Adatstruktúrák és Algoritmusok

8. gyakorlat

Rendezések I

Rendezés

Definíció. *Legyen X egy halmaz és $\leqslant$ egy reláció X-en. Az $\leqslant$ relációt **rendezés**nek nevezzük, ha rendelkezik a következő tulajdonságokkal:*

1. ***reflexív**, azaz $x \leqslant x$ minden $x \in X$ esetén.*
2. ***antiszimmetrikus**, azaz ha $x \leqslant y$ és $y \leqslant x$ akkor $x = y$ minden $x, y \in X$ esetén.*
3. ***tranzitív**, azaz $x \leqslant y$ és $y \leqslant z$, akkor $x \leqslant z$ minden $x, y, z \in X$ esetén.*
4. ***dichotom**, azaz $x \leqslant y$ vagy $y \leqslant x$ minden $x, y \in X$ esetén.*

Azt mondjuk, hogy az $(X, \leqslant)$ egy rendezett halmaz, ha $\leqslant$ egy rendezés az X-en.

A Rendező algoritmusok

1. Minimum kiválasztásos rendezés: $T(n) = \Theta(n^2)$.
2. Gyorsrendezés: $T(n) = \Theta(n^2)$, de átlagosan $T(n) = \Theta(n \log(n))$.
3. Leszámláló rendezés: $T(n) = \Theta(n)$.
4. Buborék rendezés: $T(n) = \Theta(n^2)$.
5. Beszúró rendezés: $T(n) = \Theta(n^2)$.
6. Shell rendezés: $T(n) = \Theta(n^{1,5})$.
7. Összefésülő rendezés: $T(n) = \Theta(n \log(n))$.
8. Négyzetes rendezés: $T(n) = \Theta(n^{1,5})$.
9. Számjegyes (radix) rendezés: $T(n) = \Theta(n)$.
10. Edény rendzés: átlagosan $T(n) = \Theta(n)$.
11. Kupac rendezés: átlagosan $T(n) = \Theta(n \log(n))$.

Először átnézzük a **minimumkiválasztásos rendezés**t, majd a **kiválasztás lineáris időben** algoritmust, ezt követően a **gyorsrendezés**t, mivel ez utóbbi kettő mindkettő a feloszt algoritmust használja.
Nyilván van többféle logikailag indokolt sorrend, amire a rendezések felfűzhetőek, az általunk választott nem feltétlenül a legjobb.
Ha csak egy rendező algoritmust kellene bemutatni, akkor az a buborék rendezés lenne (ami négyzetes futási idejű), talán ezt követné a beszúrásos, aztán a shell. (Persze mindegyik gyönyörű.)

Minimumkeresés tömbben

Egy A tömb $A[p \ldots r]$ résztömbében keressük a minimumot. Az algoritmus a minimum indexével tér vissza.

MINKER($A, p, r,$ minindex)

INPUT: A, p, r.

OUTPUT: minindex.

	MINKER($A, p, r,$ minindex)
1.	minindex $\longleftarrow p$
2.	FOR $j \longleftarrow p+1$ TO r DO
3.	IF $A[j] < A[$minindex$]$ THEN minindex $\longleftarrow j$
4.	RETURN(minindex)

1. Minimumkiválasztásos rendezés

Minimumkiválasztásos rendezés

MINREND(A)
INPUT: $A[1, \ldots, n]$ a rendezendő tömb.
OUTPUT: $A[1, \ldots, n]$ a rendezett tömb.

	MINREND(A)
1.	FOR $i \longleftarrow 1$ TO $n-1$ DO
2.	MINKER($A, i, n,$ minindex)
3.	IF $i <$ minindex THEN CSERE ($A[i], A[$minindex$]$)
4.	RETURN(A)

Feladat

Minimumkiválasztásos rendezés segítségével rendezzük az

$$A = [13, 22, 8, 315, 3]$$

tömböt.

$i = 1$ MINKER$(A, 1, 5)$ minindex $= 4$ CSERE $(A[1], A[4])$

$$A = [3||22, 8, 315, 13]$$

$i = 2$ MINKER$(A, 2, 5)$ minindex $= 3$ CSERE $(A[2], A[3])$

$$A = [3, 8||22, 315, 13]$$

$i = 3$ MINKER$(A, 3, 5)$ minindex $= 5$ CSERE $(A[3], A[5])$

$$A = [3, 8, 13||315, 22]$$

$i = 4$ MINKER$(A, 4, 5)$ minindex $= 5$ CSERE $(A[4], A[5])$

$$A = [3, 8, 13, 22, 315]$$

Kiválasztás lineáris időben algoritmus

A kiválasztási probléma

Adott egy A halmaz, amelynek n darab páronként különböző eleme van, továbbá adott egy $i \in \mathbb{Z}_+$ szám (nem index, hanem darabszám), amelyre $1 \leqslant i \leqslant n$. Keressük az A halmaznak azt az x elemét, amelyre teljesül, hogy pontosan $(i-1)$ darab nála kisebb kisebb eleme van a halmaznak.

Kiválasztás lineáris időben

KIVÁLASZT(A, p, r, i, x)

INPUT: A a tömb, amelynek az $A[p \dots r]$ résztömbjének keressük az i-edik legkisebb elemét. $i \in \mathbb{Z}_+$, $1 \leqslant i \leqslant r - p + 1$..

OUTPUT: x a keresett elem.

	KIVÁLASZT(A, p, r, i, x)
1.	IF $p = r$ THEN $x \longleftarrow A[p]$
2.	ELSE Az $A[p \dots r]$ tömb elemeit 5 elemből álló csoportokba soroljuk. A kapott csoportok mediánjait elhelyezzük egy M tömb $M[1 \dots k]$ résztömbjébe, ahol $k = \left\lceil \frac{r-p+1}{5} \right\rceil$
3.	$\mathrm{medmed} \longleftarrow \mathrm{med}\,(M[1], \dots, M[k])$. Ennek a mediánnak a számára $k \leqslant 5$ esetén közvetlenül történik, ellenkező esetben a Kiválasztás $\left(M, 1, k, \lceil \frac{k}{2} \rceil, \mathrm{medmed}\right)$ módon, tehát rekurzívan.
4.	FELOSZT$(A, p, r, \mathrm{medmed}, q)$. IF $i \leqslant q - p + 1$ THEN Kiválasztás(A, p, q, i, x) ELSE Kiválasztás$(A, p + 1, r, i - (q - p + 1), x)$
5.	RETURN(x)

Magyarázat

Az $A[p..r]$ tömb i-edik eleme a $p + i - 1$ elem. Azt kell megvizsgálnunk, hogy ez az elem a q indexű elemhez képest hol van a tömbben. Nézzük meg a 4. pontot.
4. IF $p + i - 1 \leqslant q$ THEN KIVÁLASZT(A, p, q, i, x)
ELSE KIVÁLASZT$(A, q + 1, r, i - (q - p + 1), x)$.
Azonban a $p + i - 1 \leqslant q$ feltétel ekvivalens az $i \leqslant q - p + 1$ feltétellel.

