9. KIVÁLASZTÁSOK

Az algoritmusok tervezése során igen gyakran találkozunk a legkisebb vagy a legnagyobb elem kiválasztásának feladatával. Az sem ritka, hogy az ezektől eltérő sorszámú k-adik elemet kell megkeresnünk. Ebben a fejezetben a kiválasztások feladatát nézzük meg közelebbről, és nem csak megoldó algoritmusokat adunk – köztük egy véletlenített eljárást is –, hanem néhány alsó korlátot is bizonyítunk a lehetséges megoldások lépésszámára.

9.1 A kiválasztás feladata és néhány alsó korlát

Ebben a fejezetben X legyen egy megszámlálható számosságú halmaz, melyen adott a $\leq$ teljes rendezés: reflexív, antiszimmetrikus, tranzitív és bármely két elem összehasonlítható. (Példáinkban X az egész számok halmaza lesz a szokásos rendezéssel.) Használjuk még az $x < y$ jelölést arra az esetre, amikor $x \leq y$ és $x \neq y$; ez egy úgynevezett erős rendezés. Jelölje X^* az X-ből képzett véges sorozatok halmazát.

Definíció: Egy $u \in X^*$-beli sorozat növekvően rendezett, ha $\forall \mathrm{i} \in [1 \ldots |u| - 1]$ esetén $(u_\mathrm{i} \leq u_{\mathrm{i}+1})$. Hasonlóan definiálhatjuk a csökkenően rendezett sorozat fogalmát is.

Megjegyzés. A továbbiakban rendezettség alatt mindig a növekvő rendezettséget értjük.

Definíció: Legyen $u \in X^*$ tetszőleges sorozat és $1 \leq k \leq |u|$ tetszőleges egész. Az $x \in X$ elemet az u sorozat k–adik elemének hívjuk, ha megegyezik az u rendezettjének k-adik elemével. Ha $k = 1$ akkor minimális, ha $k = |u|$ maximális elemről, ha $\mathrm{k} = \lceil |u|/2 \rceil$, akkor mediánról beszélünk.

Például az $u = 31, 45, 13, 24, 7, 12, 8$ sorozat minimuma 7, maximuma 45 és a mediánja 13 (a rendezett sorozat $\lceil 7/2 \rceil = 4$ indexű eleme).

A fejezet során azzal foglalkozunk, hogy milyen módon lehet megtalálni egy $s \in X^*$ sorozat k–adik elemét, illetve speciálisan a minimumát, maximumát és mediánját.

Algoritmusaink az s sorozat elemeit nem ismerik. Csak annyit tudnak tenni, hogy feltesznek $s_i \leq s_j$ alakú kérdéseket, és pusztán ezek eredménye alapján adják meg, hogy melyik az l index, melyre s_l a kívánt tulajdonságú elem. Gyakran az l index csak implicit módon van benne az algoritmus eredményében, a tényleges eredmény (s_l) egy x változóban áll elő. A fejezet során csak ilyen, úgynevezett összehasonlításos algoritmusokkal foglalkozunk.

A szemléletesség kedvéért az összehasonlításban szereplő elemekre a kieséses sportversenyekhez hasonló terminológiát használunk. Ha valamely $x \leq y$ alakú kérdésre a válasz „igen”, akkor x_i-t vesztesnek, míg y_j-t győztesnek nevezzük. Ha még azt is tudjuk, hogy $x \neq y$, akkor erős vesztesről, illetve erős győztesről beszélünk.

Legyen u és v két egyforma hosszú, X elemeiből képzett sorozat. Az u-t és v-t hasonlónak nevezzük, ha minden $1 \leq i, j \leq |u|$ párra $u_i \leq u_j$ akkor és csak akkor, ha $v_i \leq v_j$. Világos, hogy az összehasonlításos algoritmusok a hasonló sorozatokon egyformán működnek, emiatt ugyanazon kiválasztási eredményt is kell adniuk.

Mielőtt a megoldó algoritmusok elkészítésébe belefognánk, bizonyítunk két tételt arról, hogy legalább hány összehasonlítást kell tenni ahhoz, hogy egy sorozat maximumát, illetve a k–adik elemét megtaláljuk.

Tétel. Legyen *MaxKiv* egy minden bemenetre jól működő összehasonlításos maximumkereső eljárás. Ekkor ahhoz, hogy *MaxKiv* bármely n hosszúságú bemenetre a garantáltan megtalálja a legnagyobb elemet, mindig legalább $(n-1)$ összehasonlítást kell végeznie.

