Szekvenciális adatszerkezetek

4. előadás

Tömb

Tömbök – ADT

- Definíció
  - Az E alaptípusú k $(k \geq 1)$ dimenziós T tömbtípus
    - Legyen $I = I_1 \times I_2 \times \cdots \times I_k$ egy indexhalmaz, ahol $\forall j \in [1,k]: I_j = [1, n_j]$- a kezdőérték lehetne m_j is.
    - Az $A \in T$ tömbnek $N = n_1 * n_2 * \cdots * n_k$ eleme van $\{a_1, a_2, \ldots, a_N\}$
- Mindig van egy $f: I \rightarrow \{a_1, a_2, \ldots, a_N\}$ egy-egy értelmű leképezés
- Jelölés
  - $A[i_1, i_2, \ldots, i_k]$ a tömbnek az $i_1, i_2, \ldots, i_k$ indexek által azonosított eleme

Tömbök – ADT

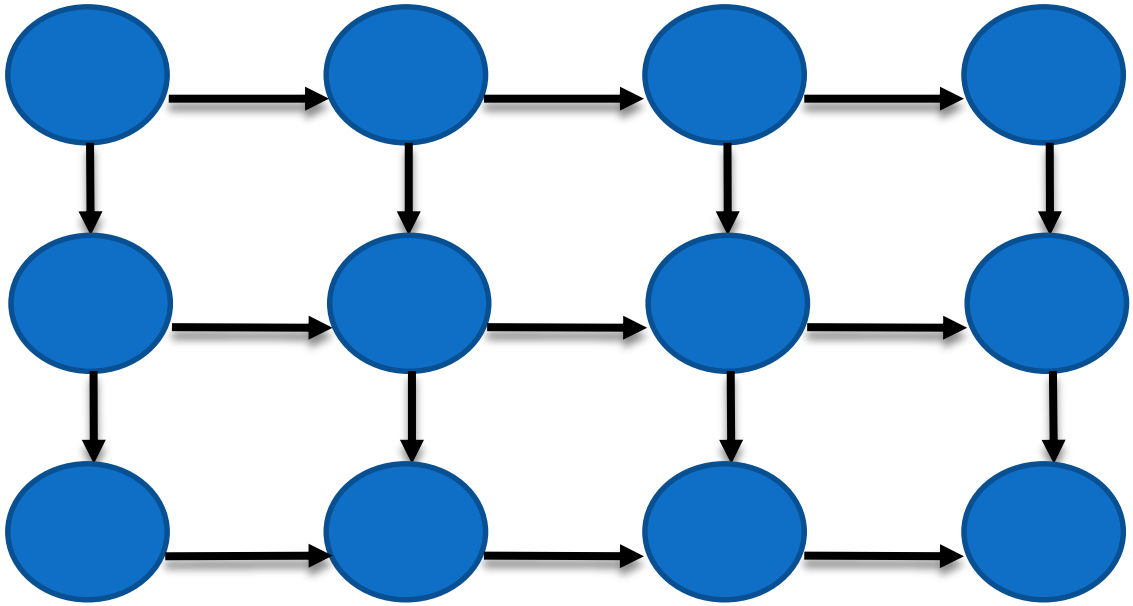
- Invariáns is adható – speciális megszorítás
  - Példa: szimmetrikus, alsó Δ, ritka, stb.
- Műveletek
  - Indexelés: az $i_1, i_2, \dots, i_k$ indexhez tartozó $A[i_1, i_2, \dots, i_k]$ elem kiválasztása
  - Elemmódosítás – értékadás: $A[i_1, i_2, \dots, i_k] := a$
  - Értékadás $A := B$
- Elnevezés
  - Vektor: $k = 1$
  - Mátrix: $k = 2$

Tömbök – ADS

- Nem kötelező szerkezetet (rákövetkezést) definiálni az elemek között
- Elfogadott a köv_j reláció bevezetése: $\forall j \in [1, k]$-ra
  - $\text{köv}_j(A[i_1, \ldots, i_j, \ldots, i_k]) = A[i_1, \ldots, i_{j+1}, \ldots, i_k]$, ha $i_j < n_j$, egyébként köv_j nem definiált.
  - Egy belső elemnek minden dimenzióban van rákövetkezője

Tömbök – ADS

- A legalább 2 dimenziós tömböt ortogonális adatszerkezetnek nevezik.
- $k = 2$-re a gráf:

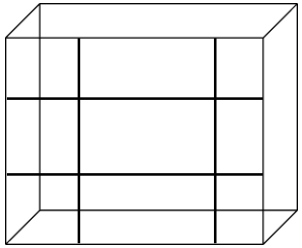


Reprezentáció

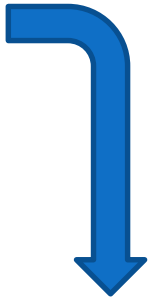
- Tömbök aritmetikai ábrázolása:
- Egy *k* dimenziós tömböt szeretnénk elhelyezni egy alkalmas méretű egydimenziós tömbben (vektorban), és megadjuk a leképezés címfüggvényét.
- Az elhelyezés – általában
  - sorfolytonos (SF)
  - oszlopfolytonos (OF)

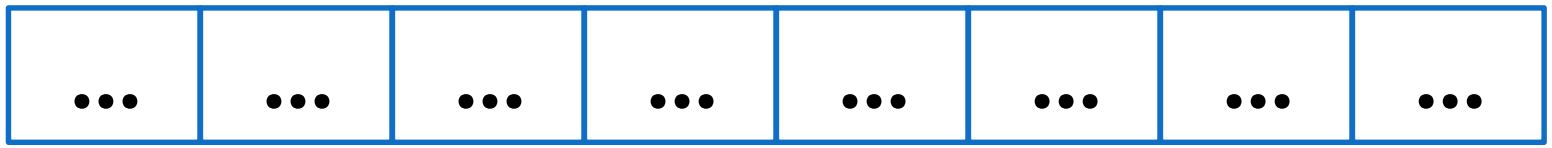
Reprezentáció

- Aritmetikai ábrázolás
  - Tömb



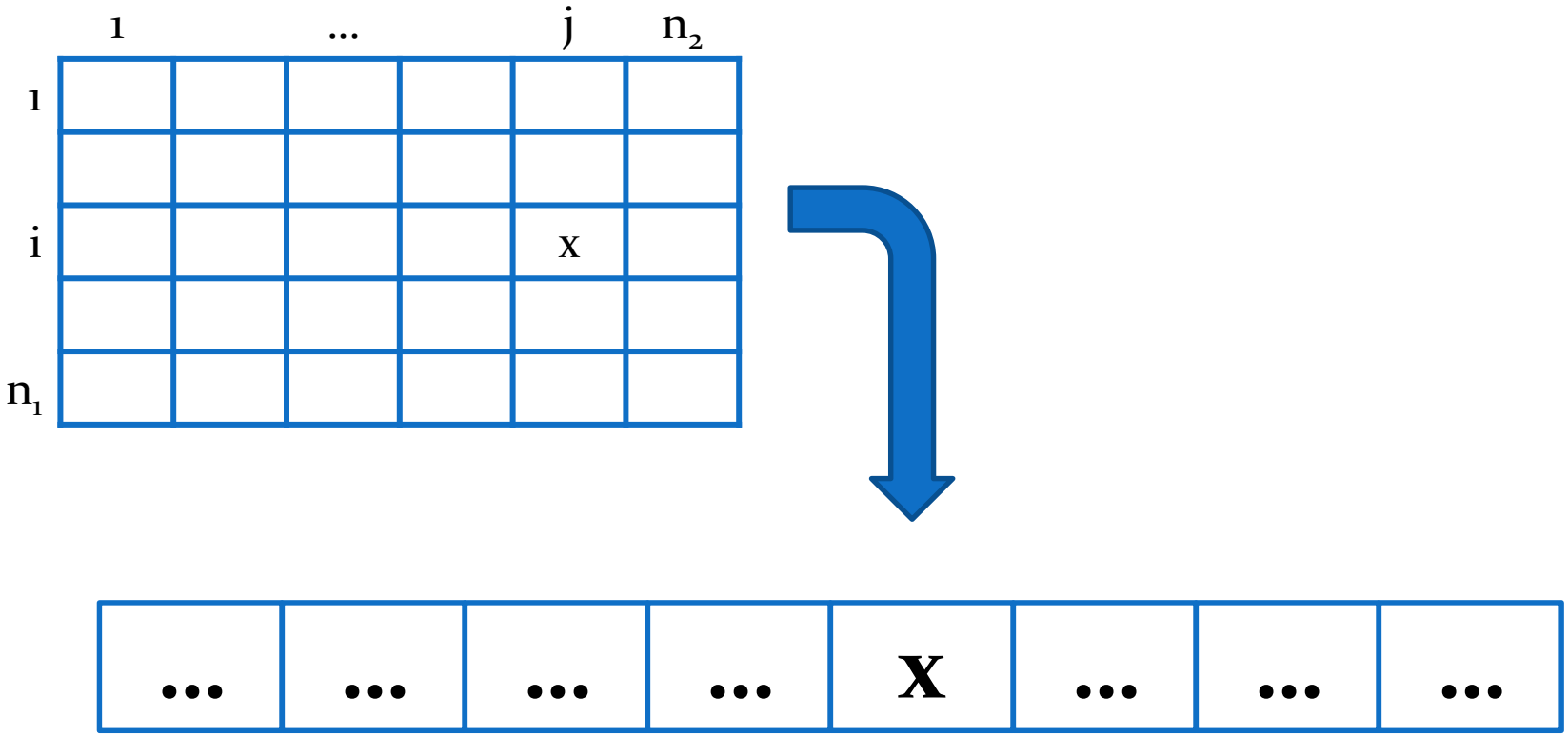
  - Leképezés





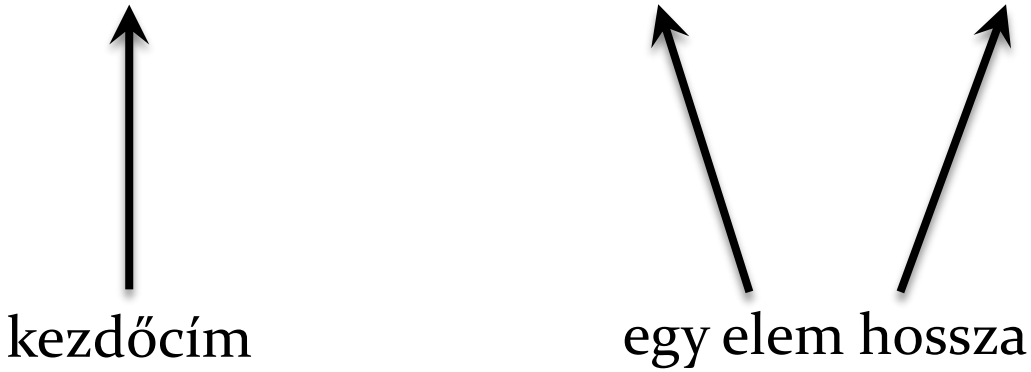
Reprezentáció

- Mátrix elhelyezése vektorban



Reprezentáció

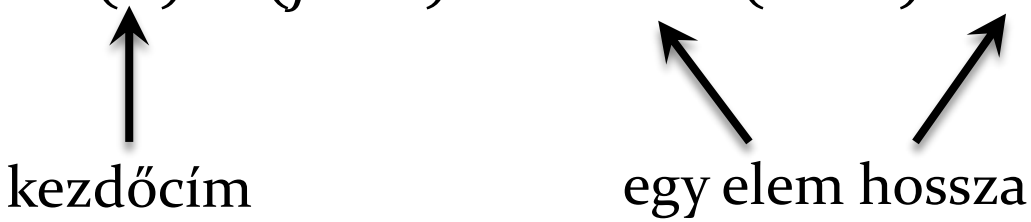
- Indexfüggvény: $\forall i \in [1, n_1], \forall\ j \in [1, n_2]$
  - SF: $\text{ind}(A[i,j]) = (i-1) * n_2 + j$
  - OF: $\text{ind}(A[i,j]) = (j-1) * n_1 + i$
- Címfüggvény: $\forall i \in [1, n_1], \forall\ j \in [1, n_2]$
  - SF: $\text{cím}(A[i,j]) = \text{cím}(A) + (i-1) * n_2 * h + (j-1) * h$
  - OF: $\text{cím}(A[i,j]) = \text{cím}(A) + (j-1) * n_1 * h + (i-1) * h$



Reprezentáció

Szokták úgy is tekinteni, hogy a mátrixnak m sora és n oszlopa van

- Indexfüggvény: $\forall i \in [1,m], \forall\ j \in [1,n]$
  - SF: $\text{ind}(A[i,j]) = (i-1) * n + j$
  - OF: $\text{ind}(A[i,j]) = (j-1) * m + i$
- Címfüggvény: $\forall i \in [1,m], \forall\ j \in [1,n]$
  - SF: $\text{cím}(A[i,j]) = \text{cím}(A) + (i-1) * n * h + (j-1) * h$
  - OF: $\text{cím}(A[i,j]) = \text{cím}(A) + (j-1) * m * h + (i-1) * h$



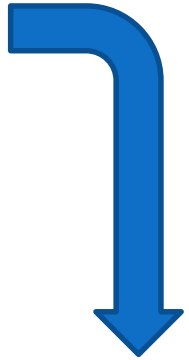
Reprezentáció

- Az R invariáns leggyakrabban a tömb alakját módosítja
  - Például megadja a 0 elemek helyét, és a nem-nulla elemek határozzák meg a tömb speciális alakját.
- Konvenció
  - a gyakran szereplő elemeket (ez általában a 0) is tároljuk egy példányban az egy dimenziós tömbben, és az erre vonatkozó hivatkozást beépítjük a címfüggvénybe

Reprezentáció

- Tridiagonális mátrix

$a_{1,1}$	$a_{1,2}$	0	0	0	0			
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	0	0	0			
0	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	0	0			
0	0	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$	0			
0	0	0	$a_{5,4}$	$a_{5,5}$	$a_{5,6}$	…		
					…			

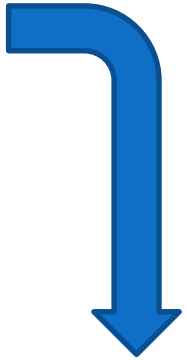


$a_{1,1}$	$a_{1,2}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$	$a_{5,4}$	$a_{5,5}$	$a_{5,6}$	…		

Reprezentáció

- Tridiagonális mátrix – „0” elem tárolásával

$a_{1,1}$	$a_{1,2}$	0	0	0	0			
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	0	0	0			
0	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	0	0			
0	0	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$	0			
0	0	0	$a_{5,4}$	$a_{5,5}$	$a_{5,6}$	…		
					…			



0	$a_{1,1}$	$a_{1,2}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$	$a_{5,4}$	$a_{5,5}$	$a_{5,6}$	…	

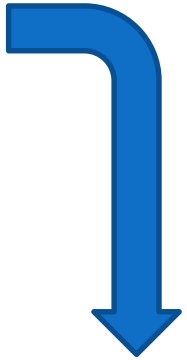
Reprezentáció

- Ha $|i-j| \leq 1$, akkor:
  - $\text{ind}(A[i,j]) = (i-1) * 3 - 1 + \begin{cases} 1, \text{ha } i > 1 \\ 2, \text{ha } i = j \\ 3, \text{ha } i < j \end{cases}$
- Ha a „0” elemet is tároljuk, a vektor elején
  - $\text{ind}(A[i,j]) = \begin{cases} (i-1) * 3 + 1, \text{ha } i = j + 1 \\ (i-1) * 2 + 1, \text{ha } i = j \\ i * 3, \text{ha } i + 1 = j \\ 1, \text{különben} \end{cases}$