Feladat

Válasszuk ki az

$$A = [17, 127, 20, 45, 81, 3, 10, 49, 50, 28, 12, 9]$$

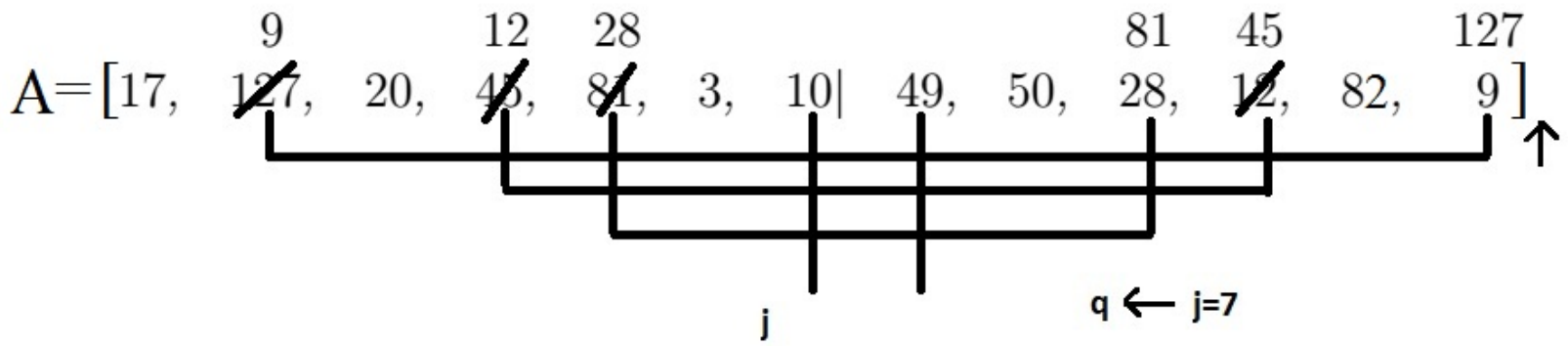
tömb 9. elemét.

1. KIVÁLASZT$(A, p = 1, r = 13, i = 9, x)$

Felosztjuk a tömböket 5-ös csoportokra, med, medmed.

	$17, 127, 20, 45, 81\|$	$3, 10, 49, 50, 28\|$	$12, 82, 9$
med	45	28	12
medmed		28	

FELOSZT$(A, 1, 13, 28, q)$ $x = 28$



q=7

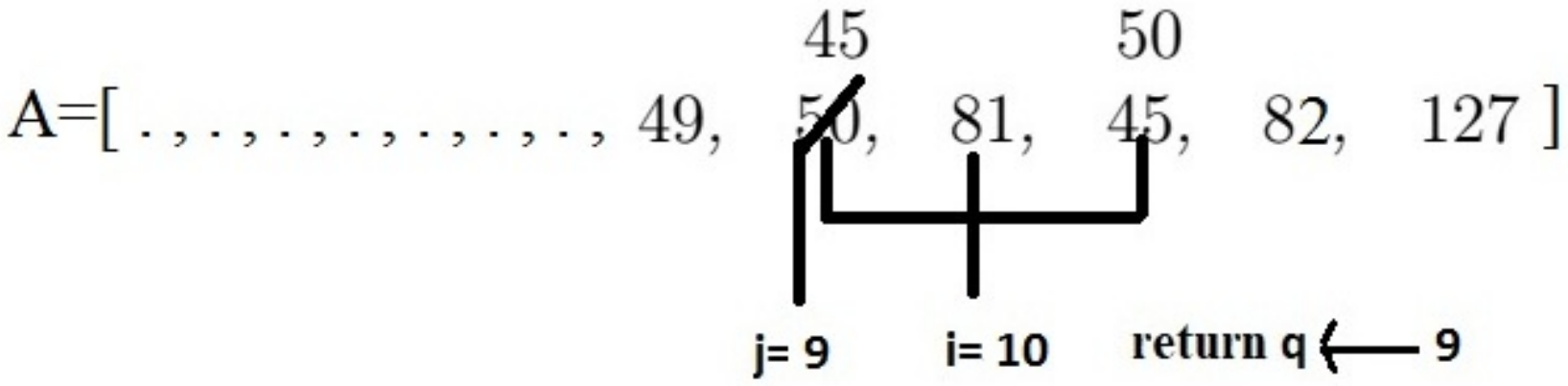
$i = 9 \overset{?}{\leqslant} q - p + 1 = 7 - 1 + 1 = 7$ (nem) $\Longrightarrow$

2.KIVÁLASZT(A, 8, 13, 9-(7-1+1)=2, x)
p r i

A med és a medmed kiszámítása

	49,	50,	81,	45,	82	127
med			50			127
medmed				50		

FELOSZT($A, p = 8, r = 13, x = 50, q$)



q=9

$i = 2 \overset{?}{\leqslant} q - p + 1 = 9 - 8 + 1 = 2$ (igaz), $\Longrightarrow$

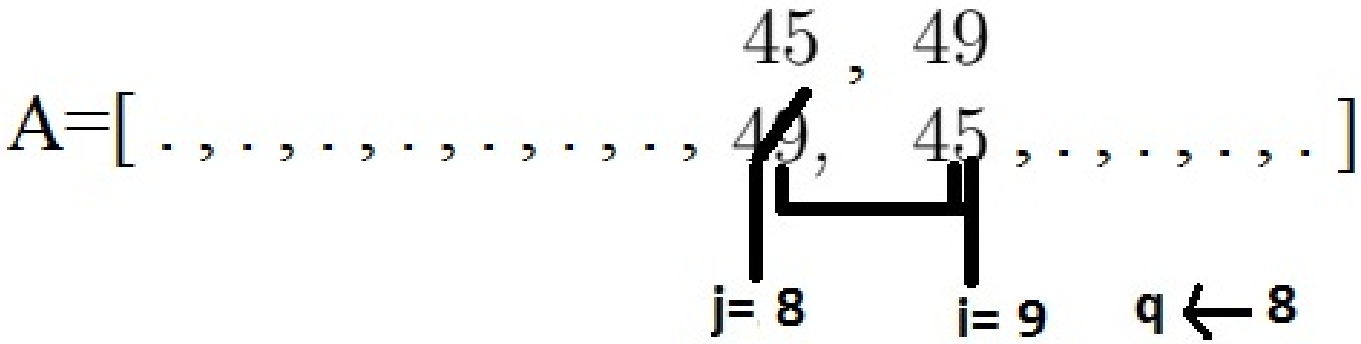
3.KIVÁLASZT(A, 8, 9, 9-8+1=2, x)
p r i

49, 45

med 45

medmed 45

FELOSZT($A, p = 8, r = 9, x = 45, q$)



q=8

IF $i = 2 \overset{?}{\leqslant} p - q + 1 = 8 - 8 + 1 = 1$ (nem) $\Longrightarrow$

4.KIVÁLASZT(A, $q+1 = 8+1 = 9$, $r = 9$, 2-(8-8+1)=1, x)
p r i

IF $p = r$ THEN $x \longleftarrow A[p] = 49$ RETURN($q = 49$).

Tehát az A tömb 9. eleme a 49.