Természetesen hasonló tétel igaz a minimum kiválasztására is.

Bizonyítás. Az állítást teljes n-re vonatkozó indukcióval látjuk be.

Az $n = 1$ esetben nyilván nincs szükség összehasonlításra, hiszen a maximum megegyezik a sorozat egyetlen elemével.

Tegyük fel, hogy az állítás n-nél ($n \geq 2$) rövidebb sorozatok esetében igaz, és legyen s egy n hosszú sorozat. Vizsgáljuk először azt az esetet, amikor az összehasonlítások legalább egyikében volt erős vesztes. Legyen az első ilyen összehasonlítás $s_i \leq s_j$. Vesztesként s_j nem lehet a maximum, de nyilván azok az elemek sem, melyek a korábbi összehasonlítások alapján vesztők voltak s_j-vel szemben. Jelöljük az így kizárt elemek számát m-el (nyilván $m < n$). Világos, hogy ezen kizárt elemek mindegyikére van legalább egy olyan összehasonlítás, melyben ők vesztők voltak. $MaxKiv$ tehát legalább m összehasonlítást használ csak arra, hogy ezt az m elemet kizárja. A többi összehasonlítás a maradék $n-m$ elemből választja ki maximumot, amihez indukciós feltételünk alapján szükséges legalább $n-m-1$ darab összehasonlítás. Az összes összehasonlítások száma tehát legalább $m+n-m-1 = n-1$.

Térjünk rá arra az esetre, amikor egyik összehasonlítás sem eredményez vesztest. Tegyük fel, hogy $MaxKiv$ legfeljebb $n-2$ összehasonlítással helyesen meghatározta a maximumot; legyen ez $x = s_l$. Az $n-2$ összehasonlítás legfeljebb $n-2$ vesztest eredményezhet. Ezek közül legalább az egyik különbözik s_l-től, legyen s_r az egyik ilyen. Tekintsük most azt a z sorozatot, ahol minden elem egyenlő, kivéve z_r-t, mely határozottan nagyobb a többinél. Világos, hogy erre a módosított sorozatra az algoritmus pontosan ugyanazokat az összehasonlításos eredményeket adja, ezért most is az l-et indexűt kell megneveznie maximumként. Ez pedig nyilván nem helyes, hiszen a módosított sorozatban nem z_l, hanem z_r a maximum. ■

Tétel. Legyen Kiv egy minden bemenetre jól működő, összehasonlításos, k –adik elemet kiválasztó algoritmus. Ekkor $MÖ_{Kiv}(n) \geq n-1+\min(n-1, n-k)$.

Átfogalmazva, a tétel azt mondja ki, hogy van olyan n hosszúságú $s \in X^*$ sorozat, melyre Kiv a k-adik elemet legalább $n-1+\min(n-1, n-k)$ összehasonlítással tudja csak megtalálni.

Bizonyítás. Csak a csupa különböző elemet tartalmazó sorozatok között keresünk ilyen s-et. Ha a Kiv eljárás egy s sorozatban helyesen megnevezi $x = s_l$-et, mint az s sorozat k-adik elemét, akkor ez az elem a nála kisebb $(k-1)$ elemmel olyan k elemű halmazt alkot, melynek x a maximuma. Ehhez az előző tétel szerint szükséges közöttük legalább $k-1$ összehasonlítás. Hasonló gondolattal láthatjuk be, hogy x és a nála nagyobb elemek között lennie kell $n-k$ összehasonlításnak. Ez összesen $n-1$ összehasonlítás. ■

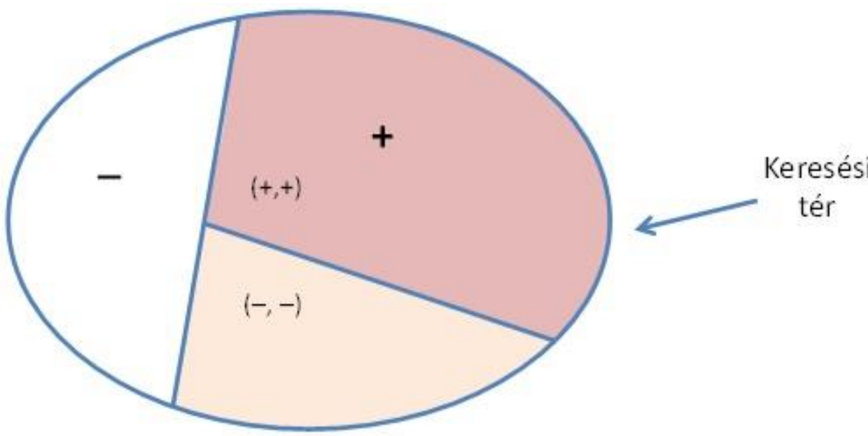
Nevezzük nem lényeginek az olyan összehasonlításokat, melyeket a *Kiv* algoritmus a kis és nagy elemek között végezett. Ezek továbbiakat jelentenek az előbbi $n-1$ összehasonlítás mellett. A kérdés az, hogy akármilyen kérdezési stratégia mellett van-e olyan n hosszú sorozat, mely esetében a kérdező nem lényegi kérdéseket is feltesz. Ha igen, akkor milyen kérdezési stratégia mellett lesz ezek száma a lehető legkisebb?

