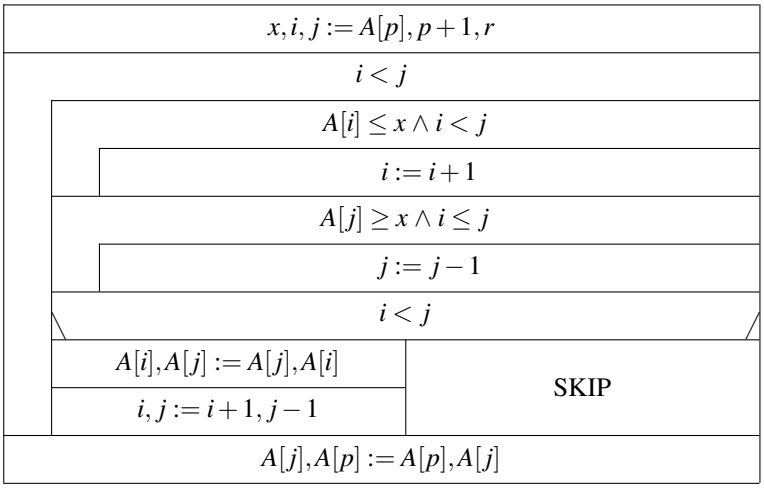
*Feladat:* Adott az egész számokat tartalmazó x vektor. Permutáljuk a vektor elemeit (helyben!) úgy, hogy a vektor egy eleme a monoton rendezés szerinti helyére kerüljön, azaz ne előzze meg őt nála nagyobb elem és utána ne legyen nála kisebb!

Először egy kicsit egyszerűbb feladatot oldunk meg: a vektor egy konkrét elemét visszük a helyére.

*Megjegyzés:* Ezt a feladatot és első megoldását eredetileg Hoare vezette be, a gyorsrendezés részeként. Ő az alábbi megoldást adta (aminek helyességét azonban nem bizonyítjuk, csak egy állapottér magyarázó táblázattal szolgálunk):

| | |
|---|---|
| A | a rendezendő vektor |
| i | balról lépkedő mutató |
| j | jobbról lépkedő mutató (a végén mutatja a pivot elem új indexét) |
| p | a vektor alsó indexe, egyben a pivot elem indexe |
| r | a vektor felső indexe |




Mi egy másik megoldást, nevezetesen Lomuto megoldását vesszük alaposan szemügyre. Ez annyiban is eltér Hoare megoldásától, hogy az első helyett az utolsó elemet teszi helyre.

*Specifikáció:*
$A = \underset{x}{\mathbb{V}} \times \underset{i}{\mathbb{Z}} \times \underset{j}{\mathbb{Z}}$
$\mathbb{V} = \text{vect}(\mathbb{Z},\mathbb{Z})$
$B = \underset{x'}{\mathbb{V}}$
$Q = (x = x')$
$R = (x \in \text{perm}(x') \wedge i \in [x.lob..x.hib] \wedge x_i = x'.hiv \wedge \forall k \in [x.lob..i-1] : x_k \leq x_i \wedge \forall k \in [i+1..x.hib] : x_k > x_i)$

$x \in \text{perm}(x')$ azt írja le, hogy az $x$ vektor elemei ugyanazok, mint az $x'$-é, csupán a sorrendjük különbözik.

*Megoldás:*
A megoldást könnyedén (egy megengedett szimultán értékadással elérhetnénk), ha már teljesülne $R'$ állapot:
$R' = (x \in \text{perm}(x') \wedge i \in [x.lob-1..x.hib-1] \wedge x.hiv = x'.hiv \wedge \forall k \in [x.lob..i] : x_k \leq x.hiv \wedge \forall k \in [i+1..x.hib-1] : x_k > x.hiv)$

Tűzzük ki tehát új utófeltételnek $R'$-t és ha oda eljutunk, akkor a szekvencia levezetési szabályát alkalmazzuk majd a meglévő programra és az $R$-be vezető $i,x_{i+1},x.hiv := i+1,x.hiv,x_{i+1}$ értékadásra, hogy a teljes megoldó programot kapjuk.