Ellenőrzés minimumkiválasztásos rendezés segítségével

$$A = \left[\boxed{17}, 127, 20, 45, 81, \boxed{3}, 10, 49, 50, 28, 12, 82, 9\right]$$

$$A = \left[3 || \boxed{127}, 20, 45, 81, 17, 10, 49, 50, 28, 12, 82, \boxed{9}\right]$$

$$A = \left[3, 9 || \boxed{20}, 45, 81, 17, \boxed{10}, 49, 50, 28, 12, 82, 127\right]$$

$$A = \left[3, 9, 10 || \boxed{45}, 81, 17, 20, 49, 50, 28, \boxed{12}, 82, 127\right]$$

$$A = \left[3, 9, 10, 12 || \boxed{81}, \boxed{17}, 20, 49, 50, 28, 45, 82, 127\right]$$

$$A = \left[3, 9, 10, 12, 17 || \boxed{81}, \boxed{20}, 49, 50, 28, 45, 82, 127\right]$$

Ellenőrzés minimumkiválasztásos rendezés segítségével

$$A = \left[3, 9, 10, 12, 17, 20 || \boxed{81}, 49, 50, \boxed{28}, 45, 82, 127\right]$$

$$A = \left[3, 9, 10, 12, 17, 20, 28 || \boxed{49}, 50, 81, \boxed{45}, 82, 127\right]$$

$$A = \left[3, 9, 10, 12, 17, 20, 28, 45 || \boxed{50}, 81, \boxed{49}, 82, 127\right]$$

$$A = \left[3, 9, 10, 12, 17, 20, 28, 45, \boxed{49}, 81, 50, 82, 127\right]$$

Látható, hogy $A[9] = 49$, azonban ehhez az eredményhez majdnem rendezni kellett az egész tömböt.

Gyorsrendezés

A gyorsrendezés valóban gyors, bár a futási ideje négyzetes, de az átlagos futási ideje $n \log(n)$, ami egy elméleti alsó korlát az összehasonlításon alapuló rendezésekre. Az algoritmus rekurzív. A rendezendő tömböt az első eleme körül felosztjuk, majd két résztömbre bontjuk a felosztás határa szerint. Ezt követően a résztömbökön futtatjuk az algoritmust mindaddig, amíg egy elemű résztömböket nem kapunk. Ezáltal a tömb rendezetté válik.

Gyorsrendezés:

Talán a legtrükkösebb rendezés a gyorsrendezés, ami a nevéhez híven valóban gyors a maga $n\log(n)$-es átlagos futási idejével. Ez a rendezés is használja a feloszt algoritmust.
$T(n) = \Theta(n^2)$, átlagosan $T(n) = \Theta(n\log(n))$.
GY(A, p, r)
INPUT: Az A tömb $A[p, \ldots, r]$ résztömbjét rendezzük.
OUTPUT: Az A tömb a rendezett $A[p, \ldots, r]$ résztömbökkel.

	GY(A, p, r)
1.	IF $p < r$
2.	THEN FELOSZT$(A, p, r, A[p], q)$
3.	GY(A, p, q)
4.	GY$(A, q+1, r)$
5.	RETURN(A)

Feladat.

A gyorsrendezés segítségével rendezzük az alábbi A tömböt.

$A = [91, 117, 52, 11, 63, 49]$

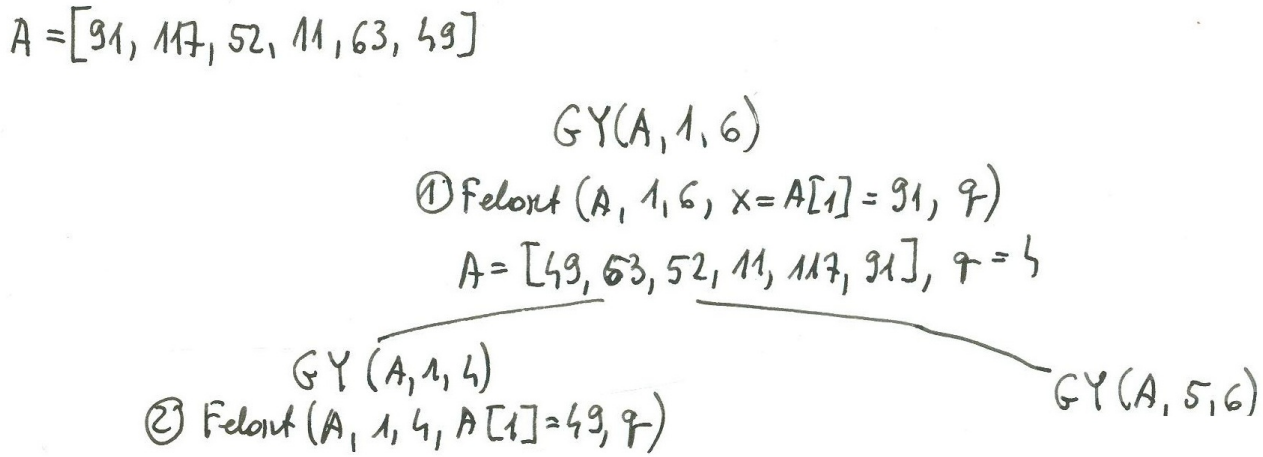
$GY(A, 1, 6)$

① $Felosz(A, 1, 6, x = A[1] = 91, q)$

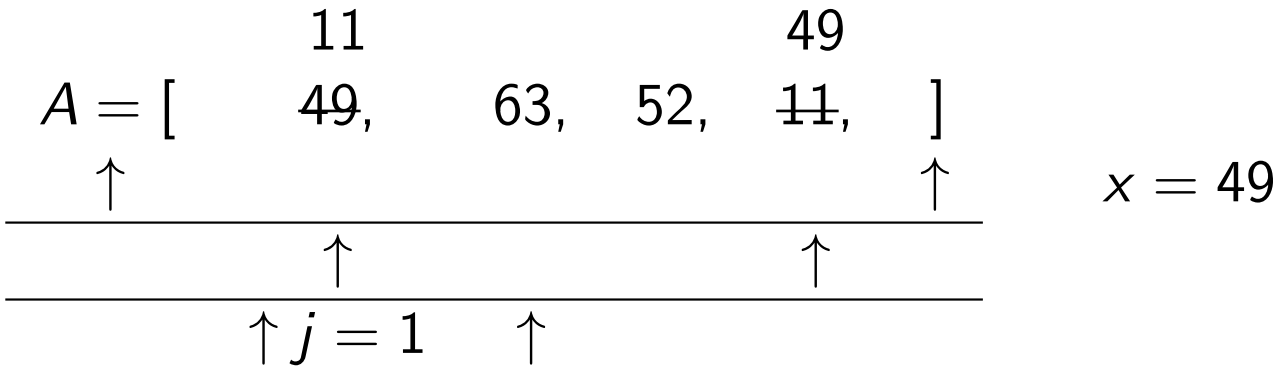
1.) $\text{FELOSZT}(A, 1, 6, x = A[1] = 91, q)$

	49				117	91		
$A = [$	~~91~~,	~~117~~,	52,	11,	~~63~~,	49	]	
$i \uparrow$	$\uparrow$					$\uparrow$	$\uparrow j$	$x = 91$
		$\uparrow$			$\uparrow$			
				$\uparrow j$	$\uparrow i$			