Reprezentáció

- Alsó háromszög mátrix

$a_{1,1}$	0	0	0	0			
$a_{2,1}$	$a_{2,2}$	0	0	0			
$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	0	0			
$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	0			
$a_{5,1}$	$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	$a_{5,5}$	…		
				…			



$a_{1,1}$	$a_{2,1}$	$a_{2,2}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{5,1}$	$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	$a_{5,5}$	…	0

Reprezentáció

- Elemeinek száma $m = n * \frac{(n+1)}{2} + 1$
  - $\text{ind}(A[i,j]) = \begin{cases} i * \frac{(i-1)}{2} + j, \text{ha } i \geq j \\ n * \frac{(n+1)}{2} + 1, \text{ha } i < j \end{cases}$

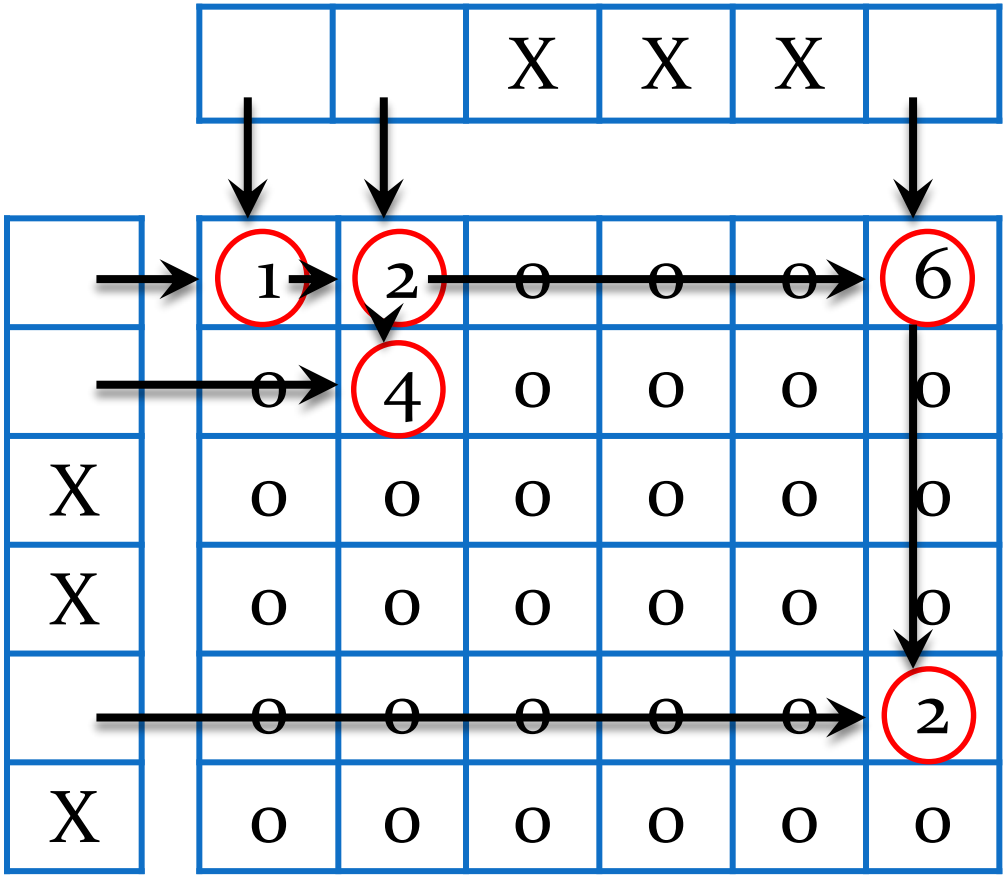
Hézagosan kitöltött mátrixok

1	2	0	0	0	6
0	4	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	2
0	0	0	0	0	0

háromsoros reprezentáció

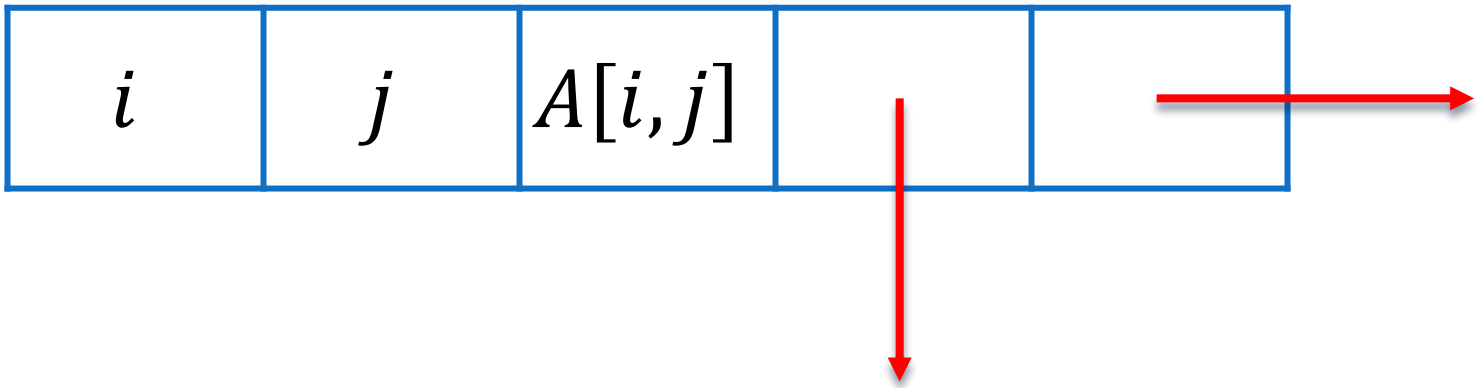
sorindex	1	1	1	2	5
oszlopindex	1	2	6	2	6
érték	1	2	6	4	2

Láncolt, illetve vegyes ábrázolás

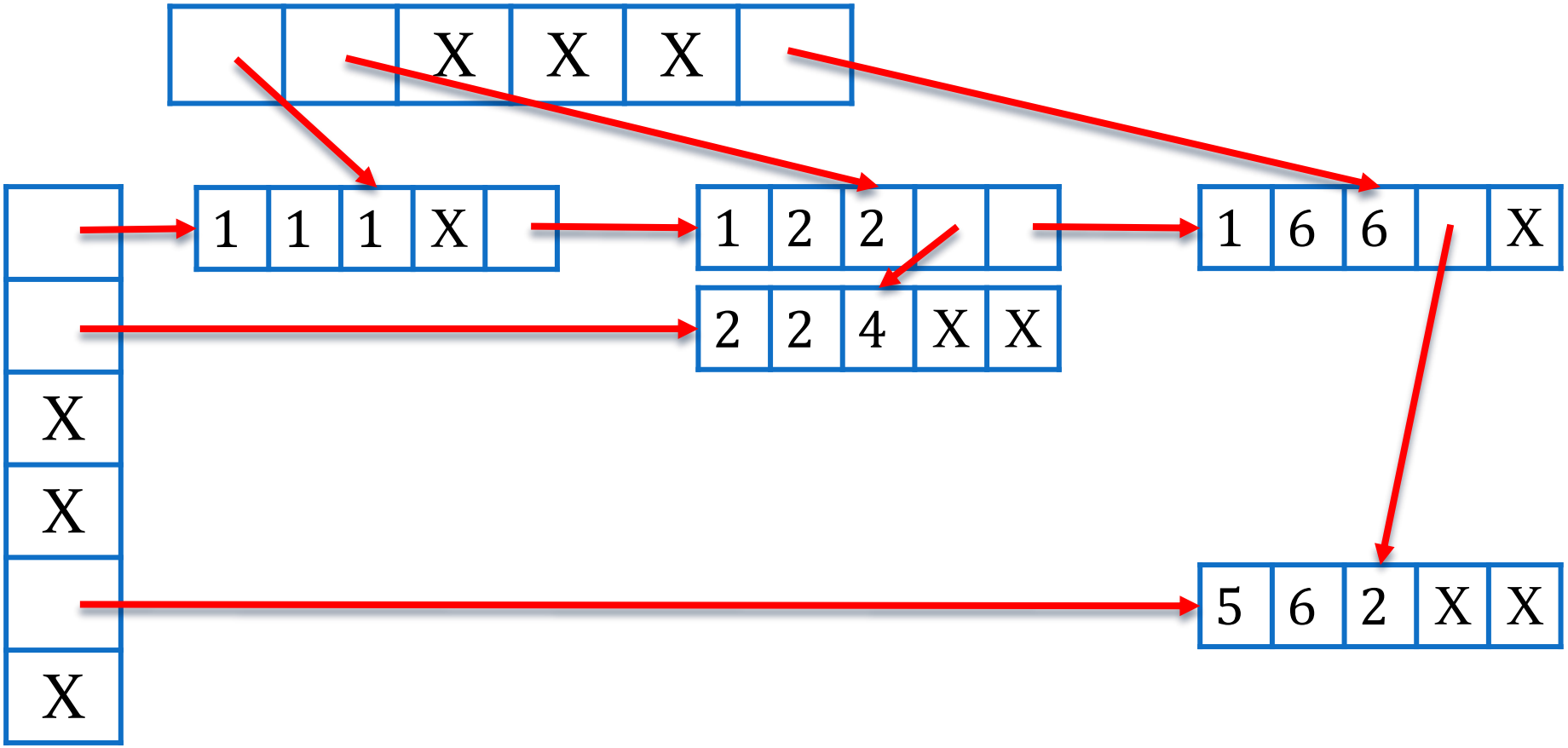


Láncolt, illetve vegyes ábrázolás

- Egy elem ábrázolása:



Láncolt, illetve vegyes ábrázolás



Láncolt, ill. vegyes ábrázolás:

- Mikor előnyös?
  - a mátrix legyen $m * n$ -es
  - a nem nulla elemek száma k
  - legyen egy érték helyfoglalása h byte
  - egy mutató helyfoglalása p byte
  - egy index helyfoglalása i byte
- A számítás
  - $(h + (2 * i) + (2 * p)) * k + (m + n) * p \ll m * n * h$

Feladatok

- Gyakorlásnak – gondolkodásra
  - Írjuk meg azt az eljárást (függvényt), ami visszaadja $A[i, j]$ értékét
  - Írjuk meg azt az eljárást, ami módosítja $A[i, j]$ értékét! (nulláról nem nullára, nem nulláról nullára)
  - Adjunk össze két ritka mátrixot!
  - Adjuk meg a szimmetrikus mátrix aritmetikai ábrázolását!

Szekvenciális adatszerkezetek

Szekvenciális adatszerkezetek

- Definíció: A szekvenciális adatszerkezet olyan $\langle A, R \rangle$ rendezett pár amelynél az $R \subseteq (A \times A)$ reláció tranzitív lezártja teljes rendezési reláció
- Az $R \subseteq (A \times A)$ reláció tranzitív lezártja az a reláció, mely tranzitív, tartalmazza R-et, és a lehető legkevesebb elemet tartalmazza
- Megadása:
  1. $R' = R \cup (R \circ R)$
  2. Ha $R \neq R'$, akkor folyt. 1.-nél, különben $R' = R_T$, a tranzitív lezárt

Szekvenciális adatszerkezetek

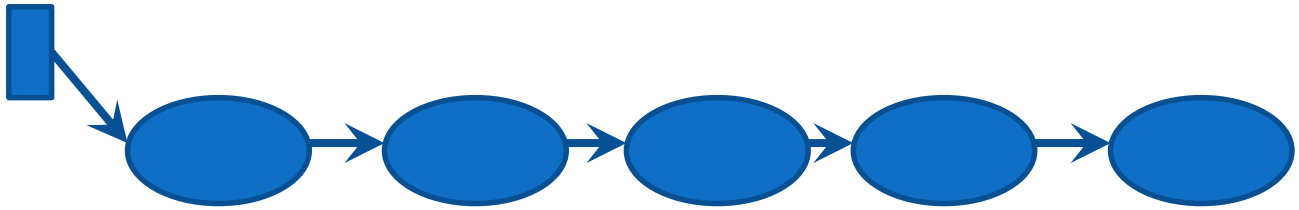
- Szekvenciális adatszerkezetben az egyes adatelemek egymás után helyezkednek el
  - Van egy logikai sorrendjük
- Az adatok között egy-egy jellegű a kapcsolat
  - Minden adatelem csak egy helyről érhető el és az adott elemtől csak egy másik látható
- Két kitüntetett elem: az első és az utolsó

Szekvenciális adatszerkezetek

- Ez egy homogén adatszerkezet, azaz azonos típusú véges adatelemek sorozata
  - Jelölése : $L = (a_1, a_2, \ldots\ a_n)$
  - Ha $n = 0$, akkor $L = ()$ az üres lista.
- A láncolt lista olyan adatszerkezet, amelynek minden eleme tartalmaz egy (vagy több) mutatót (hivatkozást) egy másik, ugyanolyan típusú adatelemre - ez a „következő” elem logikailag
- A lánc első elemének a címét a lista feje tartalmazza
  A listafej nem tartalmaz információs részt
- A lánc végét az jelzi, hogy az utolsó elemben a rákövetkező elem mutatója üres

Az adatszerkezetek osztályozása

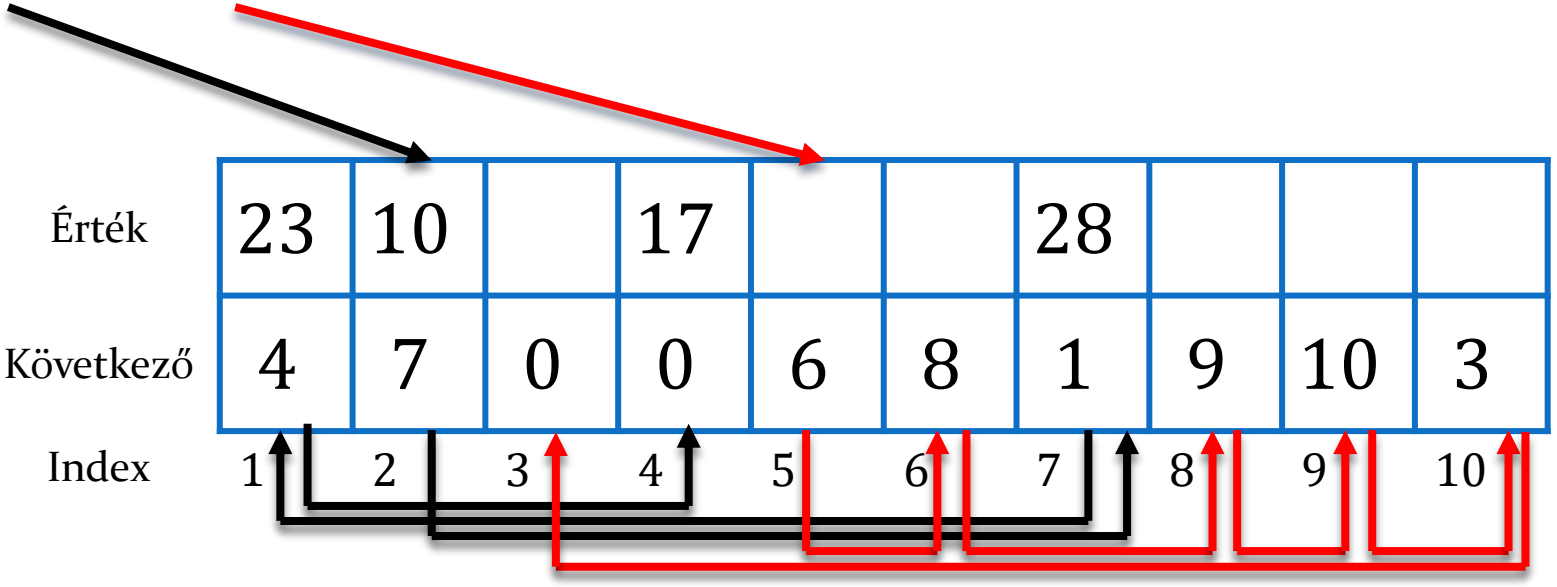
- Intuitív ADT és ADS szint



Végigmehetünk az elemeken egymás után.
Lehetőség van a módosítás, törlés, beszúrás műveletekre

Reprezentációs szint

- Statikus ábrázolás:
  tömbben, a logikai sorrendet indexek mutatják, a szabad helyek is listában:
- L: 2 SZH: 5



Dinamikus láncolt ábrázolás

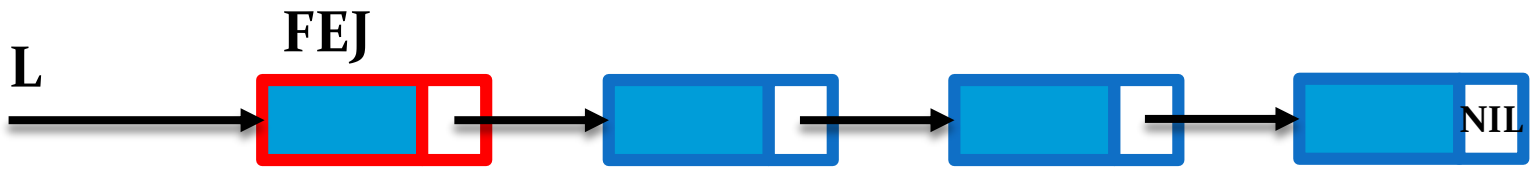
- Az adatok száma nem ismert előre.
- Nem tudunk, vagy nem akarunk feleslegesen helyet foglalni az adatoknak.
- A feladat dinamikusan változik.