A kérdés megválaszolására az irodalomban „ellenfél-módszernek” nevezett technikát használják. Véleményünk szerint a „furfangos válaszoló” kifejezés jobban kifejezi a módszer lényegét, ezért a továbbiakban ezt elnevezést használjuk.

A technika és elnevezése a barkochba játékra vezethető vissza. Ennek lényegéhez tartozik az, hogy közösen rögzítenek egy véges halmazt, a keresési teret. Ezek után egyikük, a válaszoló kiválaszt ebből a halmazból egy elemet. A másik, a kérdező, igen-nem kérdésekkel megpróbálja ezt az elemet a lehető leghamarabb kitalálni. Egy kérdés az éppen aktuális keresési teret két részre bontja, és a válaszoló megmondja, a keresett elem melyik részben van. A kérdező ezt a részt tekintve aktuális keresési térnek ugyanígy halad tovább. Ha az aktuális keresési tér már csak egy elemű, akkor ezt az elemet adja válaszként.

A furfangos válaszoló a kérdezőt a lehető legtöbb kérdésre akarja kényszeríteni. Ebből a célból a játék elején nem rögzíti le előre a kitalálandó elemet, hanem a kérdező szempontjából lehető legkellemetlenebb módon folyamatosan változtatja. Ha valamely kérdés az aktuális keresési teret nem pontosan felezi, akkor furfangosan azt a választ adja, hogy a kitalált elem a nagyobbik részben van (persze ügyelnie kell válaszainak konzisztenciájára, különben a játék ellentmondásokhoz vezethet).

A 9.1. ábra a két kérdés utáni helyzetet mutatja, ahol a + jel a mindig a nagyobbik térfelet jelöli. A válaszoló a kitalálandó elemet előbb a + jelű, majd a (+, +)-os részbe helyezi át (ha eleve nem ott lett volna).



9.1. ábra. A keresési tér

A furfangos válaszoló a kérdező stratégiájának legrosszabb esetét konstruálja meg. A kérdező a furfangos válaszoló ellen úgy védekezhet, hogy lehetőleg mindig olyan kérdést tesz fel, amely az aktuális keresési teret felezi. Ez a kérdezőnek a legrosszabb esetre nézve optimális (minimális kérdésszámmal járó) stratégiája.

Mi köze van mindennek az összehasonlításos algoritmusokhoz? Minden összehasonlításos algoritmust tekinthetünk kérdezőként, míg a kérdésekre adott válaszokat interpretálhatjuk úgy is, hogy azokat egy, a kérdezőtől független válaszoló adja meg. Az, hogy a válaszoló furfangos módon minél több kérdésre kényszeríti a kérdezőt, azt jelenti, hogy az adott algoritmus számára megkonstruál egy rossz esetet. Ha a kérdező (az algoritmus) – tudva mindezt – a lehető legjobb (optimális) kérdezési stratégiát használja, akkor ennek legrosszabb esete az összes, a feladatot megoldó algoritmus legrosszabb esetére vonatkozó alsó korlát lesz.

Térjünk vissza most az eredeti, k-adik elemet kiválasztó feladathoz. Lássuk először a furfangos válaszolónak a – bármely k-adik elemet kiválasztó algoritmusra működő – legrosszabb esetet konstruáló stratégiáját. Rögzít egy $x \in X$ értéket, mely ebben a legrosszabb esetben a sorozat k-adik eleme lesz. Ezt az értéket majdan a kérdező által utoljára kérdezett elemnek adja. Egyébként az s sorozatot nem rögzíti előre, csak a kérdező kérdéseinek függvényében. A konzisztencia biztosítására készít magának egy táblázatot, melyben az adott indexű sorozatelem státuszát és pillanatnyi értékét tárolja.