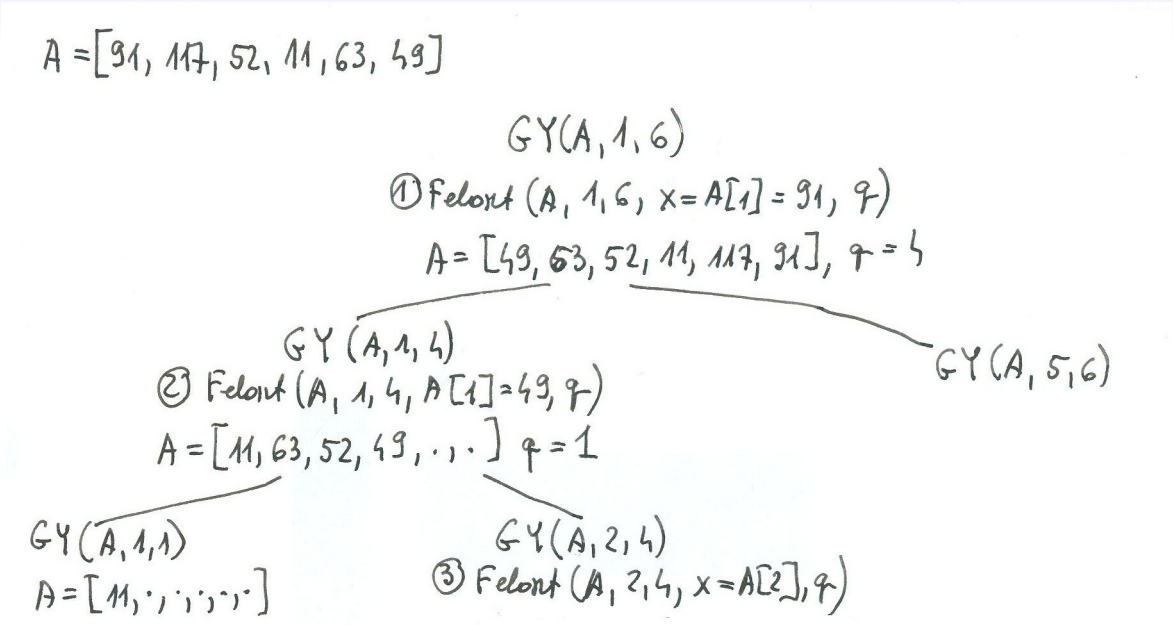
$q \longleftarrow j = 4$



2.) FELOSZT$(A, 1, 4, x = A[1] = 49, q)$



RETURN $(q \longleftarrow j = 1)$,



3.) FELOSZT$(A, 2, 4, x = A[2] = 63, q)$

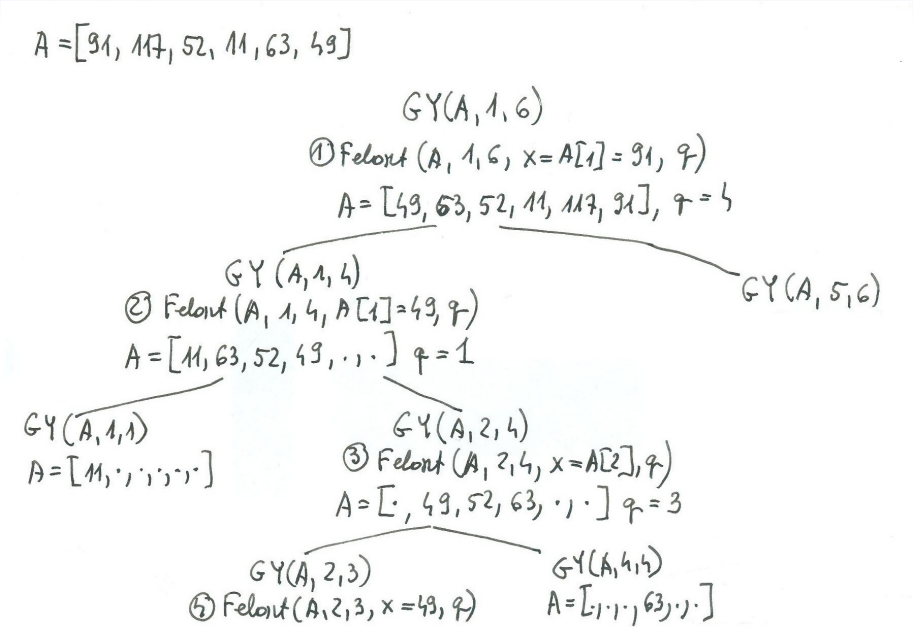
$$A = [\ \cdot,\ \overset{49}{\cancel{63}},\ 52,\ \overset{63}{\cancel{49}},\ \cdot,\ \cdot\ ] \qquad x = 63$$

$$\uparrow \qquad\qquad\qquad \uparrow$$

$$\uparrow \qquad\qquad \uparrow$$

$$\uparrow j \quad \uparrow i$$

RETURN $(q \longleftarrow j = 3)$,



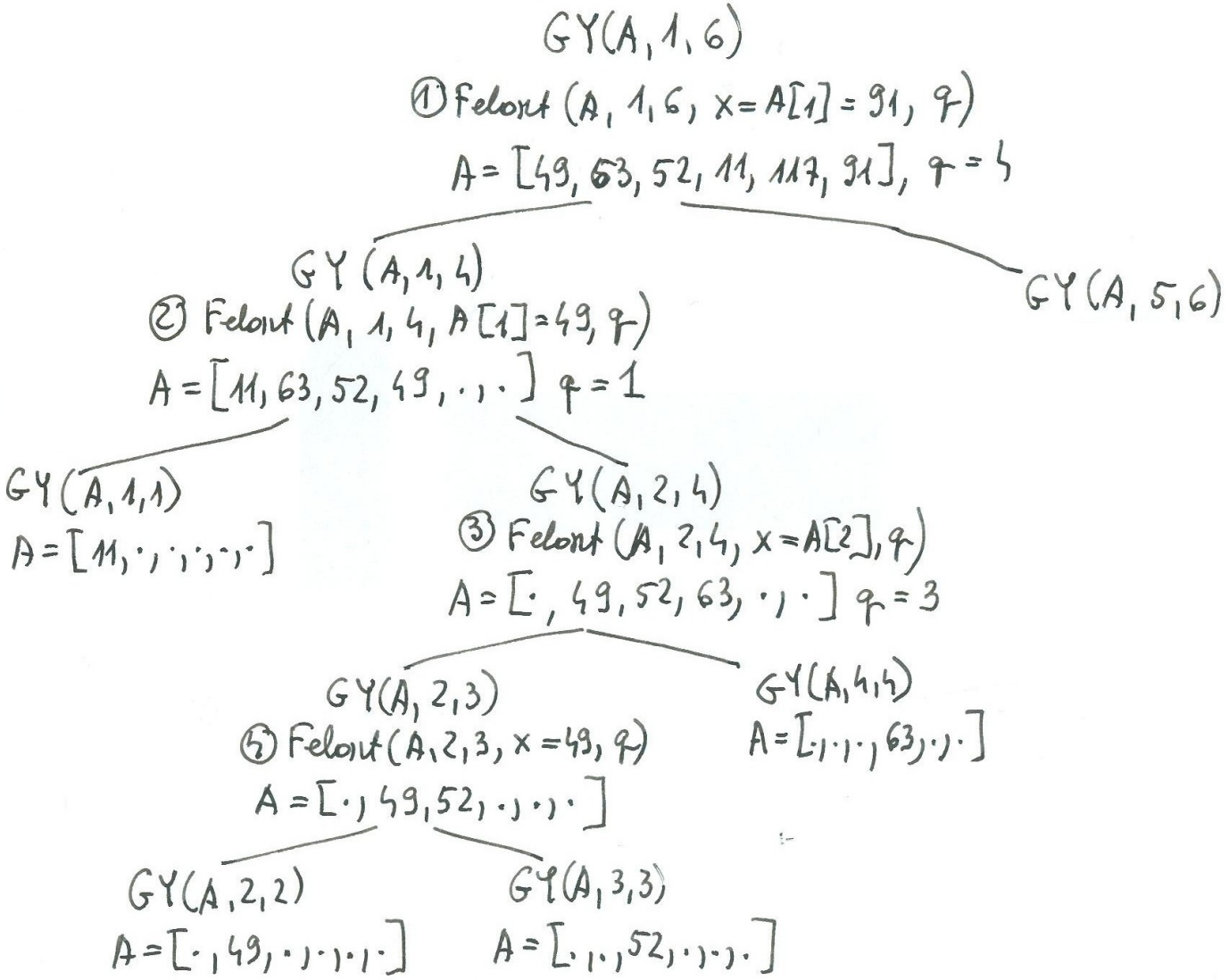
4.) FELOSZT$(A, 2, 3, x = 49, q)$

$$A = [\quad \cdot, \quad 49, \quad 52, \quad \cdot, \quad \cdot, \quad \cdot \quad]$$