Dinamikus láncolt ábrázolás

Láncolt ábrázolás

- Egyirányú láncolt lista
  - Fejelem nélkül

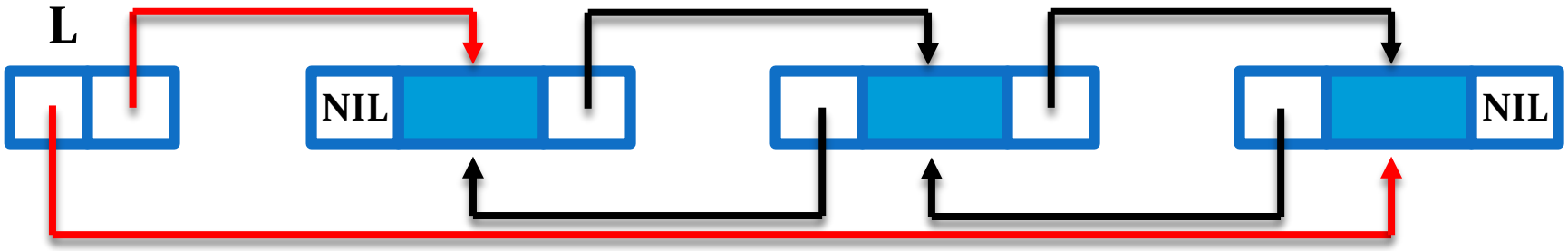
  - Fejelemmel: fejelem mindig létezik, ha üres a lista, akkor is



- Mindig van egy aktuális elemre mutató is, ez része a megvalósításnak

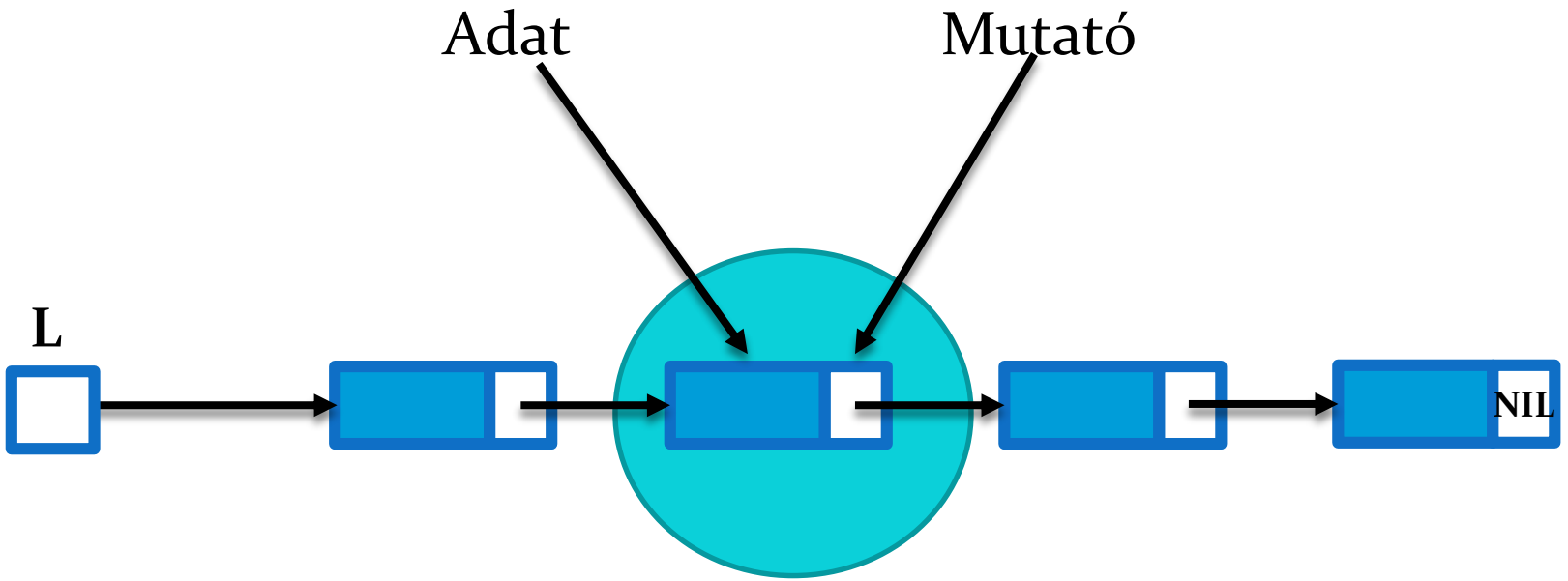
Láncolt ábrázolás

- Kétirányú láncolt lista



Láncolt ábrázolás

- A lista eleme

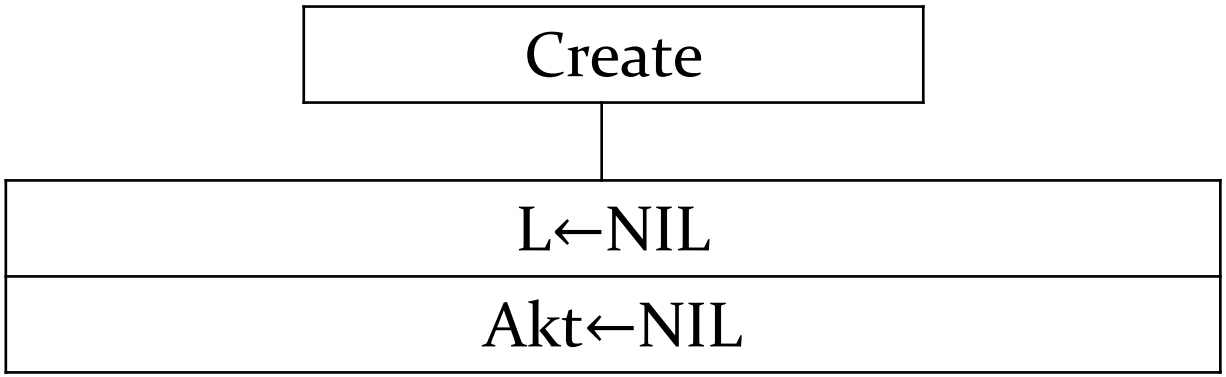


Egyszerű lista – műveletek

- A Lista típus komponensei:
  - L
    - A lista első elemének mutatója,
  - Akt
    - A lista aktuális elemének mutatója