A státuszok:

- N: nem volt még rá vonatkozó kérdés, ilyenkor értéke még nem rögzített.
- L: volt már rá kérdés és értéke x-nél nagyobb értékre már rögzítve van.
- S: volt már rá kérdés és értéke x-nél kisebb értékre már rögzítve van.
- U: az utoljára kérdezett elem, értéke x

A furfangos válaszoló nem lényegi kérdésekre kényszerítő stratégiája – a később említendő megszorításokkal – a következő:

1. N, N típusú kérdés esetében az elsőt x-nél kisebbre, a másodikat x-nél nagyobbra rögzíti a megfelelő bejegyzéssel együtt, és megadja a rögzítésnek megfelelő választ.
2. L, N ,illetve S,N típusú kérdésnél az eddig nem kérdezett elemet a másik oldalra rögzíti, kiadva az ennek megfelelő választ.
3. S, L típusú kérdésnél a már rögzített értékeknek megfelelő választ ad.
4. Az utoljára rögzített elem értékét x-re, bejegyzését U-ra állítja.

A megszorítás az, hogy nem keletkezhet $(k-1)$-nél több S bejegyzésű és $(n-k)$-nál több L bejegyzésű elem. Kicsit másképp fogalmazva, ha az egyik irány telítődött, akkor a bejegyzést és az értékét a másik oldalra kell állítani; az utolsót pedig U-ra.

Minden olyan hasonlítás, melyben van N bejegyzésű elem és még egyik irány sem telített, nem lényegi lesz. Ezek száma a kérdező stratégiájától függő. Ha nem szerencsésen kérdez, akkor akár $(n-2)$ is lehet az ilyen kérdések száma. (Az első kérdéstől eltekintve, mely mindkét irányt telíti, a többi kérdésnél 1-el telítődik valamelyik oldal). Ha viszont a kérdező ügyes és N, N bejegyzésű párokat kérdez, akkor minden kérdés telíti mindkét oldalt, ezért a kikényszerített nem lényegi kérdések száma optimális esetben $\min(k-1, n-k)$ lesz.

Azt kaptuk tehát, hogy legrosszabb esetben a nem lényegi kérdések száma még az optimális stratégiánál is $\min(k-1, n-k)$. Ezzel a bizonyítást befejeztük.

Maximum, illetve minimum esetében ($k = n$, illetve $k = 1$) a minimum egyik tagja 0, ezért visszakapjuk ez előző tétel állítását. A medián esetében, vagyis ha $k = \lceil n/2 \rceil$, pedig

$\min(k-1, n-k) = \min(\lceil n/2 \rceil - 1, \lfloor n/2 \rfloor$, ami könnyen látható módon $\lfloor (n-1)/2 \rfloor$ -vel egyenlő.

Összeadva ezt $(n-1)$-gyel, az alsó korlátra $\lfloor 3\,(n-1)/2 \rfloor$ adódik. Például $n = 5$ esetében ez az alsó korlát $\lfloor 3 * 4/2 \rfloor = 6$.

Foglalkozzunk először a maximum kiválasztással (a minimum kiválasztása ennek analogonja)!

9.2 Maximum kiválasztás

Formálisan is felírjuk a maximum-kiválasztás feladatát, majd megadjuk a jól ismert és sokszor használt megoldó algoritmust.

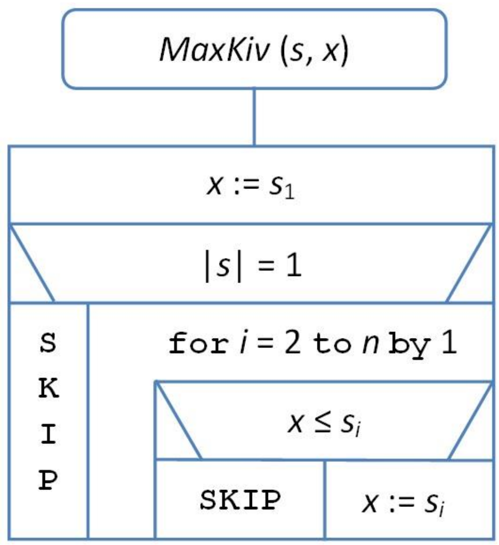
Feladat: Valamely $s \in X^*$ sorozat maximumának megkeresése.

Deklaráció: $s, s' \in X^*, x \in X$

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

Utófeltétel: $s = s' \wedge x$ az s maximális eleme.