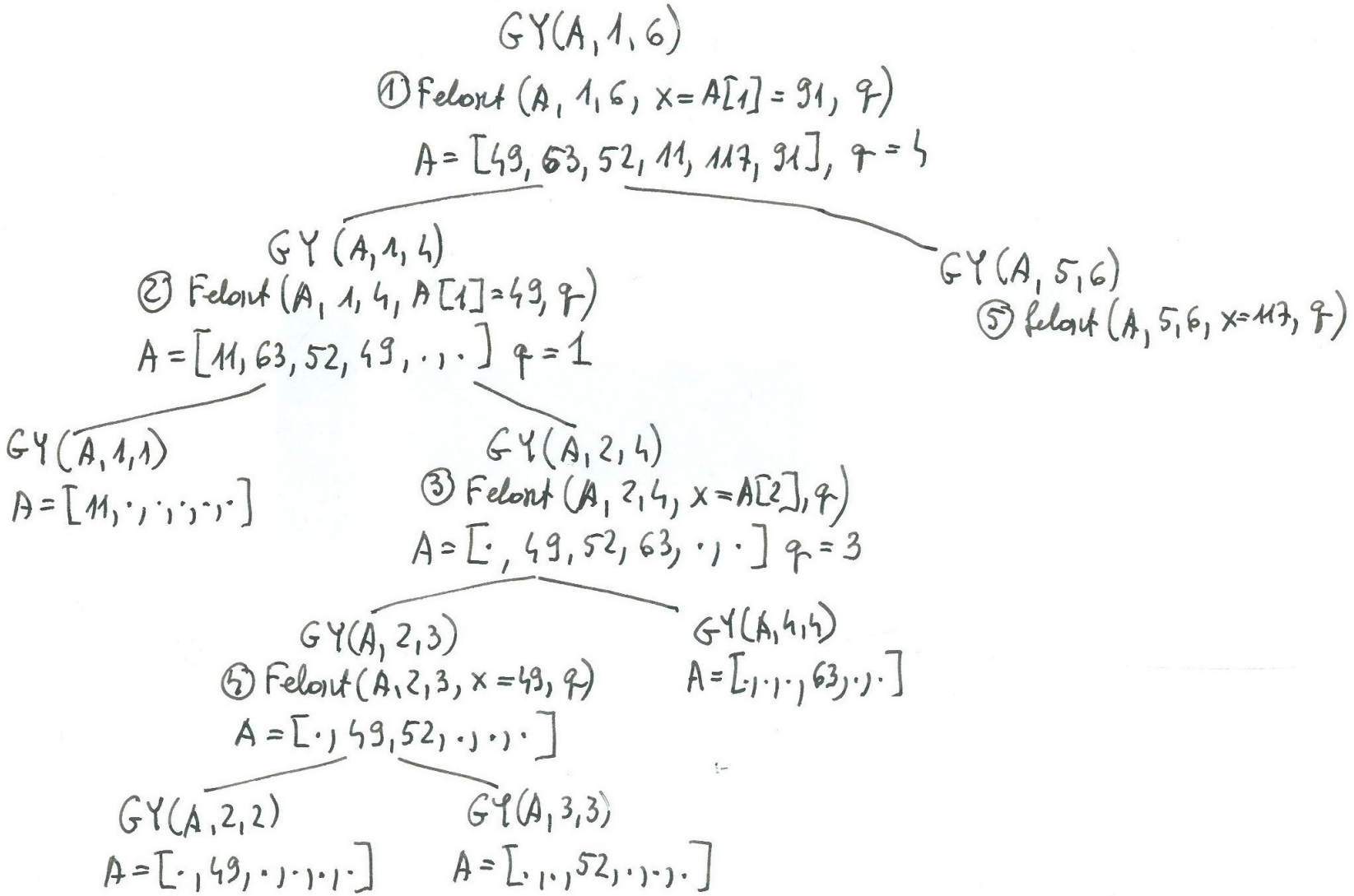
$$\uparrow j \quad \uparrow i \qquad x = 49$$