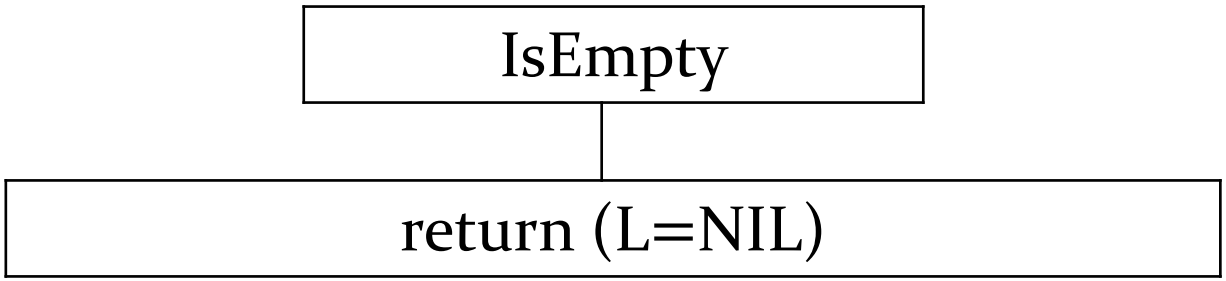
Létrehozás

- Üres listát ad vissza

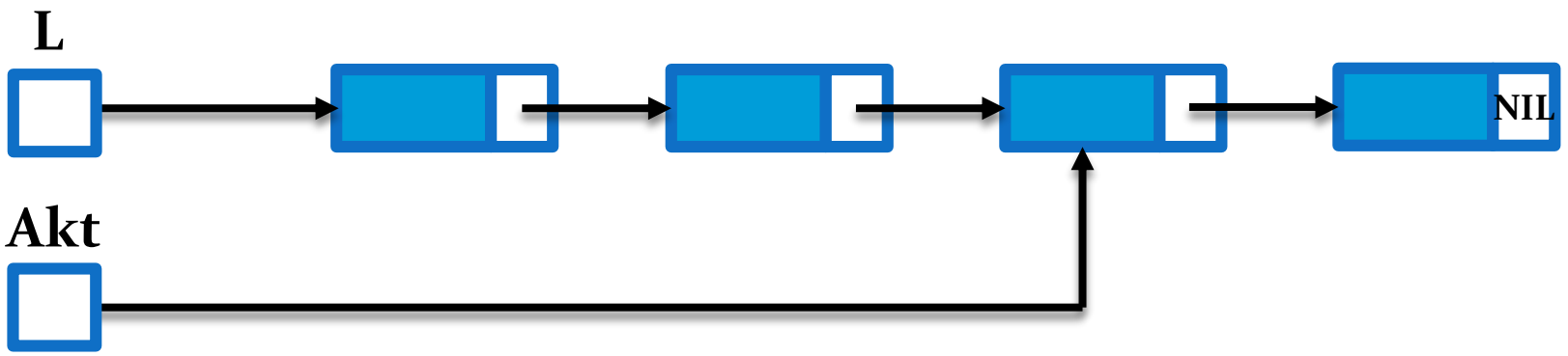


Üres lista lekérdezése

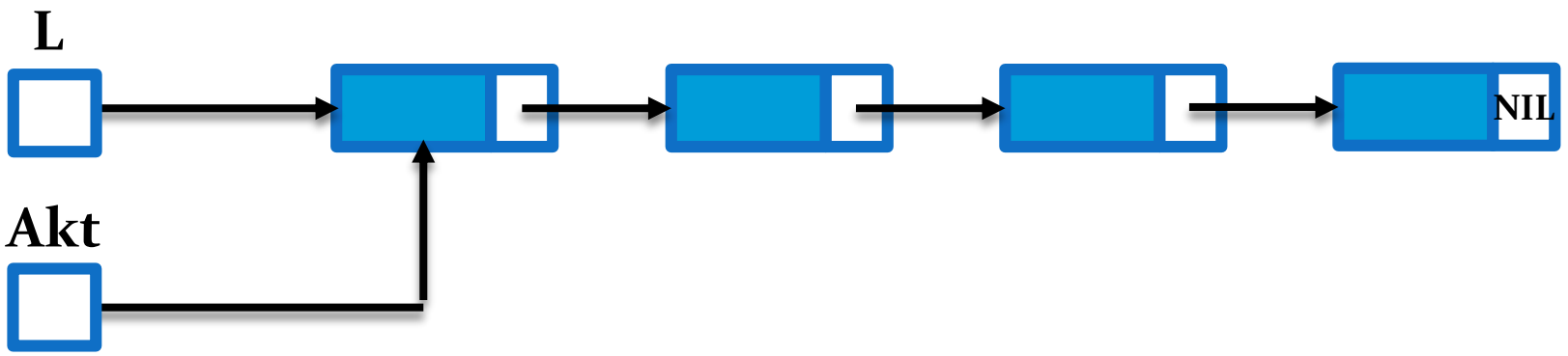
- Logikai értéket ad vissza



Első elemre áll

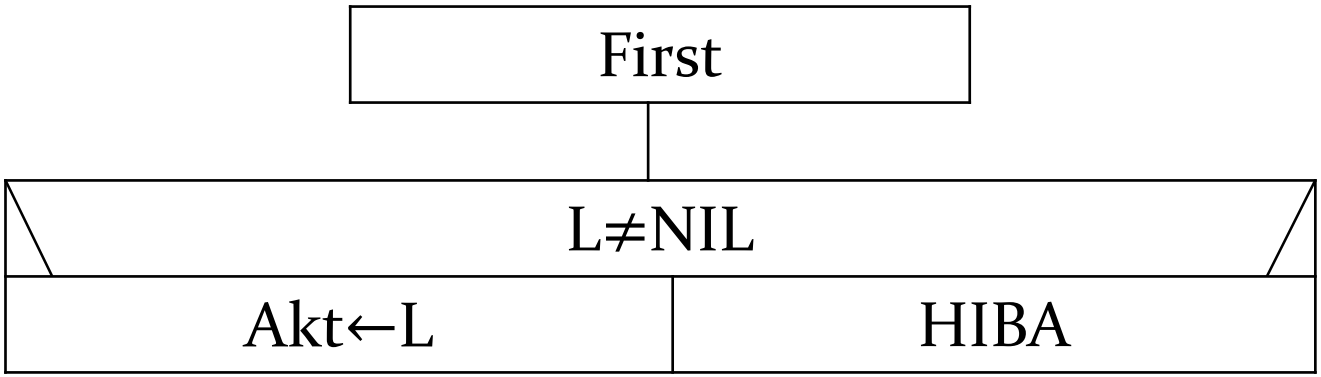


Első elemre áll

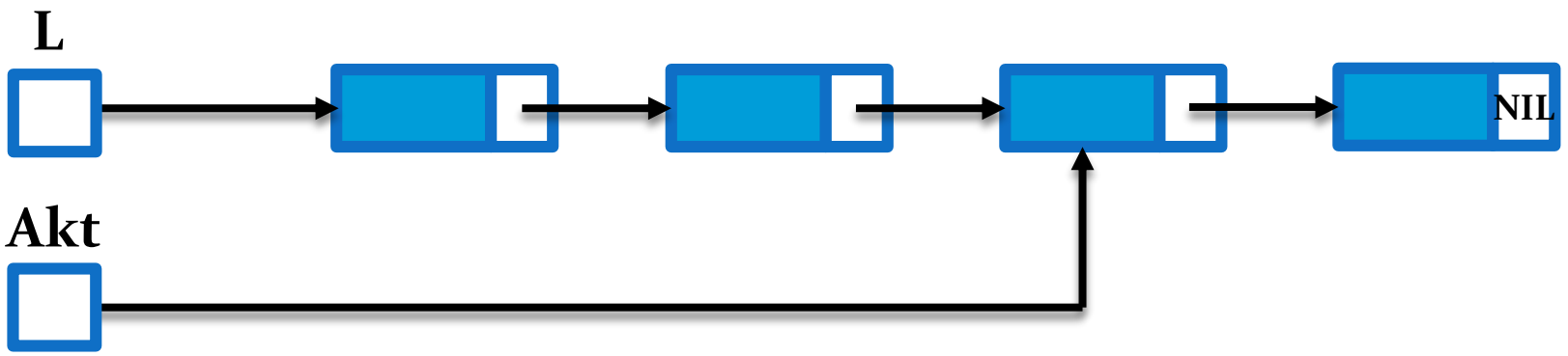


Üres lista lekérdezése

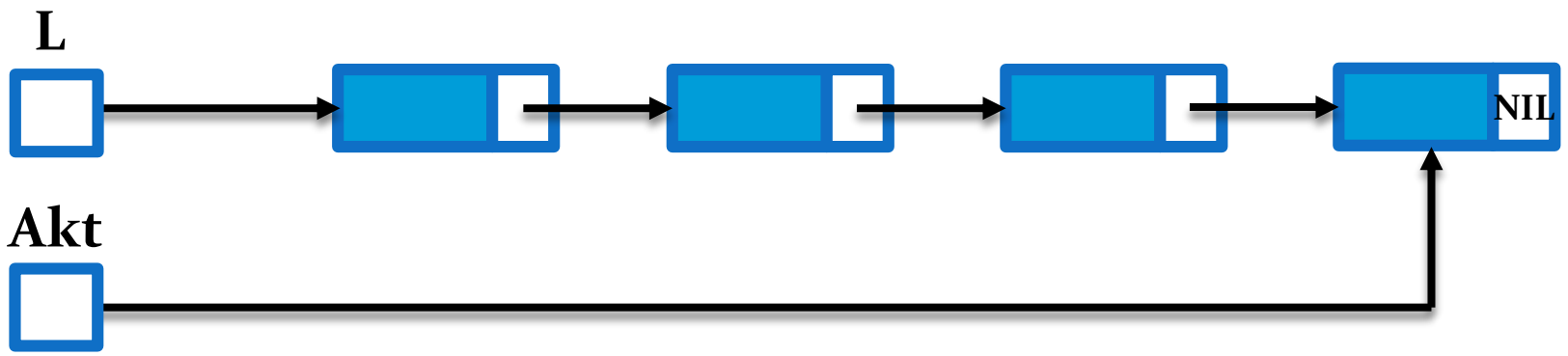
- Üres lista esetén hiba



Következő elemre áll

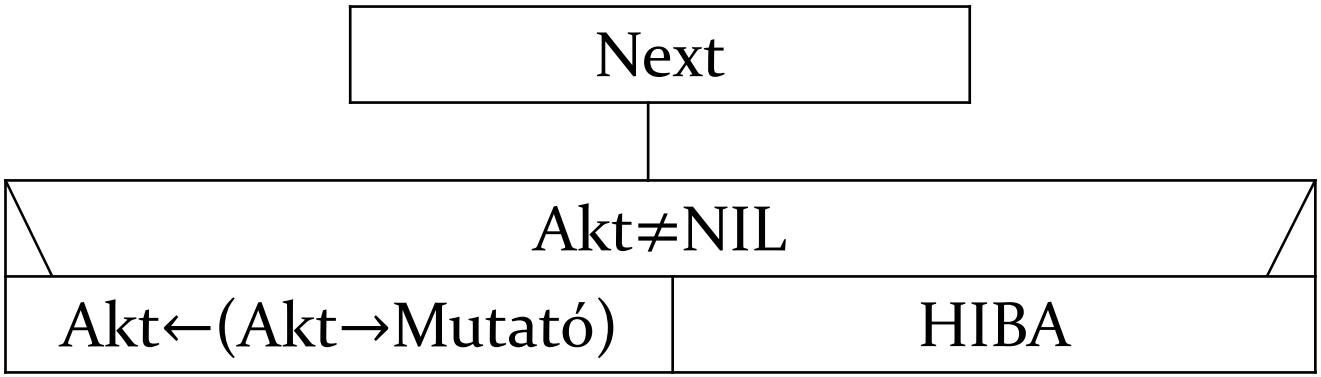


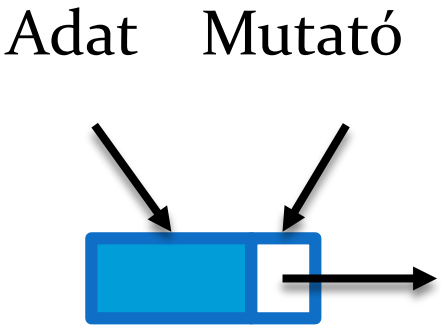
Következő elemre áll



Következő elemre áll

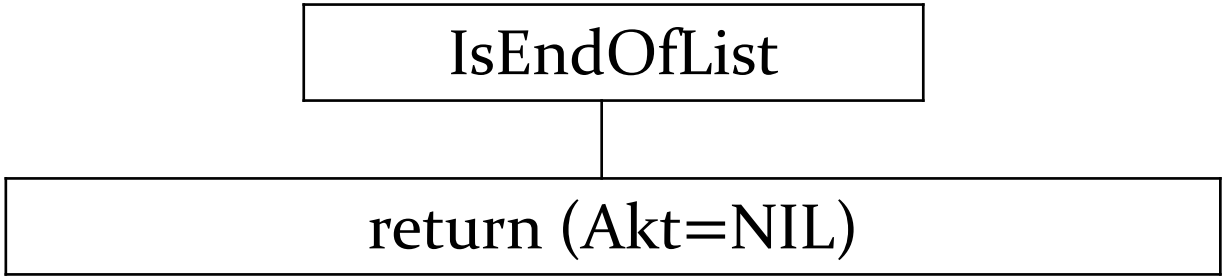
- A lista utolsó elemére kiadott Next hatása Akt=NIL lesz





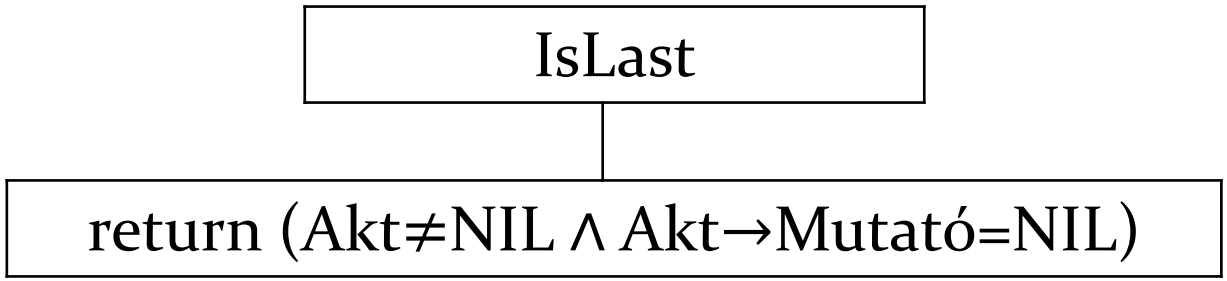
Lista végének lekérdezése

- Logikai értéket ad vissza



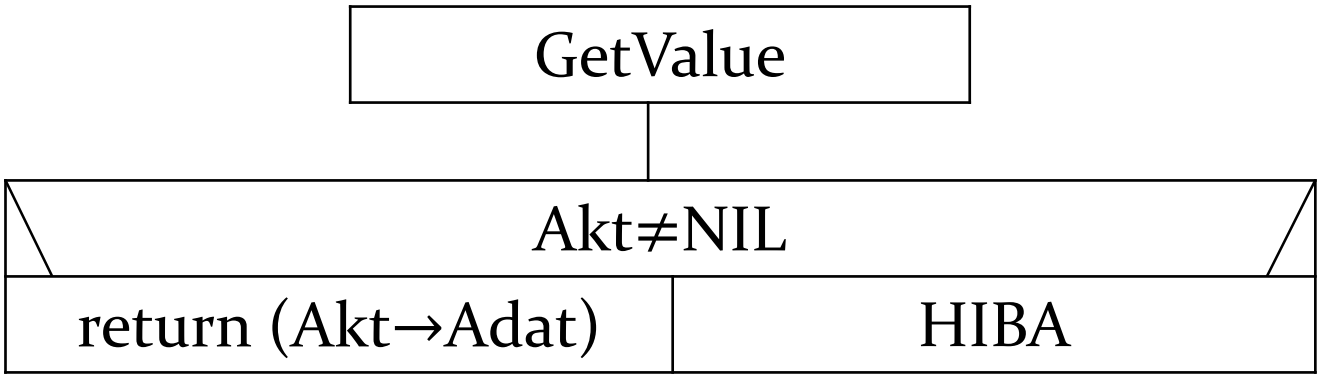
Az aktuális az utolsó elem-e

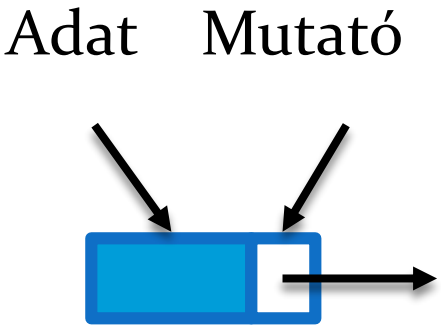
- Logikai értéket ad vissza



Aktuális elem értéke

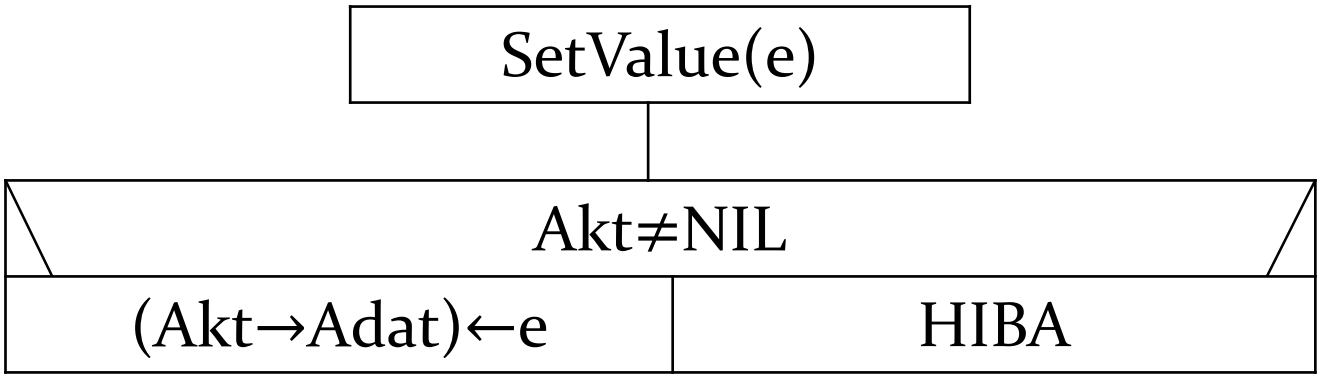
- Az aktuális elem értékével tér vissza





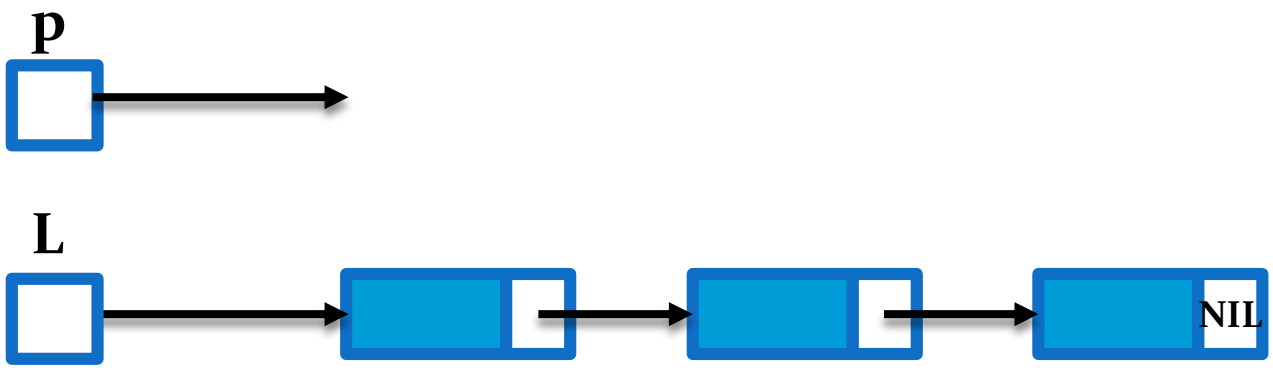
Aktuális elem módosítása

- Az aktuális elem megváltoztatása e -re



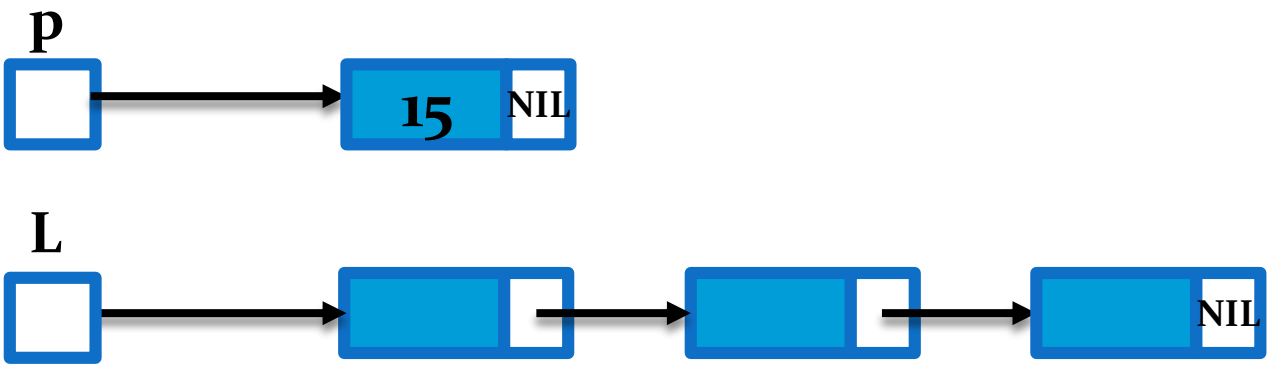
Listaelem beszúrása

- Új listaelem létrehozás és beállítása
  - Deklarálás – p változó, Node típussal



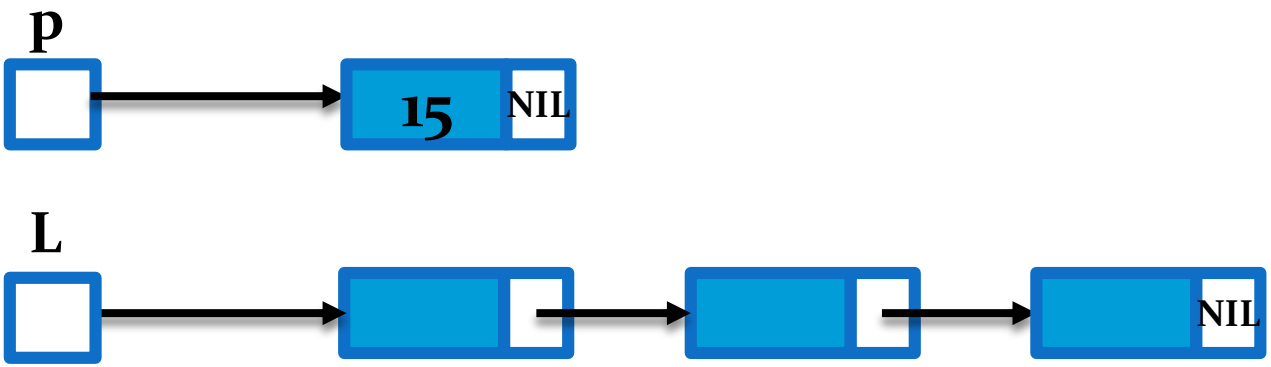
Listaelem beszúrása

- Új listaelem létrehozás és beállítása
  - Deklarálás – p változó, Node típussal
  - Létrehozás – new(p)
  - Értékének megadása