Egy megoldó algoritmus – legyen a neve *MaxKiv* – a következő:



9.2. ábra. Maximum kiválasztás algoritmusa

Elemzés. A ciklus $|s| - 1$ -szer fut le, minden iterációs lépés egy összehasonlítást tartalmaz.

$$Ö_{MaxKiv}(s) = |s| - 1,$$

$$MÖ_{MaxKiv}(n) = n - 1.$$

Ez a maximum-kiválasztó eljárás optimális, hiszen a maximum-tétel miatt ennél jobb nem készíthető.

9.3 A *k*-adik elem kiválasztása

Természetes gondolat, hogy a k-adik elemet annak definícióját követve úgy keressük meg, hogy s-et valamilyen rendezési módszerrel rendezzük, majd a rendezett sorozat k indexű tagját adjuk eredményül. Mivel egy n elemű sorozat rendezésénél az összehasonlítások száma legalább $n \log n$ nagyságrendű, ez a megoldás távol van a kiválasztási tételben megadott lineáris alsó korláttól. Ennek magyarázata kézenfekvő. Ahhoz, hogy egy x elemről detektáljuk, hogy az s sorozat k-adik eleme, valójában elegendő volna találni a sorozatban $(k-1)$ nála kisebb-egyenlő és $|s| - k$ nála a nagyobb-egyenlő elemet. A teljes rendezés felesleges többletmunkával jár, hiszen nem csak a kisebb-egyenlő és nagyobb-egyenlő elemeket adja meg, hanem azokat még – a feladat szempontjából feleslegesen – rendezi is. A következő algoritmus ezen a meggondoláson alapul.

Feladat: Valamely $s \in X^*$ sorozat k-adik elemének megkeresése.

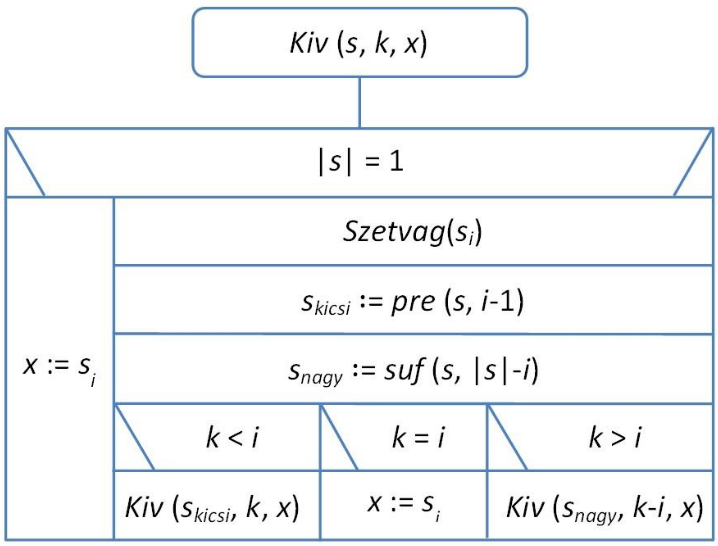
Deklaráció: $s, s' \in X^*,\ x \in X,\ i, j \in Z$

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

Utófeltétel: $s = Perm(s')$ és x az $s'k$ –adik eleme

A k-adik elemet kiválasztó $Kiv\,(s, k, x)$ algoritmus a $Szetvag\,(s, i)$ eljárással s-et úgy rendezi át, hogy a sorozat eredetileg legutolsó elemét a helyére viszi olyan módon, hogy tőle balra csak nálánál kisebb-egyenlő elemek, míg tőle jobbra nagyobb elemek kerülnek. Ha ez az elem a i –edik helyre került, akkor $k = i$ esetén készen vagyunk, $x = s_i$ a keresett elem. Ha $k < i$, akkor az első felének, s_{kicsi}-nek a k –adik elemét kell tovább keresni, egyébként pedig a második felében, s_{nagy}-ban a $(k - i)$ –ediket kell meghatározni, ugyanezzel a módszerrel.