RETURN $(q \longleftarrow j = 2)$,

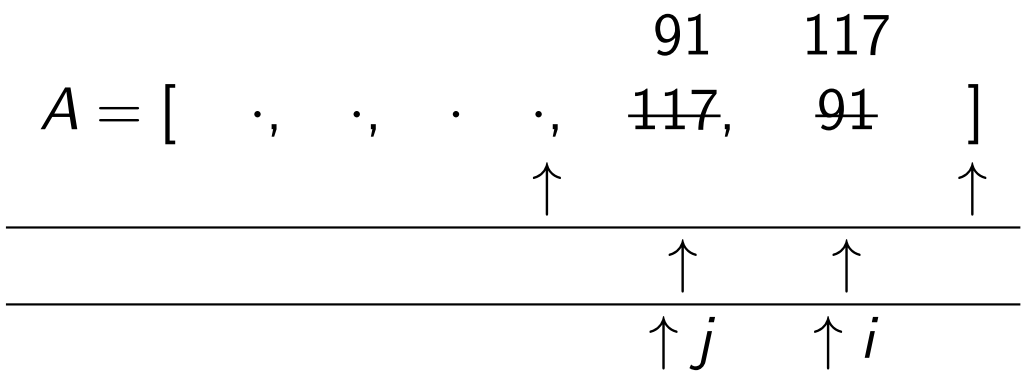
$A = [91, 117, 52, 11, 63, 49]$



$A = [91, 117, 52, 11, 63, 49]$

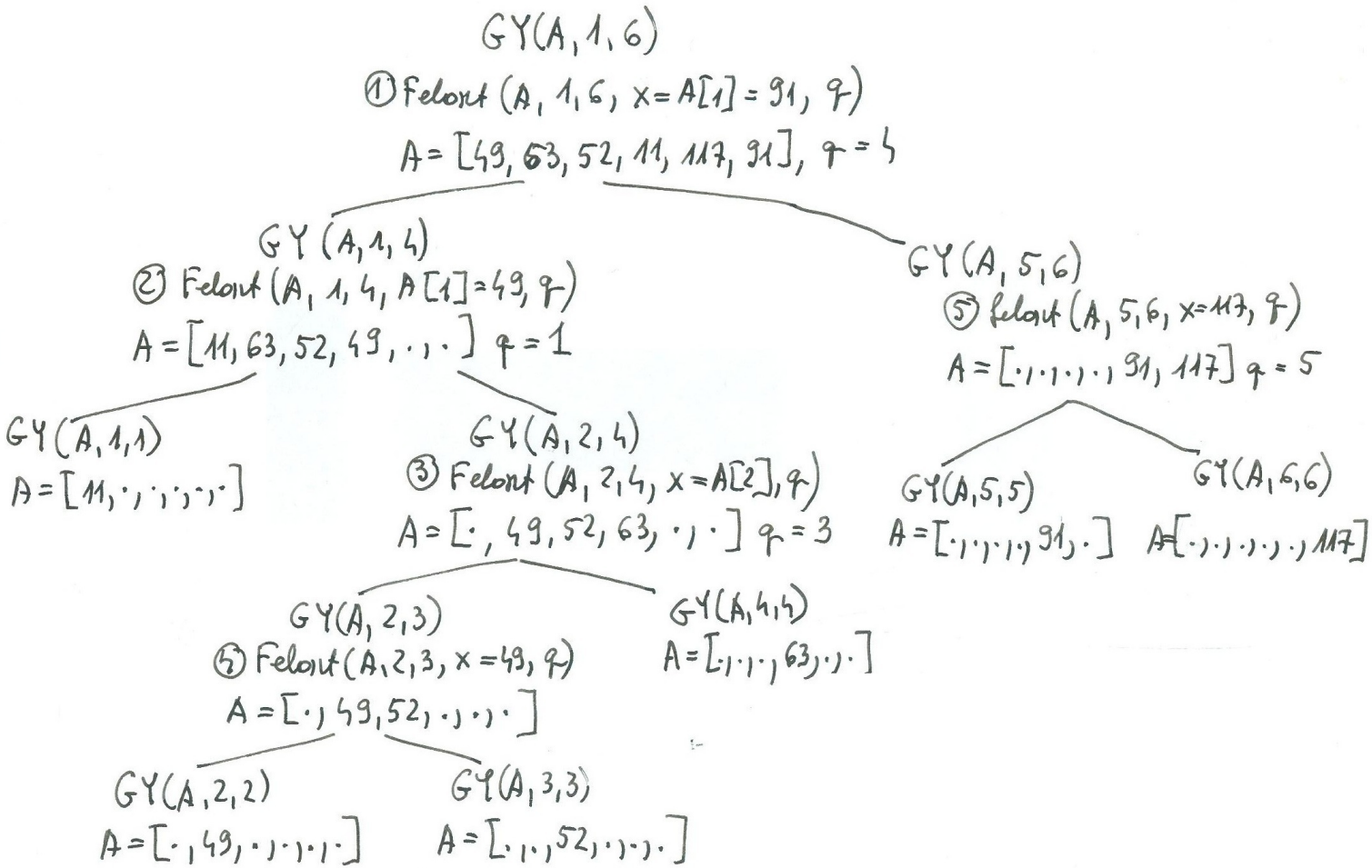


5.) FELOSZT$(A, 5, 6, x = 117, q)$

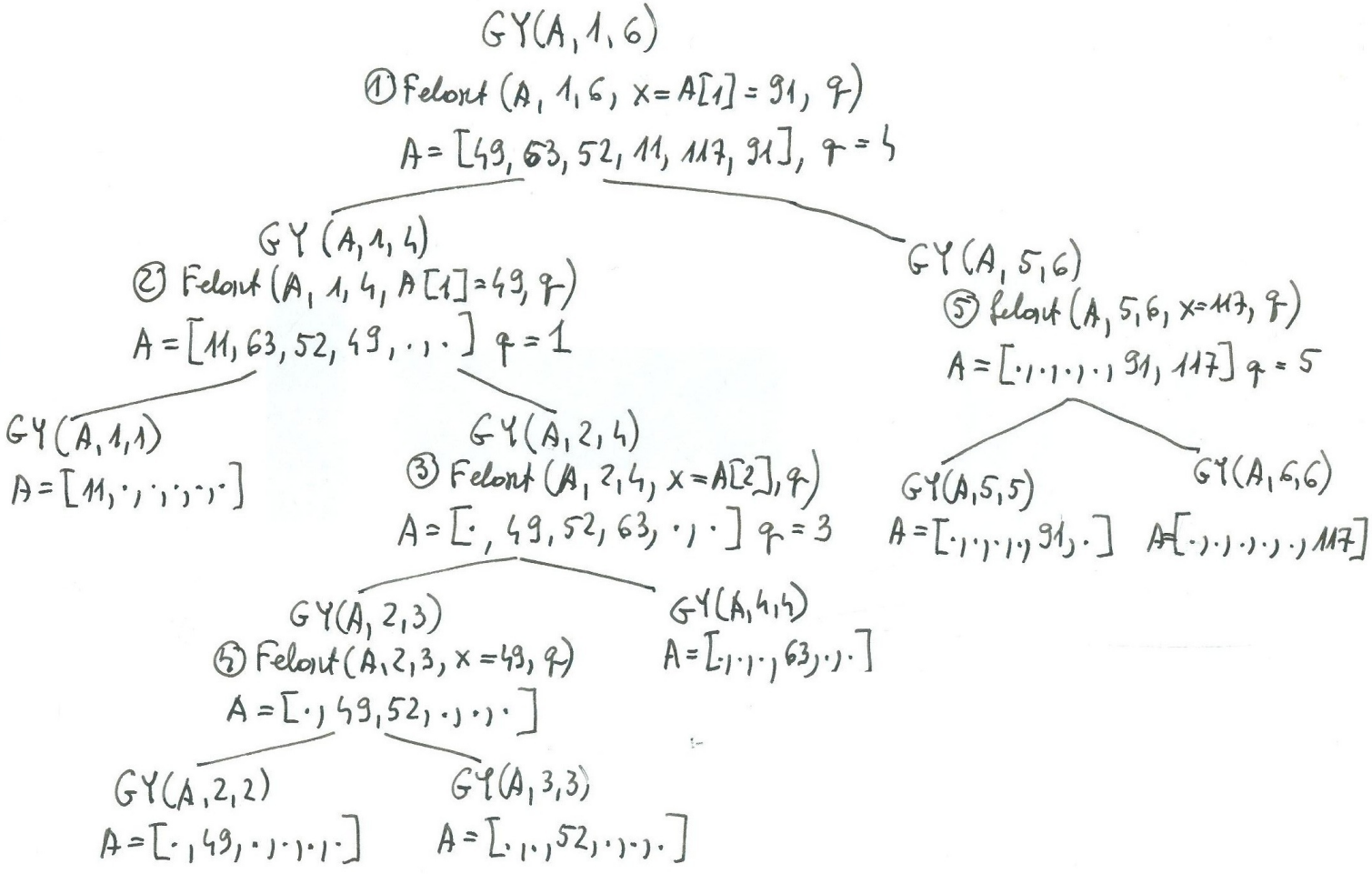


RETURN $(q \longleftarrow j = 5)$,

A = [91, 117, 52, 11, 63, 49]



