzisztencia táblája lényegében ugyanaz, mint a kérdező táblázata, csak ebben benne vannak – számára láthatók – a konstrukciónak megfelelő értékek is. A válaszoló legrosszabb esetet konstruáló stratégiája olyan, hogy csak akkor ad információt, ha feltétlenül szükséges. Ha a kérdezett pár

1. mindkét tagja (N, N) bejegyzésű, akkor mindkét elemet rögzíti, az egyik elemet kisebb értékkel L-esként, a másikat nagyobb értékkel W-vel (2 információ);
2. egyik tagja N státuszú és a másik státusza
   2.1. L, akkor az N-es elemet W bejegyzésével, a másiknál nagyobb értékkel rögzíti (1 információ);
   2.2. W vagy LW akkor az N-es elemet L bejegyzéssel a másiknál kisebb értékkel rögzíti (1 információ);
3. Ha a pár egyik tagja L státuszú és a másik státusza
   3.1. W vagy LW, akkor az L-es elem bejegyzését megtartva, annak értékét megfelelően csökkenti (0 információ);
   3.2. L, akkor a rögzítésnek megfelelő választ adja, a nagyobb értékű elem bejegyzését LW-re állítva (1 információ);
4. Ha a pár egyik tagja W státuszú és a másik státusza
   4.1. LW, akkor W-es elem bejegyzését megtartva, annak értékét megfelelően megnöveli (0 információ)
   4.2. W, akkor a rögzítésnek megfelelő választ adja, a kisebb értékű elem bejegyzését LW-re állítva (1 információ);
5. Ha a pár mindegyik tagja LW státuszú, akkor a rögzítésnek megfelelő választ adja (0 információ).

Ez a stratégia nem vezethet ellentmondáshoz, hiszen az értékek rögzítésénél a válaszoló megfelelő bejegyzéseket teszi, míg egy – már rögzített – elem értékét csak abba az irányba változtatja, amilyen annak bejegyzése.

Bármely bemenetre az a legjobb kérdező stratégia, hogy maximális számú, két információt adó kérdést tesz fel, majd amikor azok elfogytak, akkor csupa egy információt eredményező kérdést fogalmaz meg. Az így adódó kérdésszám lesz a keresett alsó korlát.

A két információt adó kérdések az (N, N) alakú párokhoz tartoznak, ilyenből legfeljebb $\lfloor n/2 \rfloor$-t lehet feltenni. A többi $2(n-1) - 2\lfloor n/2 \rfloor$ információ megszerzéséhez legalább ennyi kérdés kell. Emiatt a kérdések száma a legcélratörőbb esetben is legalább

$$Ö(n) = \lfloor n/2 \rfloor + 2(n-1) - 2\lfloor n/2 \rfloor = 2(n-1) - \lfloor n/2 \rfloor.$$

Tudjuk, hogy m pozitív egészre fennáll: $\lfloor m/2 \rfloor + \lceil m/2 \rceil$ és $\lfloor m/2 \rfloor = \lceil (m-1)/2 \rceil$. Innen

$$Ö(n) = n - 1 + \lfloor (n-1)/2 \rfloor + \lceil (n-1)/2 \rceil - \lceil (n-1)/2 \rceil = \lfloor 3\,(n-1)/2 \rfloor.$$

Ezzel a bizonyítást befejeztük.

10.2 Az r legnagyobb elem rendezett kiválasztása

Specifikáljuk azt a feladatot, amely egy n elemű sorozatra, növekvő sorrendben rendre az $(n-r+1)$-edik, $(n-r+2)$-edik, .., n-edik elem megadását tűzi ki.