Ennek megfelelően algoritmusunk rekurzív lesz. A $Kiv(s, k, x)$ eljárás programja a következő:



9.3. ábra. A $Kiv(s, k, x)$ algoritmusa

Elemzés. Most is az összehasonlítások számát számoljuk. A $Szetvag\,(s, i)$ eljárás nyilván $(n - 1)$ összehasonlítással működik. A Kiv a legrosszabb esete az, ha a szétvágás mindig 1-gyel csökkenti azt a méretet, melyben a tovább kell keresnünk. Ilyenkor az összehasonlítások száma az első $(n - 1)$ egész szám összege.

$$MÖ_{Kiv}(n, k) = n(n - 1)/2$$

A legjobb eset az, amikor a sorozat utolsó eleme pontosan a k –adik $(k = i)$, hiszen ekkor azonnal leállhatunk:

$$mÖ_{Kiv}(n, k) = n - 1$$

Mivel a legrosszabb és legrosszabb eset összehasonlításainak a száma nagyságrendben különböző, ezért érdekes megvizsgálni, hogy melyik az, amelyik inkább dominál. Az egyszerűség kedvéért tegyük fel, hogy s –et az $1,\ 2, \ldots, n$ permutációi alkotják, egyforma $1/n!$ valószínűséggel. Arra vagyunk kíváncsiak, hogy mit mondhatunk az $EÖ_{Kiv}(n, k)$ várható értékről.

Tétel. Tetszőleges $k \in [1 \dots n]$ esetén a $Kiv(s,k,x)$ eljárás végrehajtott összehasonlításainak várható számára $EÖ_{Kiv}(n,k) \leq 4n$.

Bizonyítás. (n-re vonatkozó indukcióval) Kis n-ekre (n =0, 1) az állítás nyilvánvalóan igaz. Igazoljuk az állítást $n \geq 2$-re, feltételezve, hogy kisebb elemszámra az állítás már igaz. Vezessük be a következő függvényeket:

$f(n,k) = \; EÖ_{Kiv}(n,k)$

$f_j(n,k)$ legyen az összehasonlítások várható értéke olyan feltétel mellett, hogy az utolsó helyen j volt. Ennek a feltételnek a valószínűsége $1/n$.

Amennyiben $j = k$, akkor $f_j(n,k) = n - 1$.

Ha $j \neq k$, akkor a j-nél kisebb, illetve j -nél nagyobb elemek egymás közötti sorrendje is egyforma valószínűségű. Ezért $k < j$ esetén az $(n-1)$-hez hozzáadódik még $f(j-1,k)$, míg $k > j$ esetén $f(n-j,k-j)$.

Összefoglalva az alábbi összefüggést kapjuk a feltételes várható értékekre:

$$f_j(n,k) = \begin{cases} n-1, & \text{ha } j = k \\ (n-1) + f(j-1,k), & \text{ha } k < j \\ (n-1) + f(n-j,k-j), & \text{ha } k > j \end{cases}$$

A teljes várható érték tétele alapján:

$$f(n,k) = \frac{1}{n}\sum_{j=1}^{n} f_j(n,k).$$

Beírva ide $f_j(n,k)$ előbbi alakját, az esetszétválasztás és átrendezés után a következőt kapjuk:

$$f(n,k) = (n-1) + \frac{1}{n}\sum_{j=k+1}^{n} f(j-1,k) + \frac{1}{n}\sum_{j=1}^{k-1} f(n-j,k-j)$$

Mind $j-1$, mind $k-j$ kisebb n –nél, ezért ide beírhatjuk az indukciós feltétel szerinti becsléseket:

$$f(n,k) \leq (n-1) + \frac{1}{n}\sum_{j=k+1}^{n} 4(j-1) + \frac{1}{n}\sum_{j=1}^{k-1} 4(n-j) =$$

$$= (n-1) + \frac{1}{n}\sum_{j=k}^{n-1} 4j + \frac{1}{n}\sum_{j=1}^{k-1} 4(n-j) = (n-1) + \frac{4}{n}\sum_{j=1}^{n-1} j - \frac{4}{n}\sum_{j=1}^{k-1} j + \frac{4}{n}\sum_{j=1}^{k-1} (n-j)$$

$$=$$

$$= (n-1) + \frac{4}{n} * \frac{n(n-1)}{2} + \frac{4}{n}\sum_{j=1}^{k-1}(n-2j) =$$