Listaelem beszúrása

- Új listaelem létrehozás és beállítása
  - Deklarálás – p változó, Node típussal
  - Létrehozás – new(p)
  - Értékének megadása

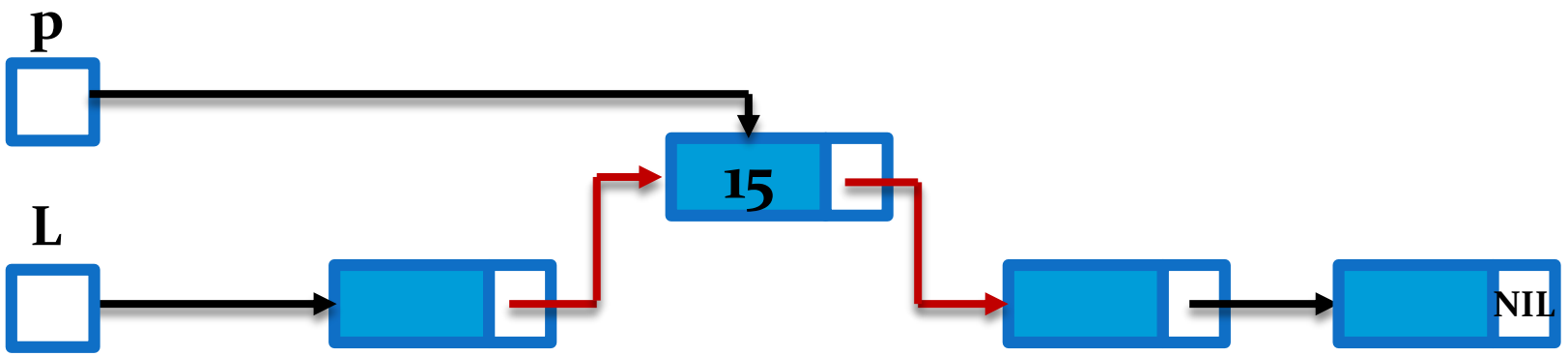


Listaelem beszúrása

- Új listaelem létrehozás és beállítása
  - Deklarálás – p változó, Node típussal
  - Létrehozás – new(p)
  - Értékének megadása
  - Új elem befűzése a láncolatba

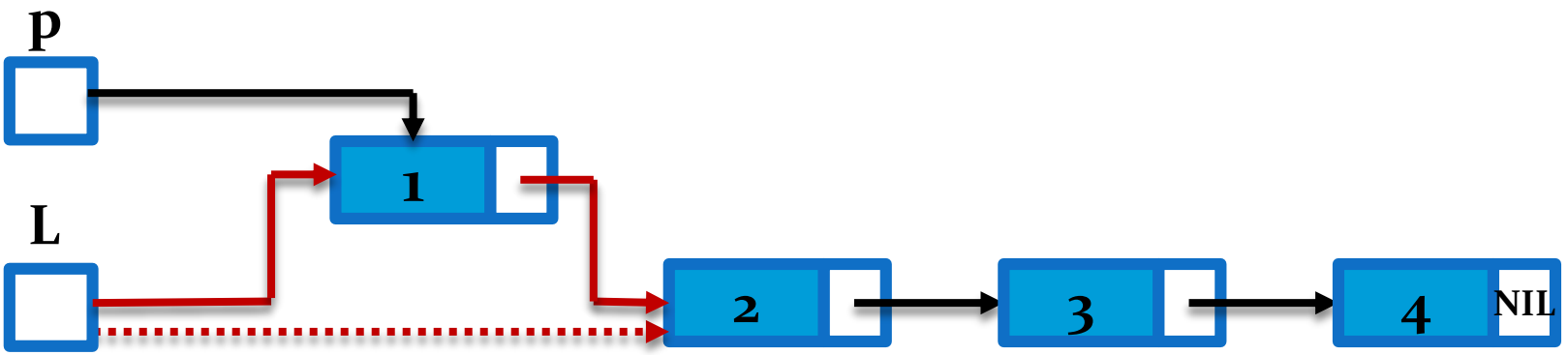
Listaelem beszúrása

- Új listaelem létrehozás és beállítása
  - Deklarálás – p változó, Node típussal
  - Létrehozás – new(p)
  - Értékének megadása
  - Új elem befűzése a láncolatba



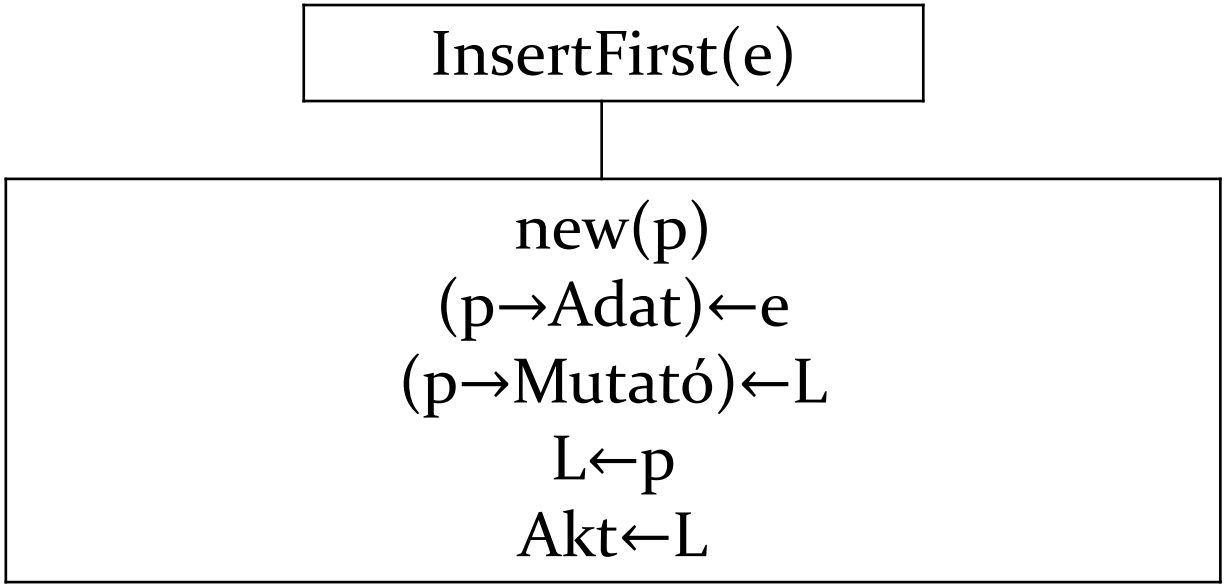
Listaelem beszúrása

- Beszúrás első elemként
  - Üres és nem üres listára is működik
  - Az Akt mutatót (aktuális elem) az újonnan beszúrtra állítja, ami az első



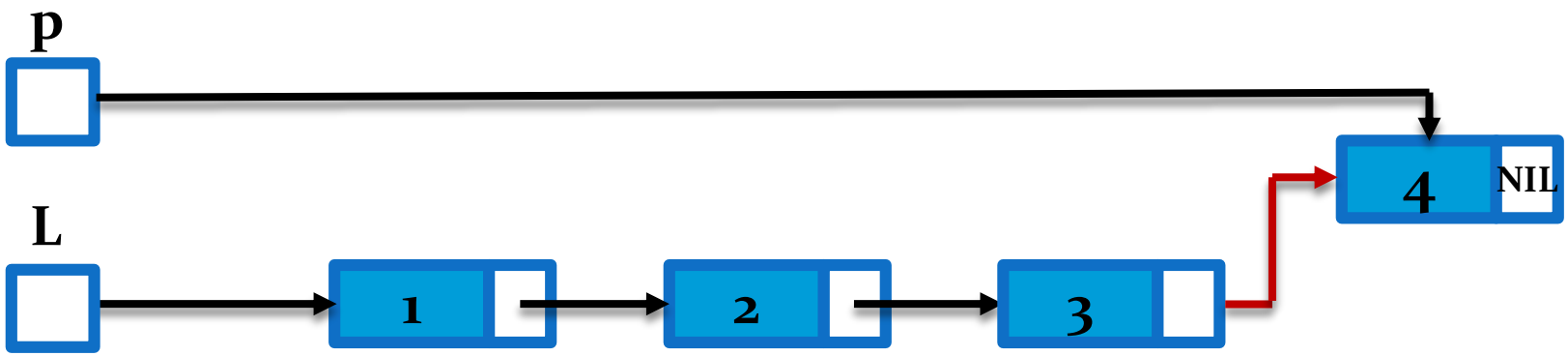
Listaelem beszúrása

- „e” adatelem beszúrása első elemként
  - Üres és nem üres listára is működik
  - Az Akt mutatót (aktuális elem) az újonnan beszúrtra állítja, ami az első



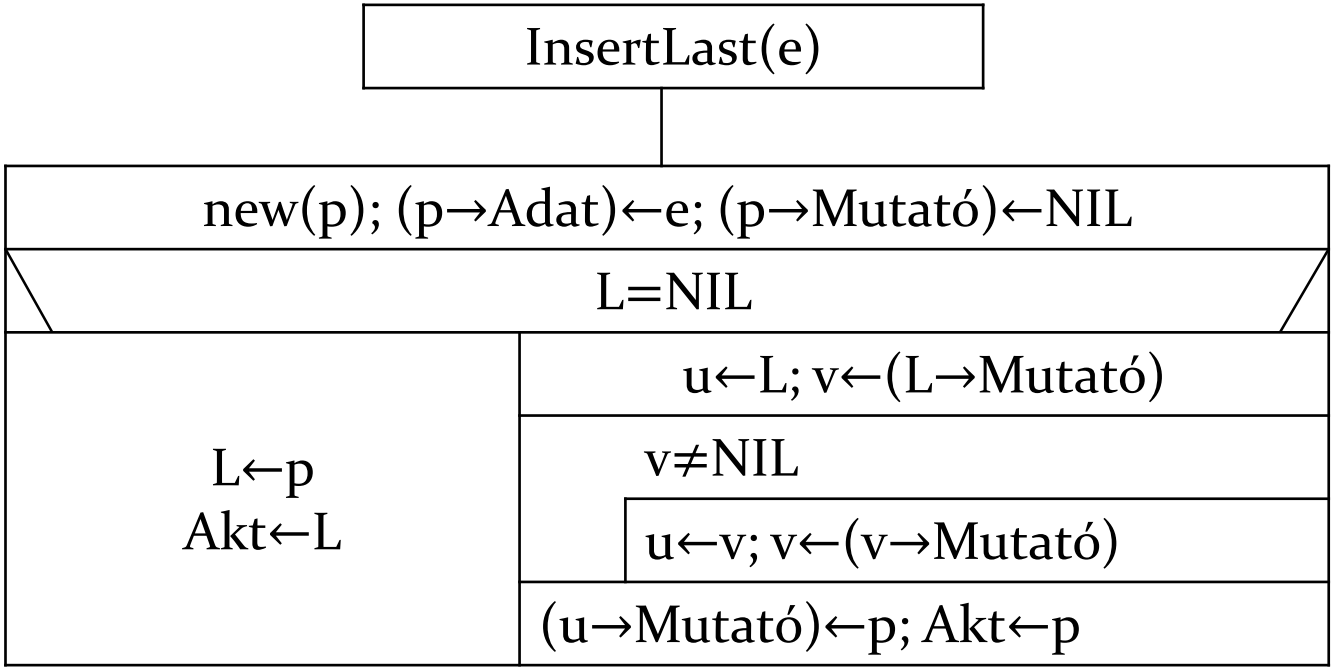
Listaelem beszúrása

- Beszúrás utolsó elemként
  - Üres és nem üres listára is működik
  - Az Akt mutatót (aktuális elem) az újonnan beszúrtra állítja, ami az utolsó



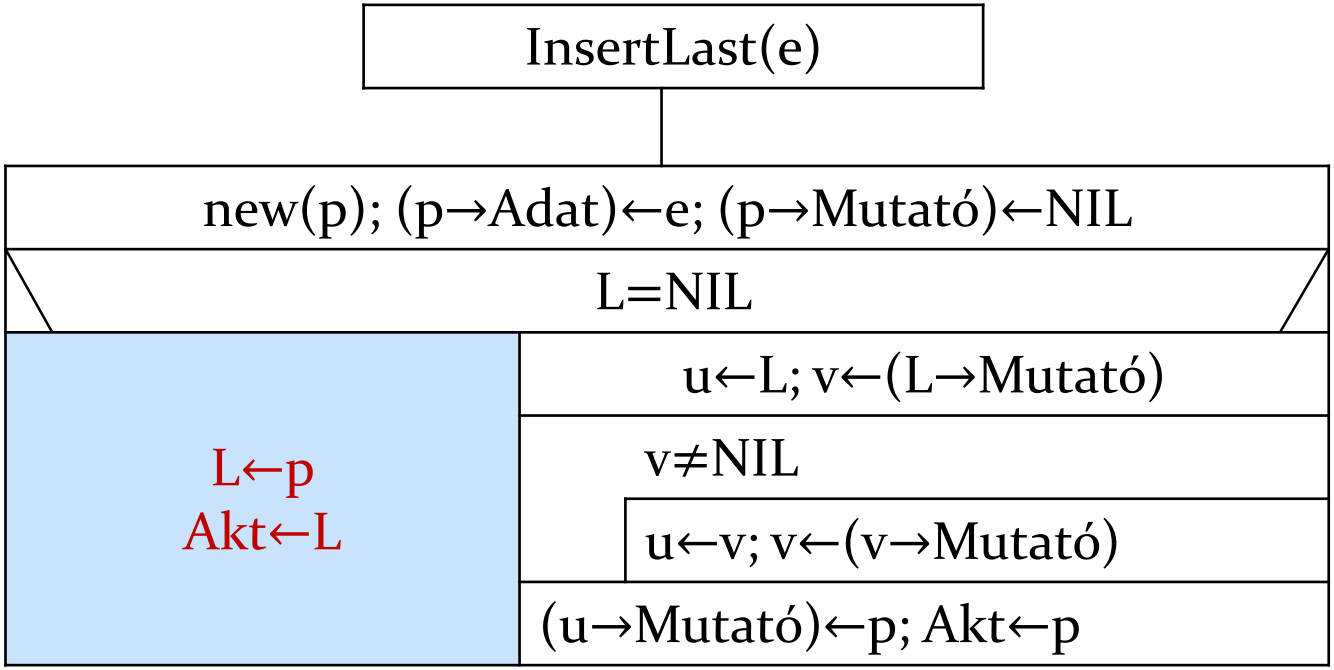
Aktuális elem módosítása

- „e” adatelem beszúrása utolsó elemként
  - Üres és nem üres listára is működik
  - Az Akt mutatót (aktuális elem) az újonnan beszúrtra állítja, ami az utolsó