Feladat. Valamely $s \in X^*$ sorozat r legnagyobb elemének rendezett kiválasztása

Deklaráció: $s, s', u \in X^*$

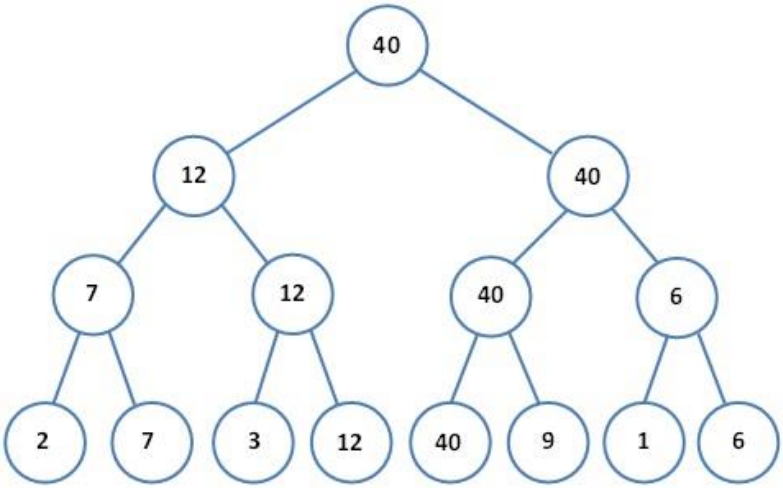
Előfeltétel: $s = s' \wedge s \neq \varepsilon \wedge 1 \leq r \leq |s|$

Utófeltétel: $s = Perm(s')$ és u az s rendezettjének r hosszú suffixe

Ez a feladat első pillantásra egyszerűen megoldható. Először kiválasztjuk a maximális elemet és kiírjuk u-ba. Ezek után a maradékból is kiválasztjuk a legnagyobbat, és kiírjuk u–ba az előző maximum elé, és így tovább r-szer. Ehhez szükséges rendre $n-1, n-2, \ldots n-r$ összehasonlítás, melyek összege $r(n-\frac{r+1}{2})$.

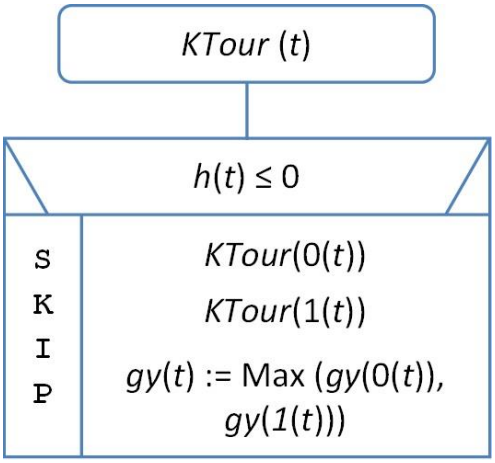
Vajon lehet-e ennél kevesebb összehasonlítással működő algoritmust találni erre a feladatra? Látszólag nem, hiszen a maximumhoz kell legalább $n-1$ összehasonlítás, a második maximumhoz maradék $n-2$ összehasonlítás, és így tovább. Ha azonban felhasználnánk az első maximum kiválasztásánál, a maradék $n-1$ elemen végzett összehasonlítások eredményét, akkor ezek számával csökkenthető a második legnagyobb elem kiválasztásához szükséges összehasonlítások száma. Ehhez meg kell jegyezni a maximum kiválasztásához elvégzett összehasonlításokat és azok eredményét. Az információ tárolásához az úgynevezett tournament (versenyfa) adatszerkezetet fogjuk használni. A tournament a nevét a kieséses sportversenyek eredménytáblázata nyomán kapta.

Egy t tournament olyan teljes bináris fa, amelynek leveleiben egy $n = 2^m$ (ahol m a fa magassága) hosszú sorozat helyezkedik el, belső pontjaiban pedig a két gyerekcsúcs értékének maximuma található (lásd: 10.2. ábra).



10.2. ábra. Egy kitöltött tournament

A t tournament gyökerében mindig a leveleinek maximuma áll. Nevezzük tournament váznak az olyan teljes bináris fát, melynek csak a leveleiben találhatók értékek. Egy ilyen vázat touranmentté alakító eljárás nyilván maga is maximum kiválasztó algoritmus lesz.