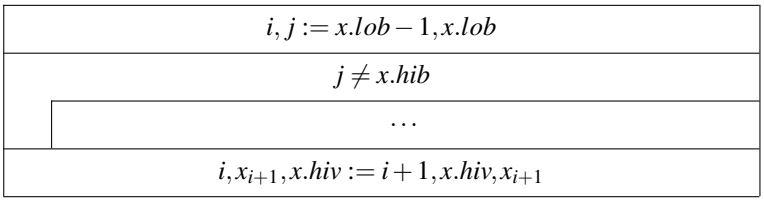
Most írjuk fel az új utófeltételhez elvezető ciklus invariánsát:
$P = (x \in \text{perm}(x') \wedge j \in [x.lob..x.hib] \wedge i \in [x.lob-1..j-1] \wedge x.hiv = x'.hiv \wedge \forall k \in [x.lob..i] : x_k \leq x.hiv \wedge \forall k \in [i+1..j-1] : x_k > x.hiv)$

Ellenőrízzük le a ciklus feltételeit:
1. $\boxed{Q \Rightarrow P}$

$Q' = (Q \wedge i = x.lob-1 \wedge j = x.lob)$

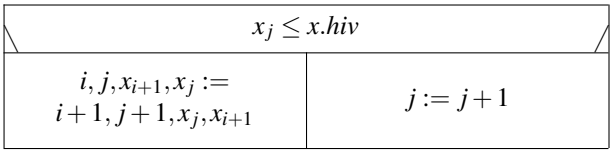
Látható, hogy $Q' \Rightarrow P$ és $Q \Rightarrow \text{lf}(i, j := x.lob - 1, x.lob, Q') = Q$.

2. $\boxed{P \wedge \neg\pi \Rightarrow R'}$
Ebből a pontból kapjuk meg a ciklusfeltételt, hiszen $P$ és $R'$ összehasonlításából $\neg\pi$-re $x.hib = j$ adódik. Tehát $\pi = (j \neq x.hib)$.




3. $\boxed{P \wedge \pi \Rightarrow t > 0}$
Jelen esetben ez: $j \in [x.lob..x.hib - 1] \wedge \cdots \Rightarrow t > 0$. Tehát $t := x.hib - j$ egy megfelelő terminálófüggvény.
4. $\boxed{P \wedge \pi \Rightarrow \text{lf}(S_0, P)}$

Nézzük az alábbi struktogramot $S_0$-ként:



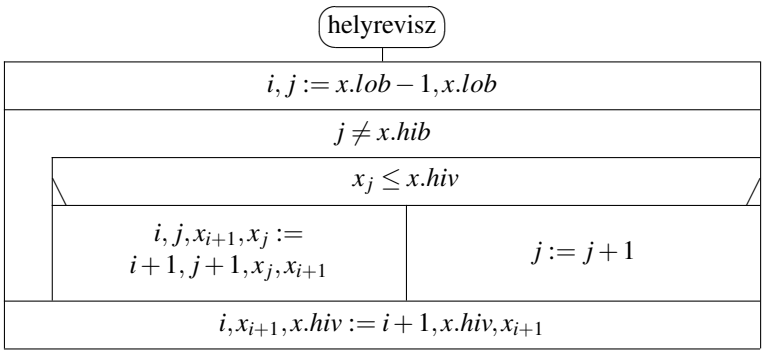

Annak bizonyítását, hogy ez az elágazás mindig növeli $j$ értékét (és így csökkenti a terminálló függvényt), az olvasóra bízom.

4. $\boxed{P \wedge \pi \Rightarrow \text{lf}(S_0, P)}$
Ezen állítás igazolásához az elágazás levezetési szabályának feltételeit kell ellenőríznünk:
4/1. $\boxed{P \wedge \pi \Rightarrow \left(\bigvee_{i=1}^{n} \pi_i\right)}\surd$

4/2. $\boxed{\forall i \in [1..n] : P \wedge \pi \wedge \pi_i \Rightarrow \text{lf}(S_i, P)}$