$A = [91, 117, 52, 11, 63, 49]$

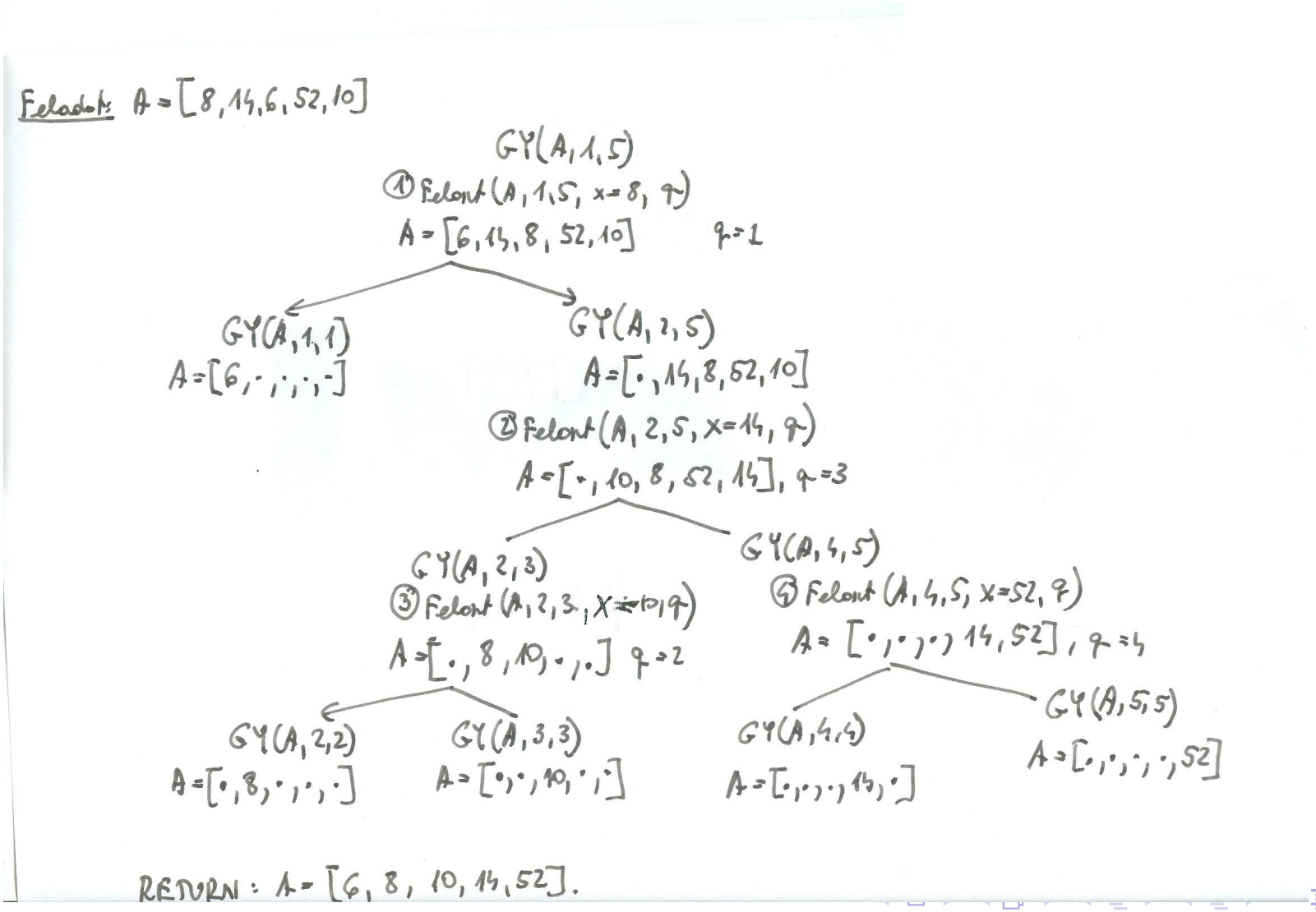


RETURN

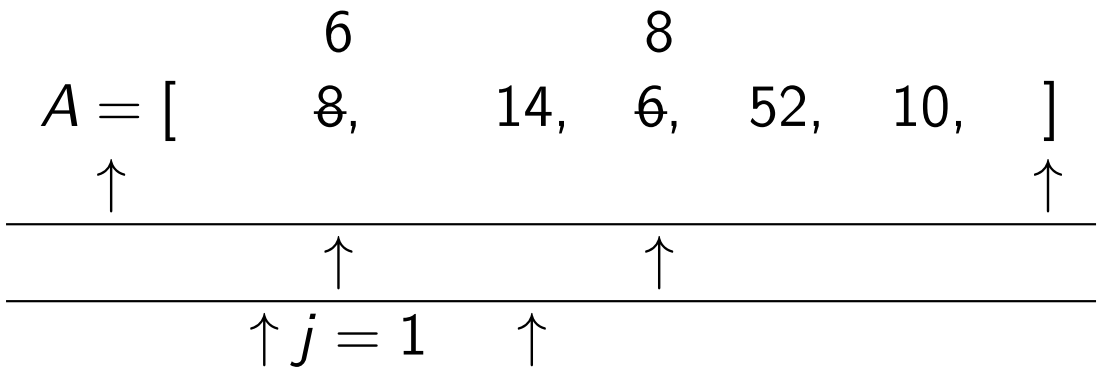
$A = [11, 49, 52, 63, 91, 117]$

Feladat.

A gyorsrendezés segítségével rendezzük az alábbi A tömböt.