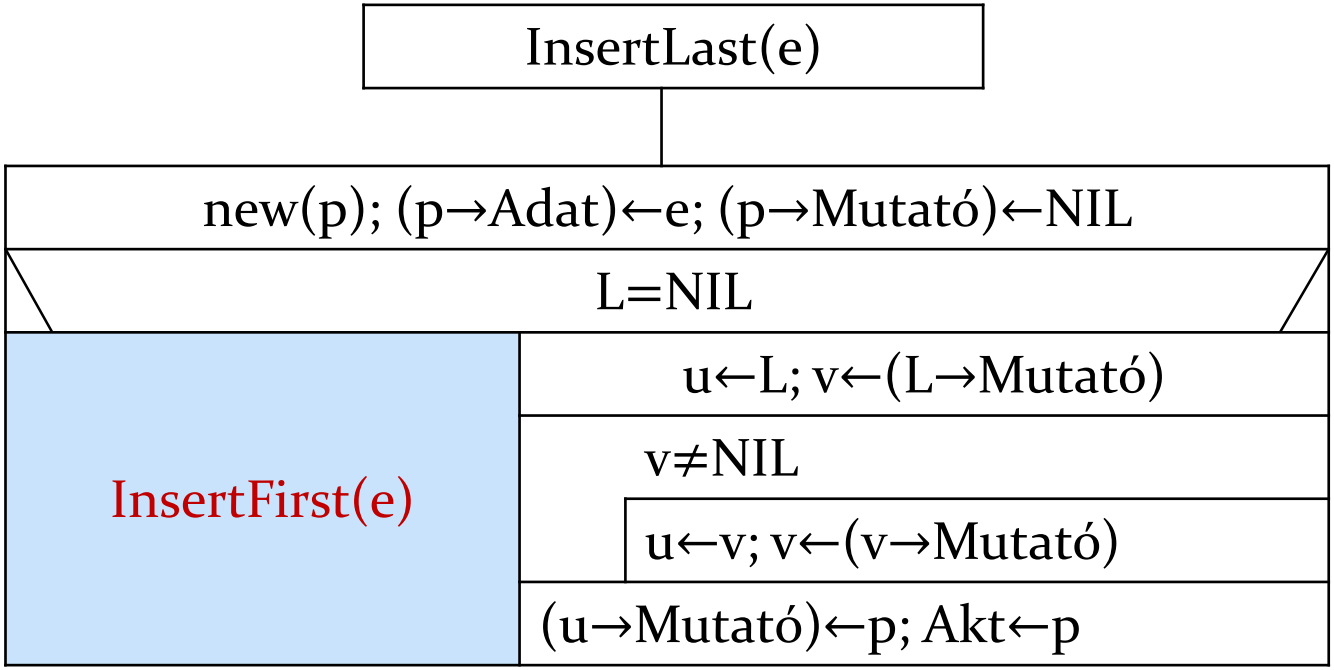
Aktuális elem módosítása

- Ha a lista üres
  - Ami egyezik az InsertFirst(e) algoritmusával



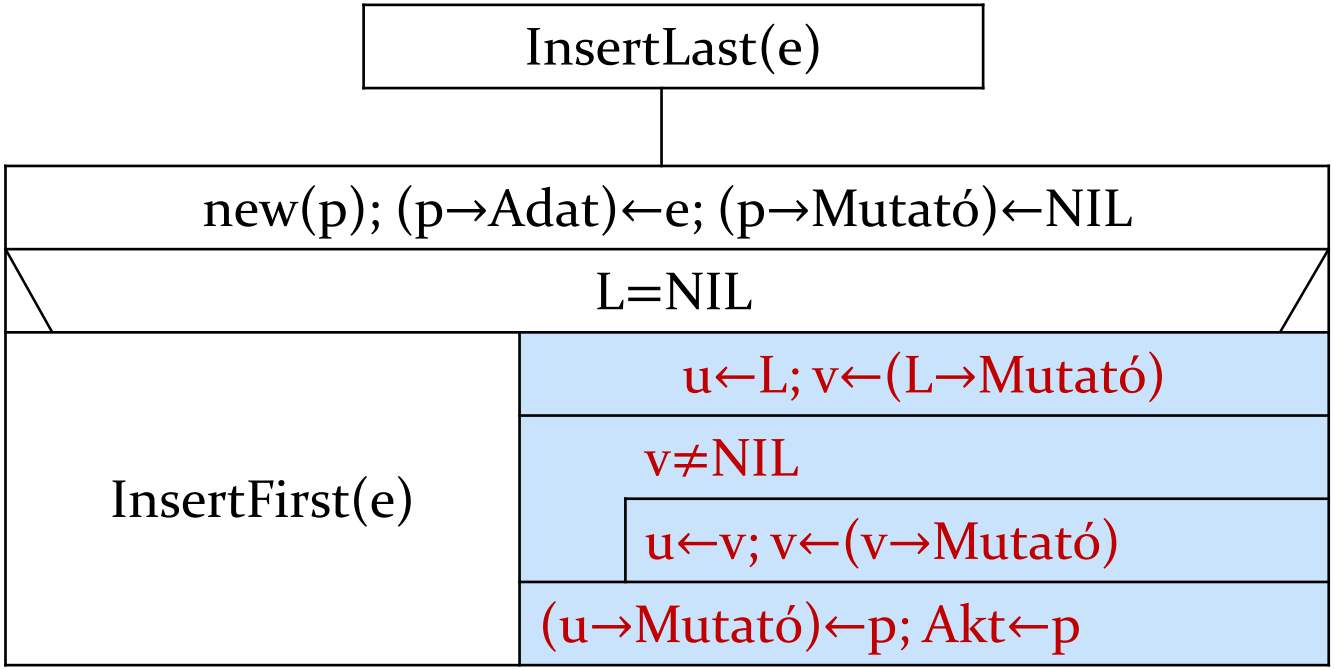
Aktuális elem módosítása

- Ha a lista üres
  - Ami egyezik az InsertFirst(e) algoritmusával



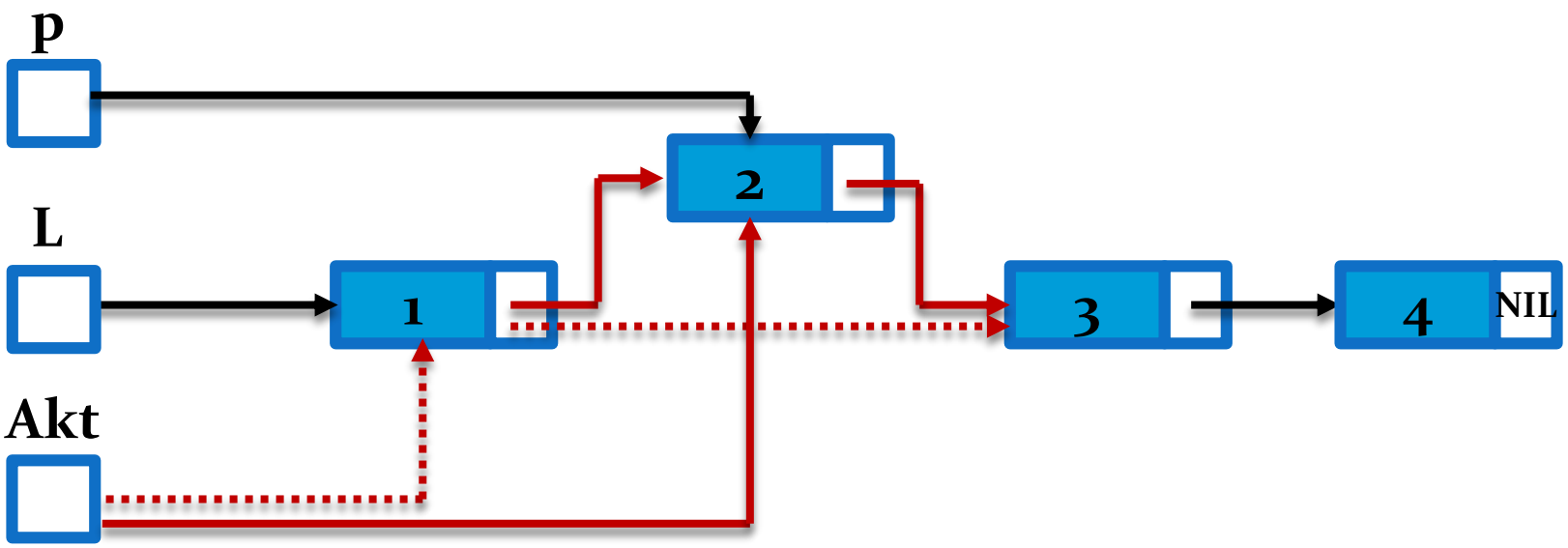
Aktuális elem módosítása

- Ha a lista nem üres
  - Meg kell keresni az eredeti lista utolsó elemét
  - Amögé kell beszúrni az új elemet



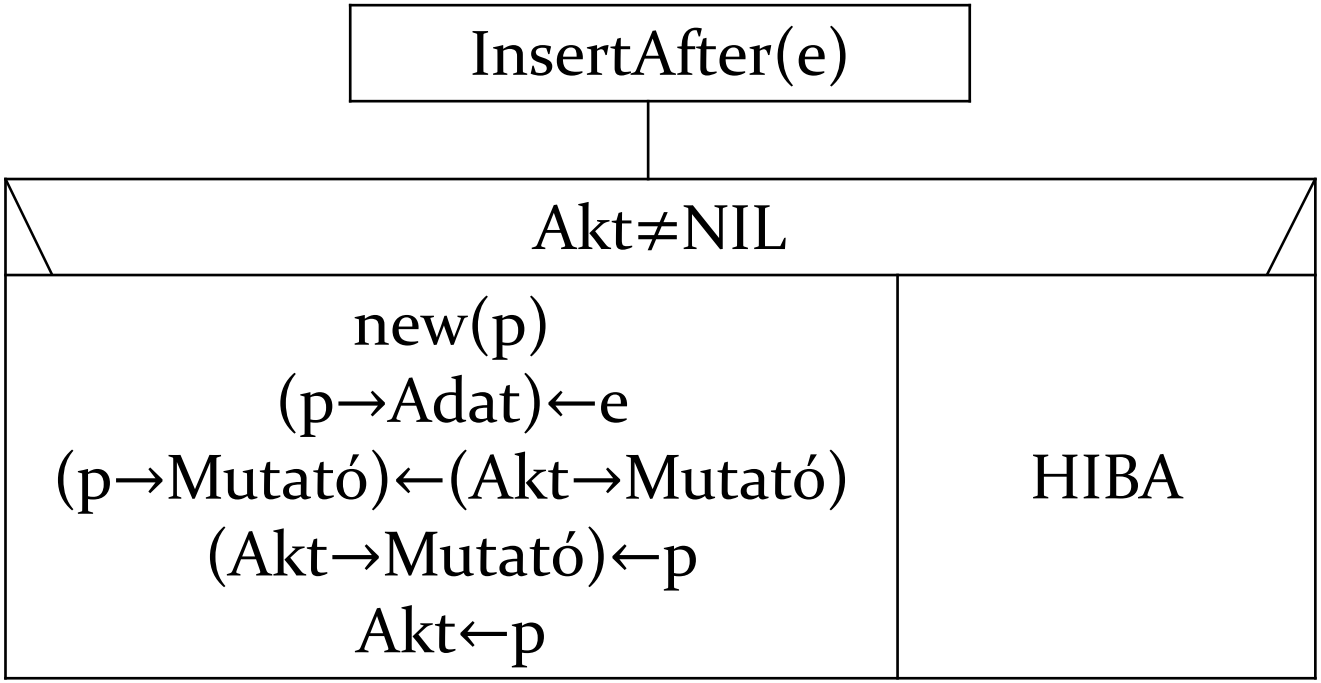
Listaelem beszúrása

- Beszúrás az aktuális elem után



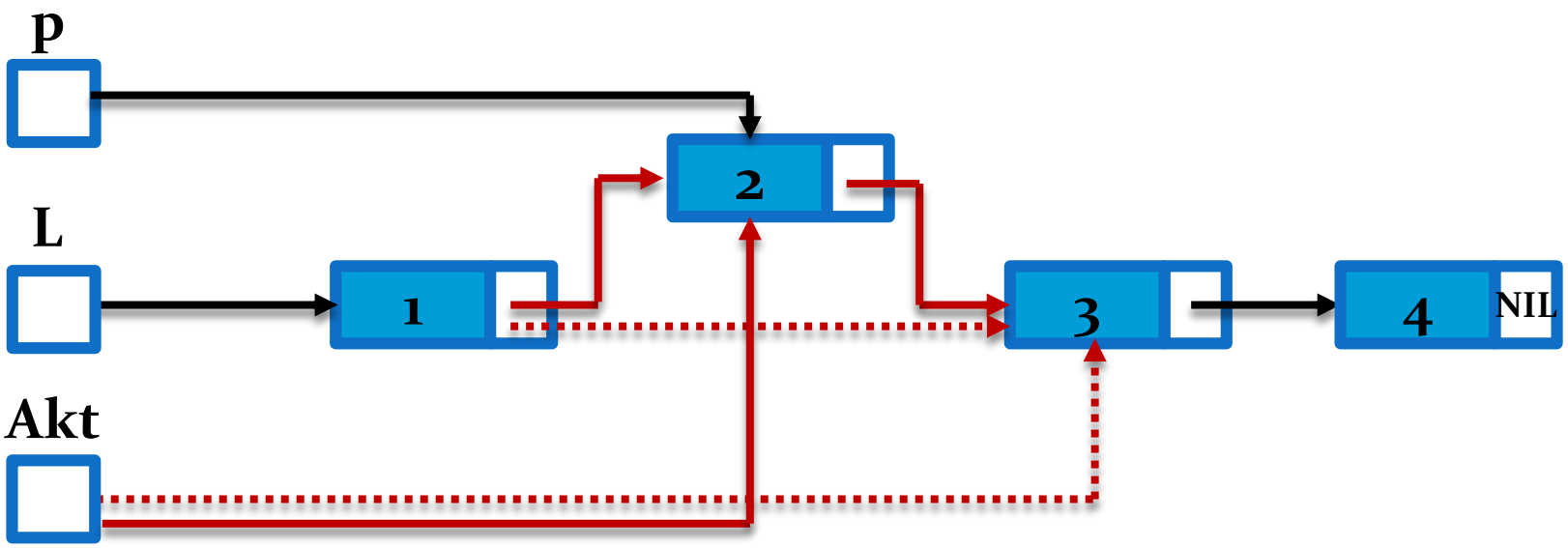
Listaelem beszúrása

- Beszúrás az aktuális elem után
  - Ha nincsen aktuális elem, az hiba
  - Az újonnan beszúrt lesz az aktuális elem