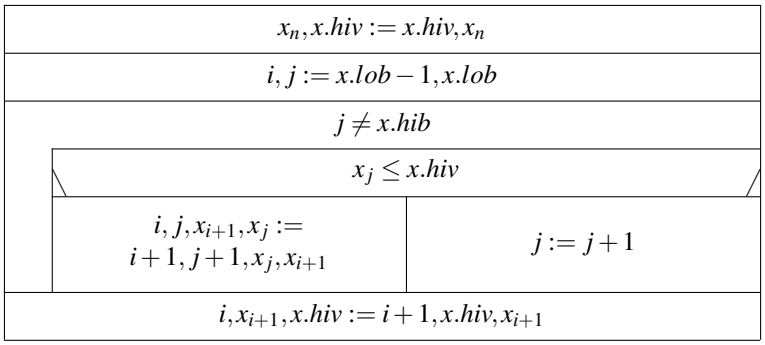
1. $x_j \leq x.hiv$:
   $x \in \text{perm}(x') \wedge j \in [x.lob..x.hib - 1] \wedge i \in [x.lob - 1..j - 1] \wedge x.hiv = x'.hiv \wedge \forall k \in [x.lob..i] : x_k \leq x.hiv \wedge \forall k \in [i + 1..j - 1] : x_k > x.hiv \wedge x_j \leq x.hiv$
   $\overset{?}{\Rightarrow}$
   $\text{lf}(i, j, x_{i+1}, x_j := i + 1, j + 1, x_j, x_{i+1}, P) = (y \in \text{perm}(x') \wedge j + 1 \in [y.lob..y.hib] \wedge i + 1 \in [y.lob - 1..j] \wedge y.hiv = x'.hiv \wedge \forall k \in [y.lob..i + 1] : y_k \leq y.hiv \wedge \forall k \in [i + 2..j] : y_k > y.hiv$, ahol $y.hib = x.hib \wedge y.lob = x.lob \wedge \forall h \in [x.lob, x.hib] \setminus \{i + 1, j\} : y_h = x_h \wedge y_{i+1} = x_j \wedge y_j = x_{i+1}$. $\surd$
2. $x_j > x.hiv$:
   $x \in \text{perm}(x') \wedge j \in [x.lob..x.hib - 1] \wedge i \in [x.lob - 1..j - 1] \wedge x.hiv = x'.hiv \wedge \forall k \in [x.lob..i] : x_k \leq x.hiv \wedge \forall k \in [i + 1..j - 1] : x_k > x.hiv \wedge x_j > x.hiv$
   $\overset{?}{\Rightarrow}$
   $\text{lf}(j := j + 1, P) = (x \in \text{perm}(x') \wedge j + 1 \in [x.lob..x.hib] \wedge i \in [x.lob - 1..j] \wedge x.hiv = x'.hiv \wedge \forall k \in [x.lob..i] : x_k \leq x.hiv \wedge \forall k \in [i + 1..j] : x_k > x.hiv)\surd$

Ezzel megkaptuk a szakmában helyrevisznek becézett feladat megoldását (ez volt a mi egyszerűsített feladatunk).




Ám nekünk az volt a feladatunk, hogy egy adott elemet vigyünk a helyére. Egészítsük ki a specifikációt ennek megfelelően:
$A = \underset{x}{\mathbb{V}} \times \underset{n}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{j}{\mathbb{Z}}$
$\mathbb{V} = \text{vect}(\mathbb{Z},\mathbb{Z})$
$B = \underset{x'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}}$
$Q = (x = x' \wedge n = n' \wedge n \in [x.lob..x.hib])$
$R = (n = n' \wedge x \in \text{perm}(x') \wedge i \in [x.lob..x.hib] \wedge x_i = x'_{n'} \wedge \forall k \in [x.lob..i-1] : x_k \leq x_i \wedge \forall k \in [i+1..x.hib] : x_k > x_i)$

A szekvencia levezetési szabályával bizonyítható, hogy ezt a feladatot az alábbi program oldja meg:




A most megismert algoritmus legnagyobb jelentősége, hogy a jelenleg ismert leggyorsabb (és egyben „leghóbortosabb”) rendező algoritmus szerves része. Az érdekesség kedvéért itt közöljük egy olyan verzióját ennek a rendezőnek, ami a modell kereteibe igazából (például a rekurzió miatt) nem illik, de intuitíven megérthető. A rendező mély tárgyalását az Algoritmusok és adatszerkezetek című tárgy keretei között hallgathatjuk meg.