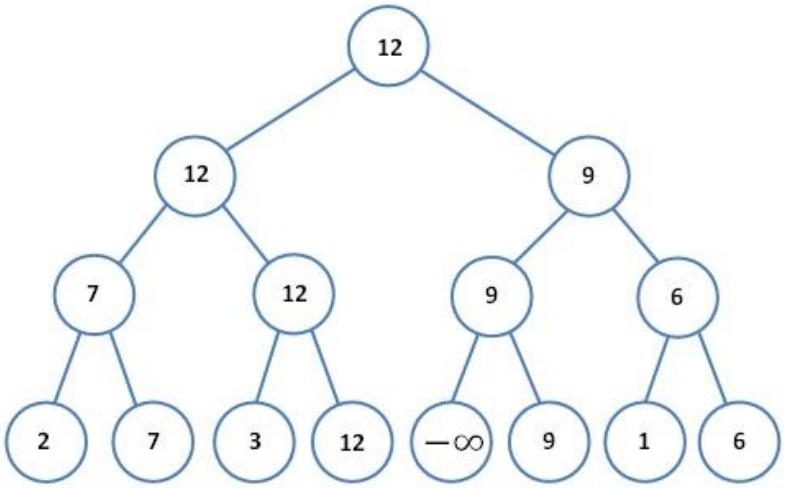


10.3. ábra. A *KTour* eljárás

Egy t fa akkor és csak akkor tournament, ha a baloldala és jobboldala eggyel kisebb magasságú tournamentek, és a gyökerében a bal és jobb oldali részfa gyökerének maximuma található. Ezt a tulajdonságot felhasználva könnyen készíthetünk olyan rekurzív eljárást, amely egy vázat átalakít tournamentté (lásd: 10.3. ábra).

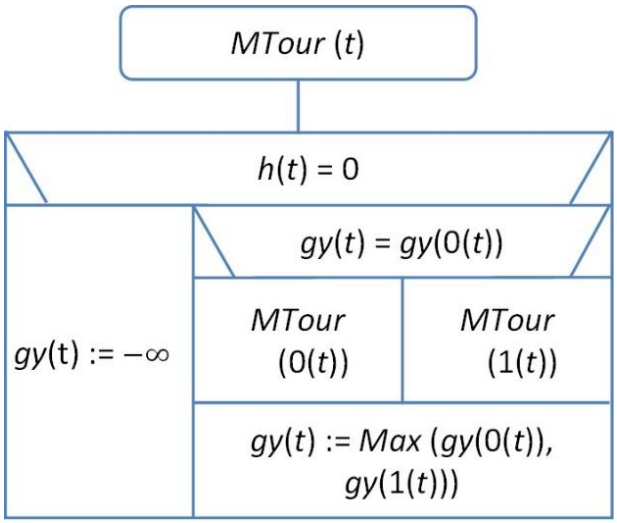
Elemzés. A $KTour$ eljárás összehasonlításainak száma pontosan a fa belső pontjaninak száma. A belső pontokat könnyen számba vehetjük. A levelek fölötti szinten 2^{m-1} belső pont van, a felette lévőn 2^{m-2}, és így tovább a gyökérig. Ezek m elemű mértani sorozatot alkotnak, melynek összege $2^m - 1$, tehát $n - 1$. Ezek szerint a $KTour$ optimális maximum kiválasztást megvalósító algoritmus.

A második legnagyobb elemet (majd pedig a továbbiakat) úgy lehet meghatározni, hogy a mindenkori maximális elemet – az u sorozatba való kiírása után – a lehető legkisebb elemmel, vagy egy extremális elemmel helyettesítjük. Vezessük be erre a célra a $-\infty$ szimbólumot. A helyettesítés után ez még nem tournament, de a maximum ágának újraszámolásával könnyen azzá alakítható (lásd: 10.4. ábra).