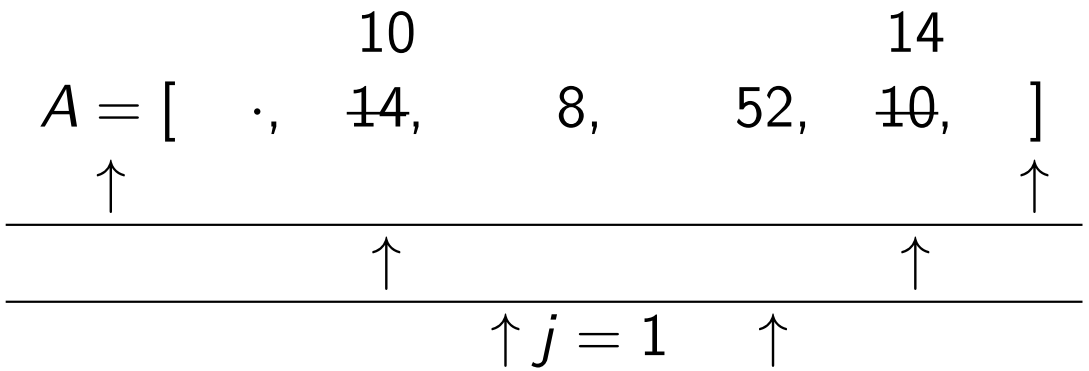
1.) FELOSZT$(A, 1, 5, x = 8, q)$



$x = 8$

RETURN $(q \longleftarrow j = 1)$,

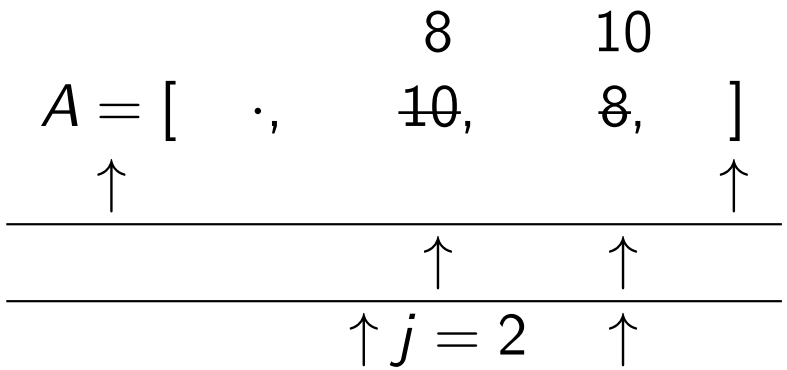
2.) FELOSZT$(A, 2, 5, x = 14, q)$



$x = 14$

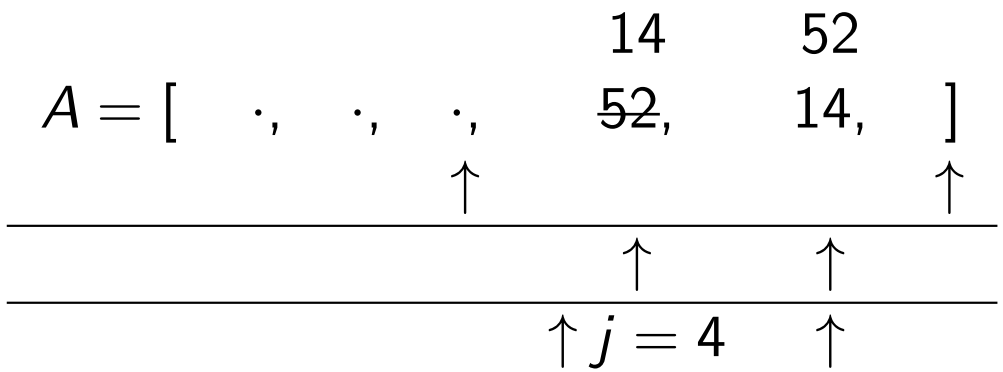
RETURN $(q \longleftarrow j = 1)$,

3.) FELOSZT$(A, 2, 3, x = 10, q)$



RETURN $(q \longleftarrow j = 2)$,

4.) FELOSZT$(A, 4, 5, x = 52, q)$



RETURN $(q \longleftarrow j = 4)$,

A leszámláló rendezés

A **Leszámláló rendezés** egy lineáris idejű rendezési algoritmus, ami nagyon kecsegtetően hangzik, azonban nagy ára van. Csak kis felső korláttal rendelkező pozitív (vagy akár nemnegatív) számok rendezésére alkalmas. Rendkívül ötletes, (mármint ha szabad ilyet mondanom). Hatalmas előnye, hogy stabil rendezés, azaz az azonos kulcsú elemek eredeti sorrendjét megőrzi. Ezért ezt a rendező algoritmust felhasználja más rendező algoritmus is.

A buborékrendezés

A buborékrendezés szintén négyzetes idejű. Rendkívül egyszerű. Tömb elemeinek a rendezésére alkalmas. Vagy lentről fölfelé vagy fentről lefelé haladunk végig a tömb elemein. Ha az egymást követő elemek nem a megfelelő sorrendben vannak, akkor megcseréljük őket. Így ha lentről felfelé haladunk, akkor a legnagyobb elem a tömb "tetejére" míg ha fentről lefelé haladunk, akkor a legkisebb elem a tömb legaljára kerül. Ez a buborék, amely vagy lentről felfelé, vagy fentről lefelé halad. Ezt az eljárást kell ismételni mindaddig, amíg rendezetté nem válik a tömb. Van lehetőség az algoritmus finomítására. Az egyik egy logikai típusú változó bevezetése, ami azt figyeli, hogy volt-e csere. Ha nem volt csere, akkor már rendezett a tömb. A másik módszer az, hogy váltogatjuk az összehasonlítás irányát. Ez utóbbi algoritmus a koktélrendezés, amivel azonban már nem foglalkozunk.

A beszúró rendezés

A **beszúró rendezés** egy négyzetes futási idejű rendezés. El sem tudok képzelni jobb rendezést, ha Zh-kat kell rendeznünk betűrend szerint.

A Shell rendezés

A **Shell rendezés** egy $n^{1,5}$ futásidejű algoritmus, amely gyorsabb, mint a négyzetes, de lassabb, mint az $n \log(n)$ idejű. Teljes neve: Shell rendezés csökkenő növekményekkel. Az algoritmus először a rendezendő tömb távoli elemeit mozgatja, ez a távolság a növekmény. Amikor a növekmény eléri az egyet, akkor egy beszúró rendezés valósul meg, de eddigre a tömb már majdnem rendezetté válik.

A négyzetes rendezés

A négyzetes rendezés szintén $n^{1,5}$ futási idejű.

Az összefésülő rendezés

Összefésülő rendezés. Ehhez az algoritmushoz szükségünk van egy összefésül algoritmusra. Az összefésül algoritmus rendezett tömbök rendezett összefésülésére szolgál. Tehát van két rendezett tömbünk, ezeket egy többé fésüljük össze úgy, hogy megint csak rendezett tömböt kapjunk. A Batcher-féle páros-páratlan összefűzés "csupán" egy gyors és ötletes módszer a rendezett összefésülés kivitelezésére. Az összefésülő rendezés egy rekurzív, $n \log(n)$ futási idejű algoritmus. Úgy múködik, hogy a tömböt (valahol középen) kettévágjuk. A kettévágást mindaddig folytatjuk, amíg egy elemű résztömböket nem kapunk. Az egyelemű résztömbök rendezettek. Ezt követően a rendezett résztömböket rendezett módon összefűzzük. Ezáltal a tömb rendezetté válik.

A számjegyes, vagy radix rendezés

A **számjegyes, vagy radix rendezés** szintén lineáris futási idejű. Ez a rendezés alkalmas ugyanannyi számjegyből álló számok rendezésére. A radix rendezés stabil rendezés, tehát az azonos kulcsú elemek eredeti sorrendjét megőrzi. A radix rendezéshez szükség van stabil rendezésre, például a leszámláló rendezésre. Az algoritmus roppant egyszerű. A stabil rendezéssel a rendezendő számokat a legkisebb helyiértékű (tehát utolsó jegyük) szerint rendezzük. Ezt követően a számjegyeket a második, harmadik, legkisebb, végül a legnagyobb helyiértékű jegyük alapján rendezzük. Ezáltal a számok sorozata rendezetté válik.

Az edényrendezés

Az edényrendezés is lineáris idejű. Ez az algoritmus is akkor fut gyorsan, ha a rendezendő elemek (matematikailag valós számok) egyenletes eloszlásúak egy $[a, b[$ intervallumban.
(Az egyenletes eloszlás azt jelenti, hogy nagy elemszám esetén az adatelemek részintervallumba esésének valószínűsége arányos a részintervallum hosszával.)
Mivel egy $[a, b[$ intervallum mindig áttranszformálható a $[0, 1[$ intervallumba, így csak a $[0, 1[$ intervallum esetét tárgyaljuk.
A $[0, 1[$ intervallumot osszuk fel n egyenlő hosszúságú részintervallumra. Ezek a kis intervallumok az edények.
Létrehozunk egy n elemű tömböt, amely láncolt listák fejeit tartalmazza.
Van egy ügyes függvényünk, amely egy tetszőleges $[0, 1[$-beli valós száról kapásból megmondja, hogy melyik $\frac{1}{n}$ hosszúságú részintervallumba tartozik, majd beszúrjuk a megfelelő láncolt listába. Ezután a láncolt listákat rendezzük, majd egymás után fűzzük.

A kupac rendezés

A **kupac rendezés** egy $n \log(n)$ futási idejű rendezés, amely a kupac adatstruktúrán alapul. a kupac adatstruktúra egy fagráf, amely sok érdekes tulajdonsággal rendelkezik, például a gráf gyökere a legnagyobb elem. A gyökérelem lesz a tömb legutolsó eleme. Ezáltal elromlik a kupac struktúra. Helyreállítjuk a kupac struktúrát és iteráltan alkalmazzuk az eljárást.

KÖSZÖNÖM A FIGYELMET!