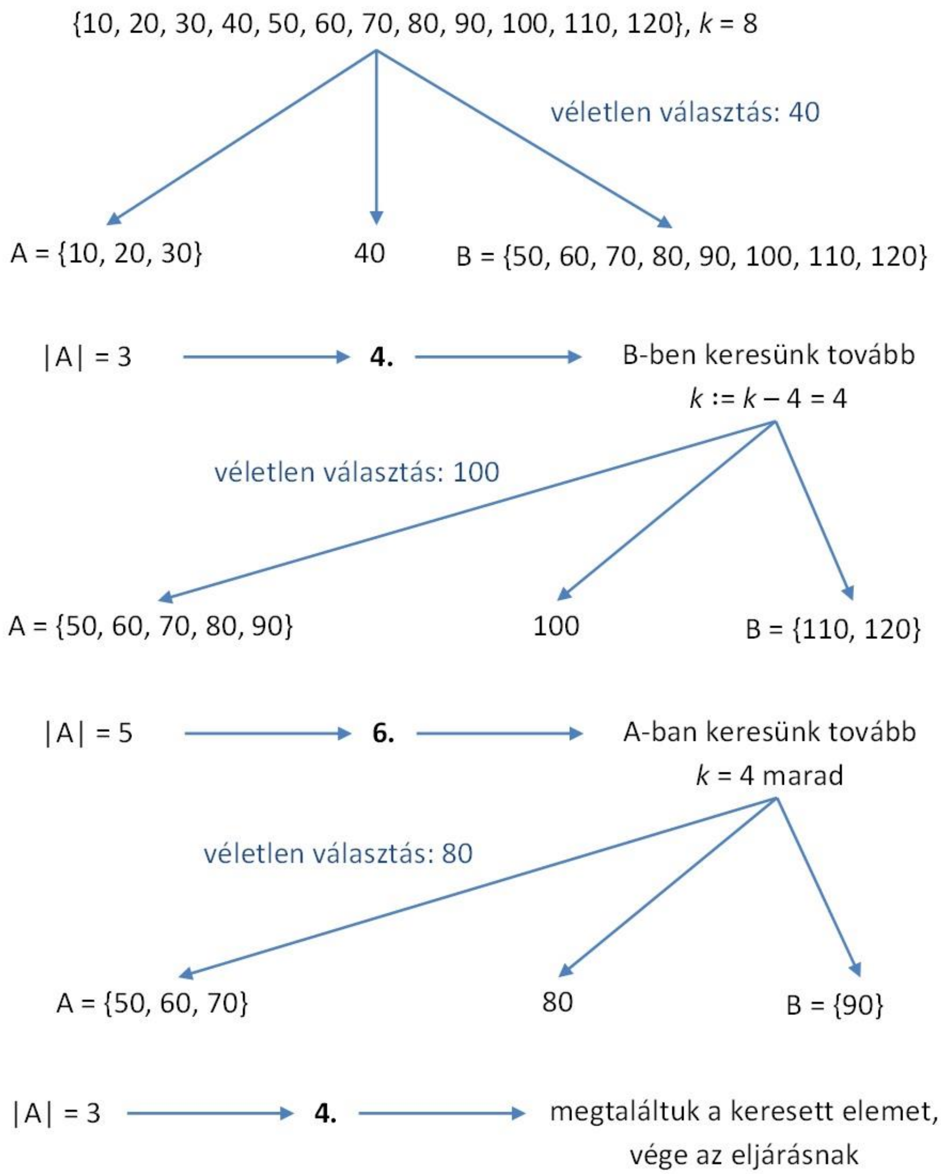
$$= 3(n-1) + \frac{4}{n}(k-1)\frac{(n-2) + \big(n-2(k-1)\big)}{2} =$$

$$= 3(n-1) + \frac{4}{n}(k-1)(n-k) \leq 3(n-1) + \frac{4}{n}\left(\frac{n-1}{2}\right)^2 \leq 3n + \frac{4}{n}\frac{n^2}{4} = 4n.$$

Közben alkalmaztuk a számtani sorozatok összegképletét, illetve a számtani és mértani közép közötti egyenlőtlenséget. ■

9.4 A k-adik elem kiválasztás véletlenített algoritmusa

A $Kiv(s, k, x)$ algoritmus összehasonlításainak várható számánál feltételeztük, hogy s-ben minden permutáció egyformán valószínű. A gyakorlatban ez sokszor nincs így, például a bemenő sorozat már közel rendezett is lehet. Ha ilyenkor valamilyen kis értékre keressük a k-adik elemet, akkor a méret csak lassan csökken. Ennek kivédésére véletlenített algoritmust alkalmazunk, nem az utolsó elemet, hanem a sorozat véletlenül választott tagját használja a szétvágás során.



9.4. ábra. A véletlenített algoritmus működése

Példa: Tekintsük a $10,\ 20, \dots,\ 120$ sorozatot, és keressük a $k = 8$-adik elemet!