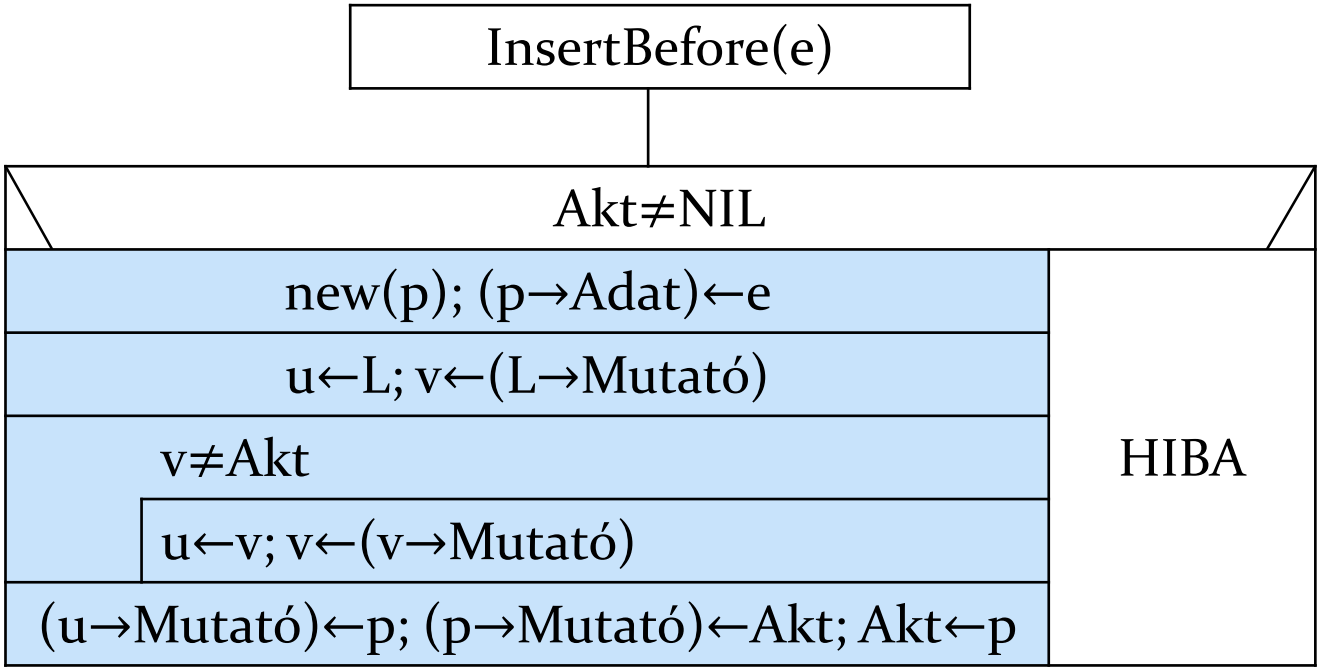
Listaelem beszúrása

- Beszúrás az aktuális elem elé



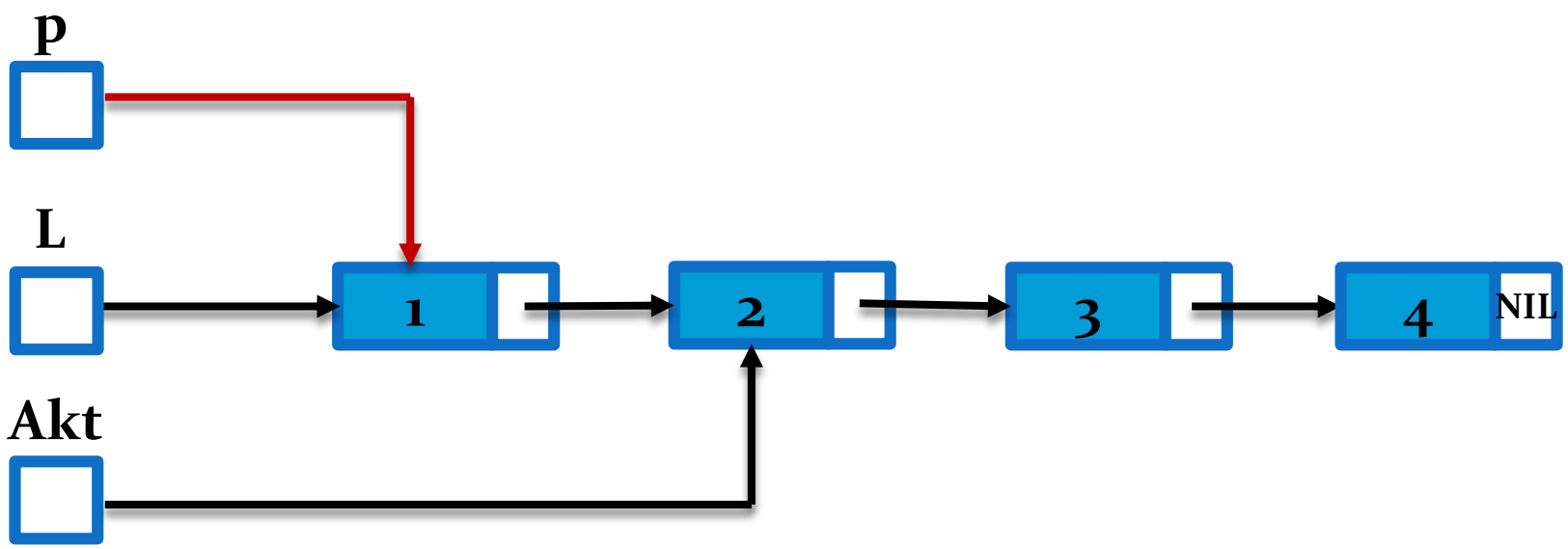
Listaelem beszúrása

- Beszúrás az aktuális elem után
  - Ha nincsen aktuális elem, az hiba
  - Az újonnan beszúrt lesz az aktuális elem



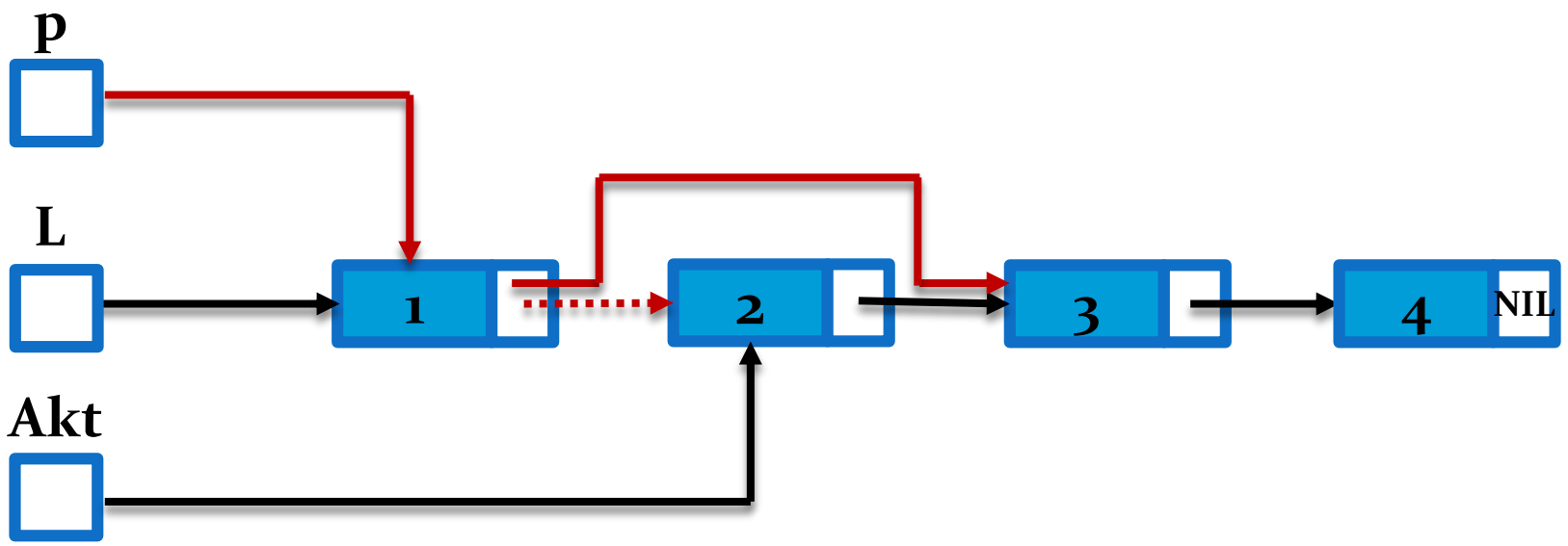
Listaelem törlése

- Aktuális elem törlése
  - Törlendő elem megelőzőjének megkeresése
  - Láncolás megváltoztatása (átláncolás)



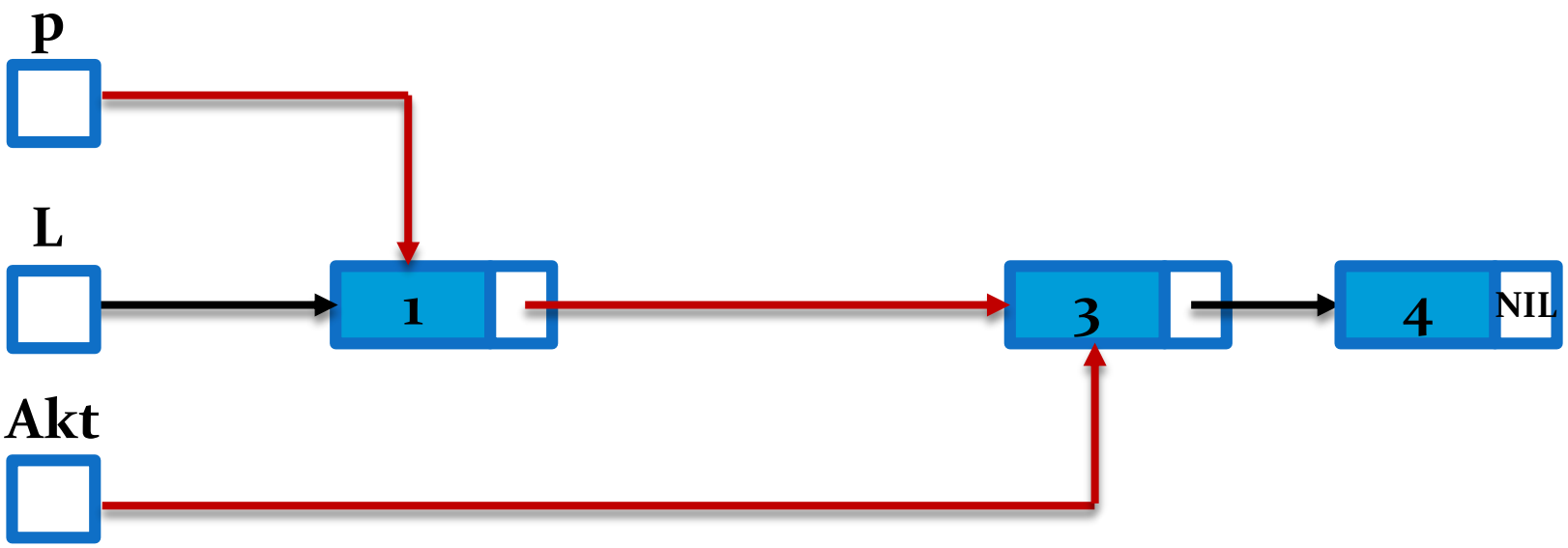
Listaelem törlése

- Aktuális elem törlése
  - Törlendő elem megelőzőjének megkeresése
  - Láncolás megváltoztatása (átláncolás)



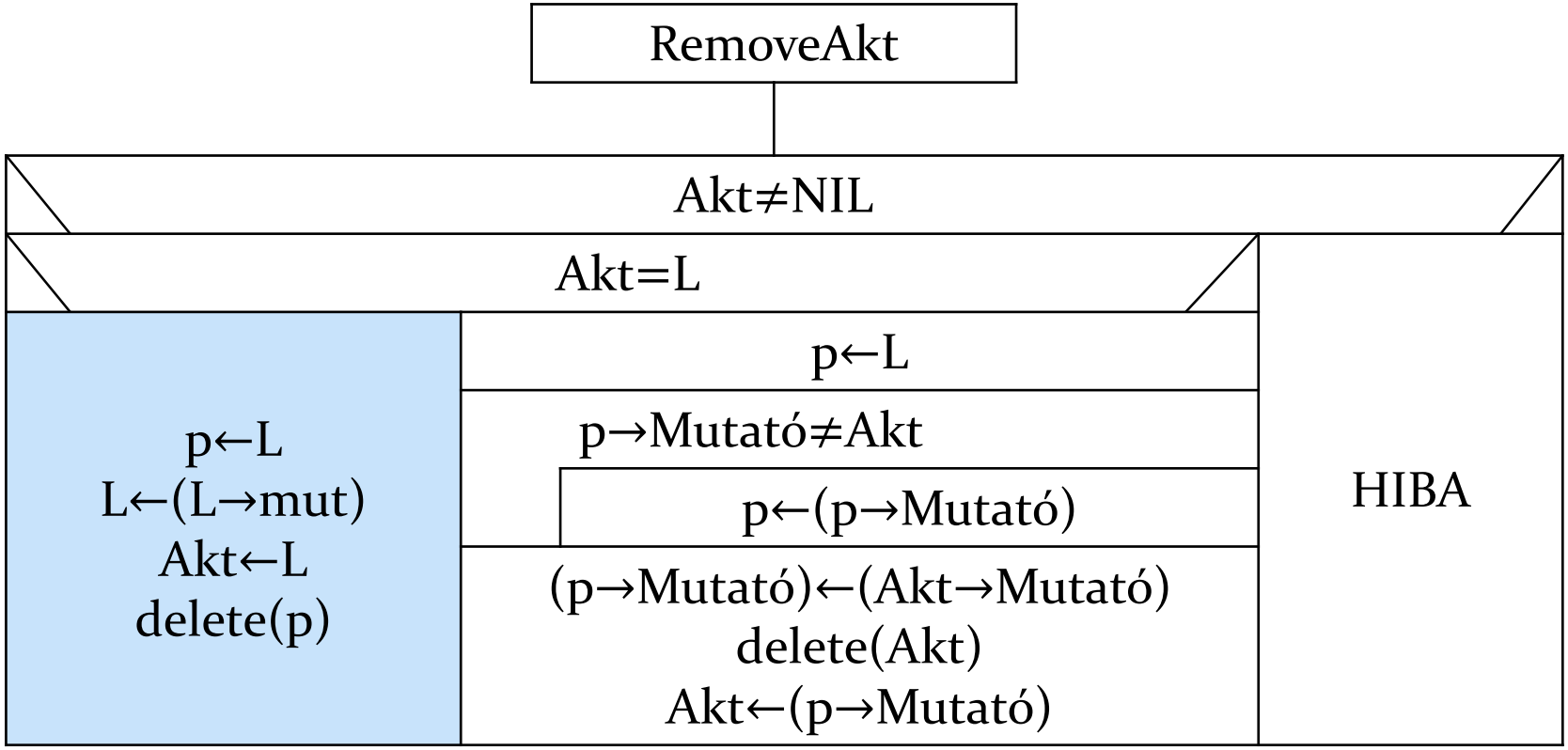
Listaelem törlése

- Aktuális elem törlése
  - Törlendő elem megelőzőjének megkeresése
  - Láncolás megváltoztatása (átláncolás)
  - Memóriából eltávolítás (törlés)
  - Akt beállítása