10.4. ábra. Tournament a maximum ág újrarajzolásával

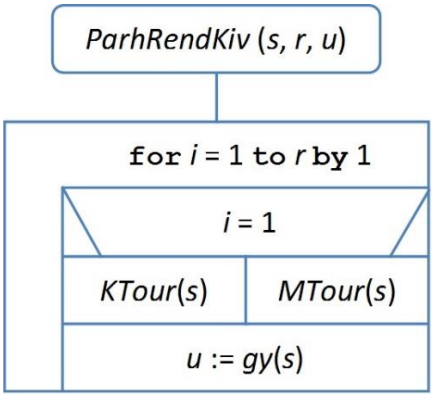
A további maximumokat kiválasztó $MTour$ eljárás is felhasználja a tournamentek rekurzív felépítését, valamint azt is, hogy csak az eddigi maximumot tartalmazó ágat kell újra számolni (hiszen a másik fele változtatás nélkül megmaradt tournamentnek).



10.5. ábra. Az *MTour* eljárás

Elemzés. Az első fordulós eredményt az eddigi maximum ágán már tudjuk is, hiszen a $-\infty$ ellenfele biztos győztes. Ennek megfelelően a második (és minden további) maximum kiválasztása az $MTour$ eljárással $m-1=\log_2 n-1$ összehasonlítással történhet meg.

Legyen $PárhRendKiv(s,r,u)$ most már az az eljárás, mely $KTour$ egyszeri, majd az $MTour$ eljárás $(r-1)$-szeri alkalmazásával rendezetten kiválasztja az s sorozat r számú legnagyobb elemét.



10.6. ábra. A *ParhRendKiv* eljárás

Elemzés. Az *r* számú legnagyobb elemet rendezetten kiválasztó $PárhRendKiv$ eljárás (lásd: 10.6. ábra) összehasonlításainak számát $KTour$ és $MTour$ összehasonlítás száma alapján könnyen becsülhető: $Ö_{PárhRendKiv}(t) \leq n-1+(r-1)(\lg(n)-1)$

Ha n nem kettő hatvány, akkor kiegészítjük a megfelelő $q=2^{\lceil \lg(n) \rceil}-n$ számú $(-\infty)$ extremális értékkel kettő hatvánnyá; mindet egy-egy valódi elemmel párosítjuk, amelyek így „erőnyerők" lesznek. Az erőnyerők összehasonlítás nélkül jutnak be a következő fordulóba és így az összehasonlítások száma q–val kevesebb a $(2^{\lceil \lg(n) \rceil}-1)$ lépésszámnál. Innen $KTour$ összehasonlítás-száma, nem kettő hatványok esetében $(2^{\lceil \lg(n) \rceil}-1)-q=n-1$, tehát optimális algoritmus marad.

Az $MTour$ eljárás most $\lceil \lg(n) \rceil - 1$ összehasonlítással működik. Emiatt nem kettő-hatványokra a $PárhRendKiv(s,r)$ műveletigénye az $n-1+(r-1)(\lceil \lg(n) \rceil - 1)$ kifejezéssel becsülhető felülről. Kettő hatványokra innen kiadódnak speciális esetként a rájuk vonatkozó képletek.

Az r elem rendezett kiválasztásának eljárását $r=n$ mellett alkalmazva, az outputon megjelenik a levelekben elhelyezet teljes kiinduló sorozat rendezettje. A $PárhRendKiv(s,n)$ algoritmus tehát egyben egy rendezési módszer is, melyet tournament rendezésnek (versenyrendezésnek) nevezünk. (Erre a rendező eljárásra visszatérünk a 15. fejezetben.) A tournament rendezés műveletigénye az előzőek szerint $n-1+(n-1)\lceil \lg(n) \rceil$. Amint azt a rendezések ismertetése során látni fogjuk, az összehasonlítások nagyságrendje szempontjából ez optimális rendező módszer.

Arra a kérdésre, hogy lehet-e a $RendKiv(s,r)$ eljárásnál jobb algoritmust találni erre a feladatra, nemleges választ ad *Kiszlicin* tétele, melyet bizonyítás nélkül közlünk.

Tétel. Bármely olyan algoritmus, amely minden *n* hosszúságú bemenetre az r legnagyobb elemet rendezett sorrendben garantáltan megtalálja, mindig legalább

$$(n-r+\sum_{i=1}^{r-1}\lceil \lg(n-r) \rceil)$$

számú összehasonlítást végez.