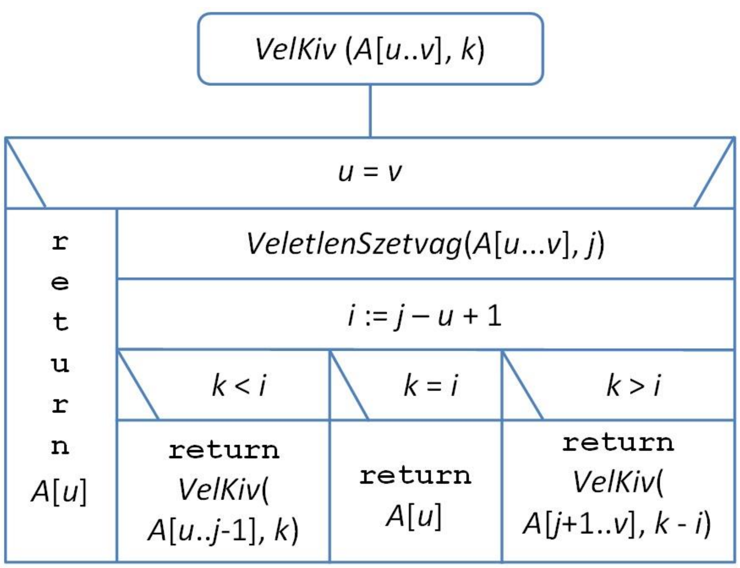
1. A véletlen választás eredménye legyen 40, így ennek megfelelően bontsuk szét a sorozatot: $A =\ 10,\ 20,\ 30$ *és* $B = 50,\ 60, \dots,\ 120$. Mivel $|A| = 3$, tudjuk, hogy a 40 a 4. elem a számsorozatban, ezért a 8-adik elemet az A sorozatban kell keresnünk. Mivel a számsorozat első 4 elemét leválasztottuk, ezért a B sorozatban már nem a 8-adik, hanem a 4. elemet keressük.
2. A bemenő sorozatunk $50,\ 60, \dots,\ 120$, a véletlen választás eredménye legyen 100, így $A = 50, \dots,\ 90$ *és* $B = 110,\ 120$ sorozatok keletkeznek, így a 100 a 6. legkisebb elem, tovább kell tehát keresnünk a A halmazban, továbbra is a 4. elemet.
3. A bemenő sorozatunk $50, \dots,\ 90$, a véletlen választás eredménye legyen 80, így $A = 50, \dots, 90$ és $B = 90$, tehát a 80 itt a 4. elem, vagyis megtaláltuk a $k = 8$ -adik elemet.

Az algoritmus működését a 9.4. ábrával illusztráltuk.

Foglalkozzunk most azzal az esettel, amikor a sorozat típust az $A[1..n]$ tömbben reprezentáljuk. Most az s_{kicsi} és s_{nagy} sorozatok a tömb darabjai, ezért eljárásunk további lépéseiben már nem az egész tömbön működik, hanem általában egy u és v közötti darabján ($1 \leq u \leq v \leq n$). Emiatt az algoritmust kicsit általánosabban írjuk meg, a tömb $A[u..v]$ szeletén. Olyan függvényeljárást készítünk, mely a k-adik elem értékét adja vissza.

Legyen $VeletlenSzetvag$ olyan eljárás, mely a $Szetvág$ eljárást az $A[u..v]$ tömbdarabon egy onnan véletlenül választott elemmel végzi el úgy, hogy j-ben adja vissza a szétválasztó elem helyét. Világos, hogy ez az $A[u..v]$-nek az $i = (j - u + 1)$-edik eleme lesz. Ekkor i-t kell összehasonlítanunk k-val. Ha egyenlők, akkor készen vagyunk és ezt az értéket adjuk vissza. Ha $i > k$, akkor $A[u..j - 1]$-ben keressük tovább a k –adik elemet, míg $i < k$ esetén $A[j + 1..v]$-ben keressük a $(k - i)$-edik elemet.

A $Kiv(A[u..v], k)$ rekurzív algoritmus külső hívása a teljes $A[1..n]$ tömbbel és az eredetileg megadott k értékkel történik. Később az eljárás önmagát egy résztömbre hívja meg, esetleg módosított k értékkel. Az eljárás működésének leírását a 9.5. ábra tartalmazza.



9.5. ábra. A *VelKiv* algoritmus

Elemzés. A véletlenített kiválasztás elemzése formálisan ugyanúgy történhet, mint a determinisztikus változaté. Az elvi eltérés abban van, hogy míg a Kiv eljárás esetében az input sorozat a véletlenszerű, addig a véletlenített változatban a véletlen magában az algoritmusban, nevezetesen a $VeletlenSzetvag(A[u..v], j)$ eljárás működésében jelenik meg.