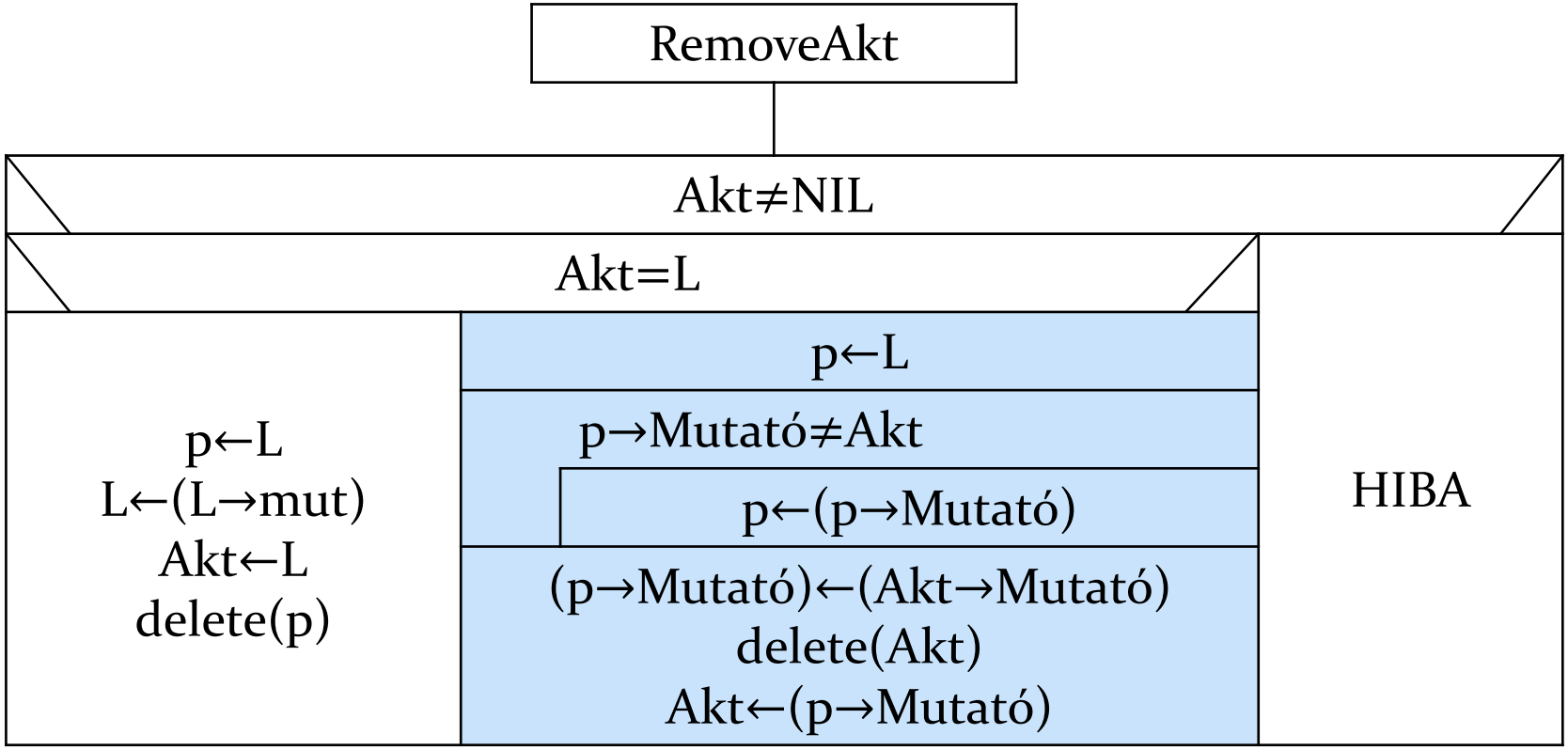
Listaelem törlése

- Aktuális elem törlése
  - Ha az aktuális az első



Listaelem törlése

- Aktuális elem törlése
  - Ha az aktuális nem az első



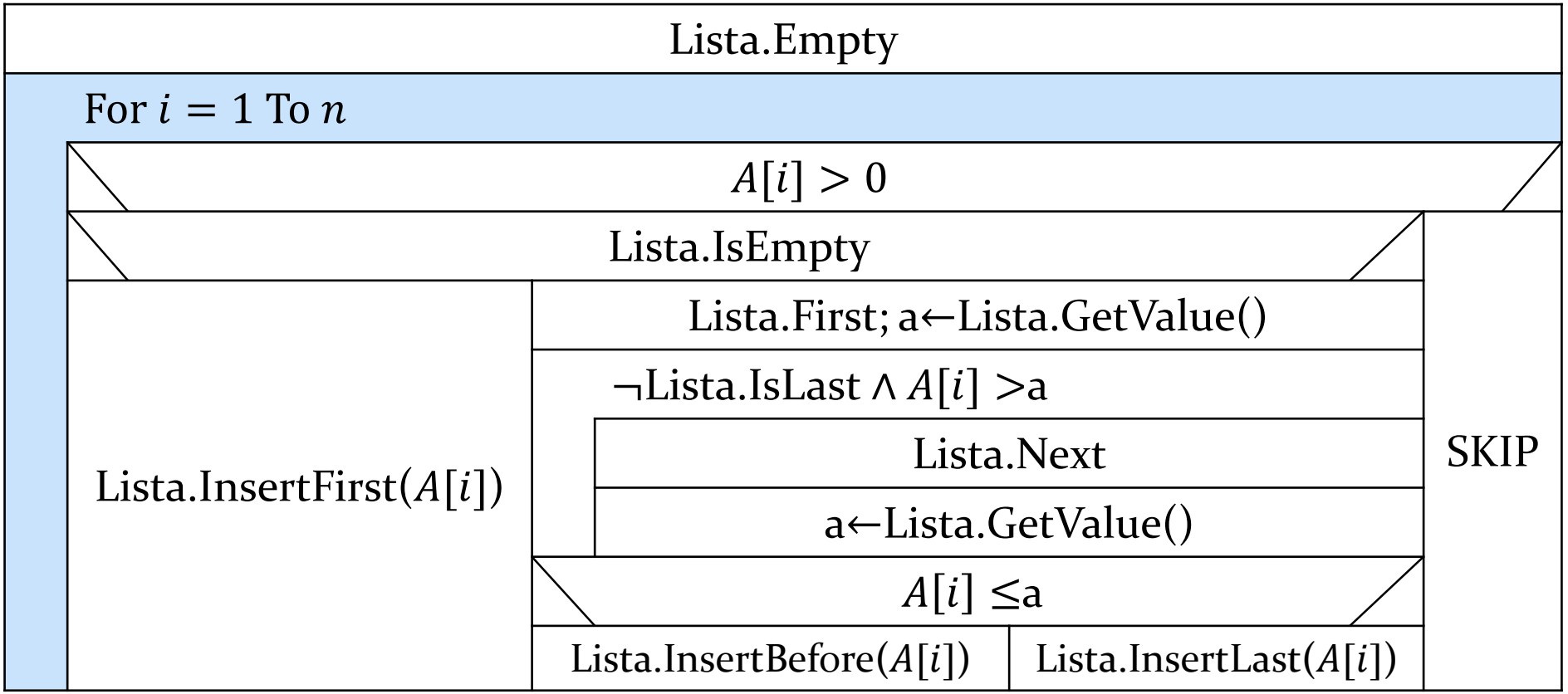
Egyszerű lista – Műveletek

- Megjegyzések – lehetőségek
  - Az Akt nem változik a módosítás során
  - További műveletekre példa
    - Teljes lista törlése
    - Listák összefűzése,
    - Elemszám lekérdezése
  - Ebben az implementációban nem hatékony a megvalósítás
    - Last, Remove, InsertBefore, InsertLast
  - Hatékonnyá tehető
    - Kétirányú láncolással

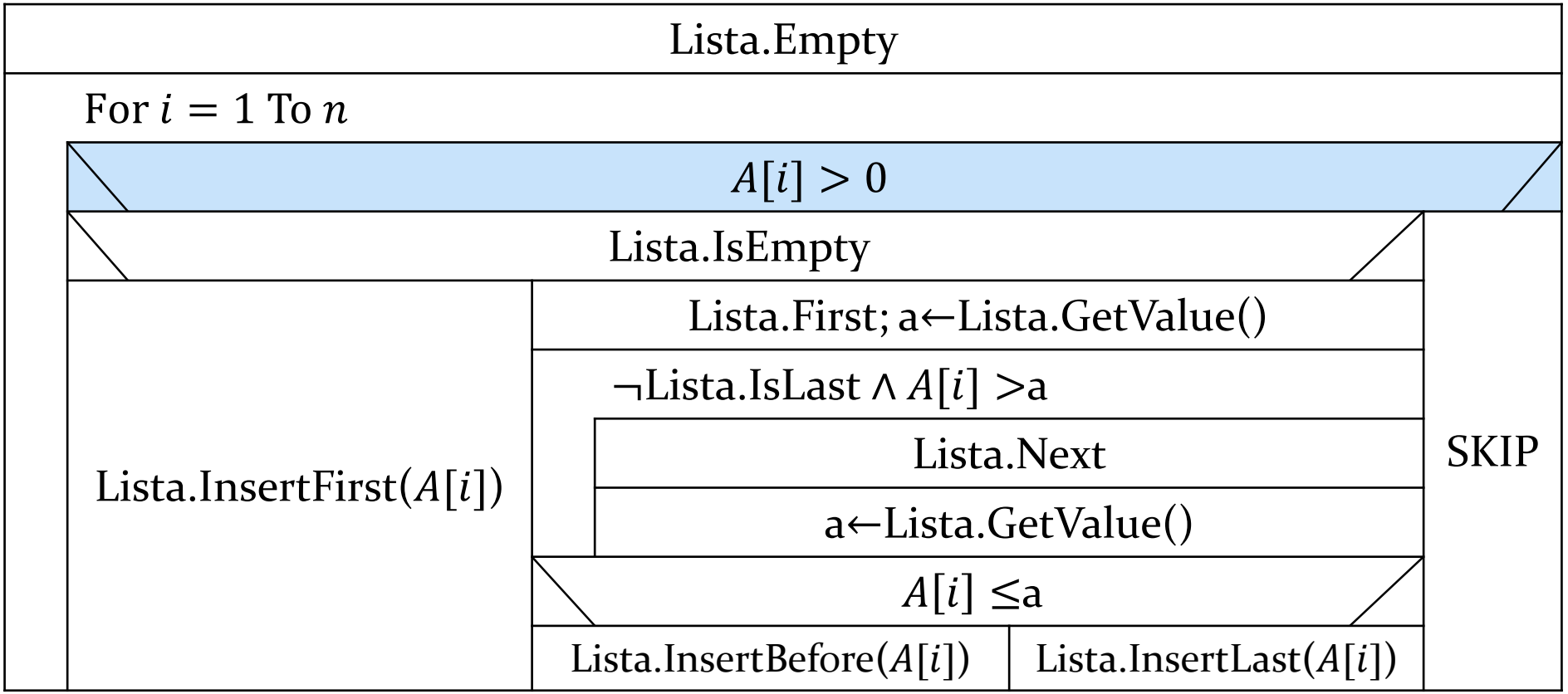
Példa

- Elemek sorbarendezése lista használatával
  - Adott az $A[1..n]$ egészeket tartalmazó tömb
  - Helyezzük el a pozitív elemeit rendezett módon egy listába

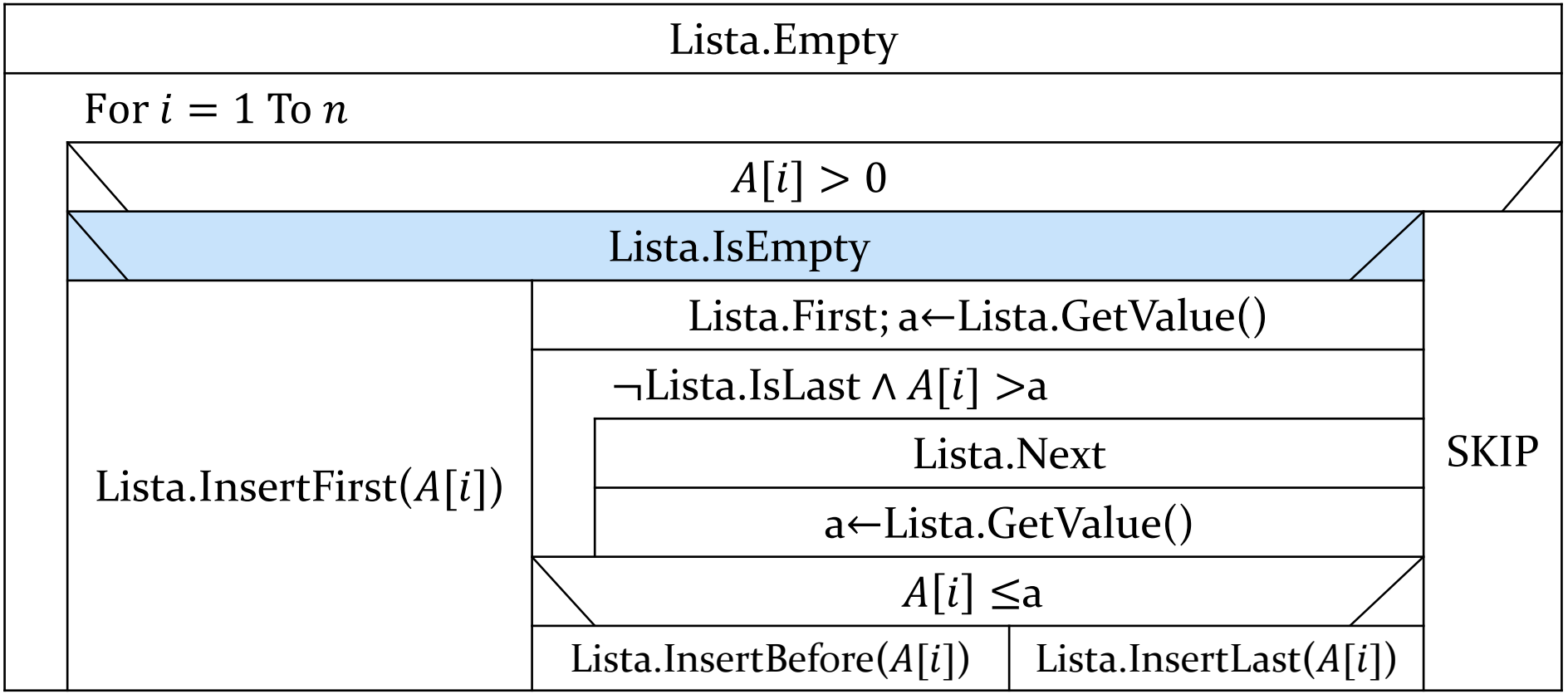
Az algoritmus



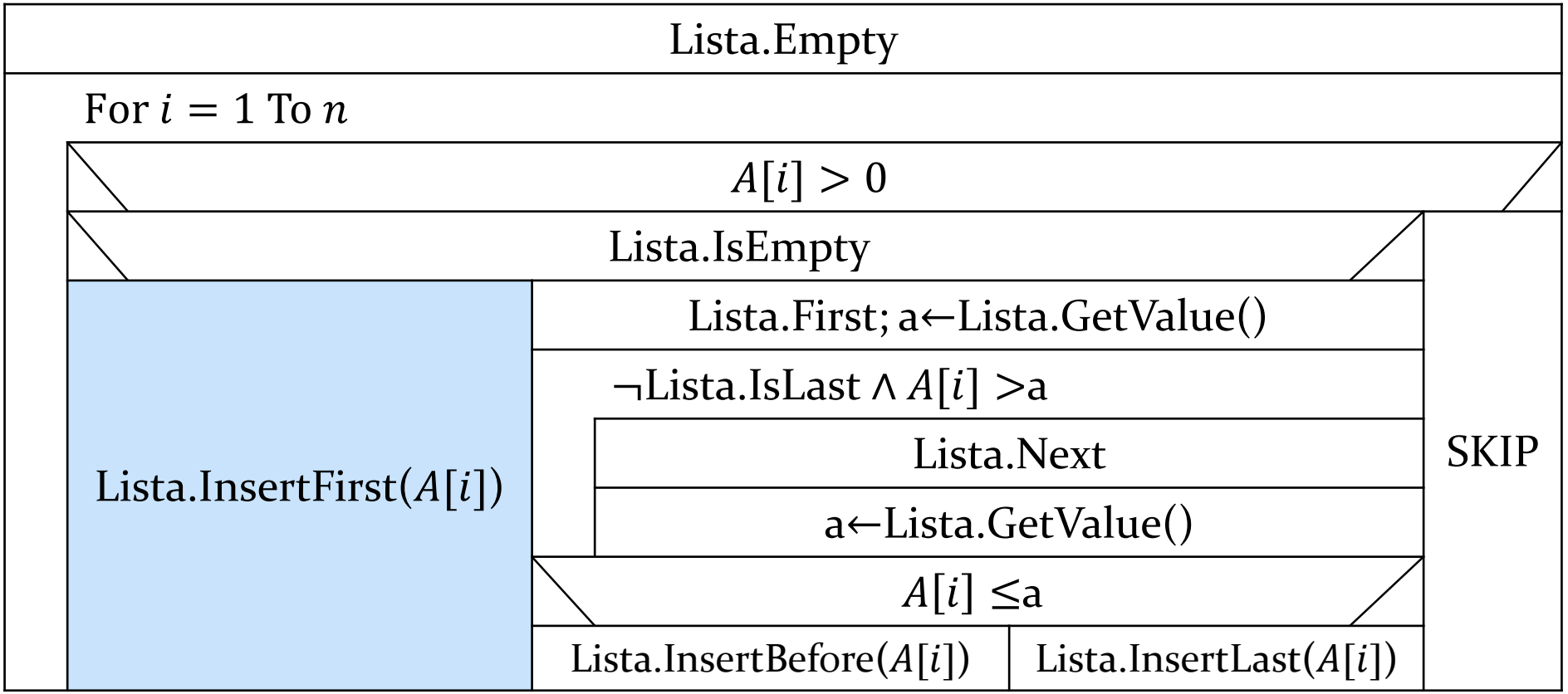
Az algoritmus



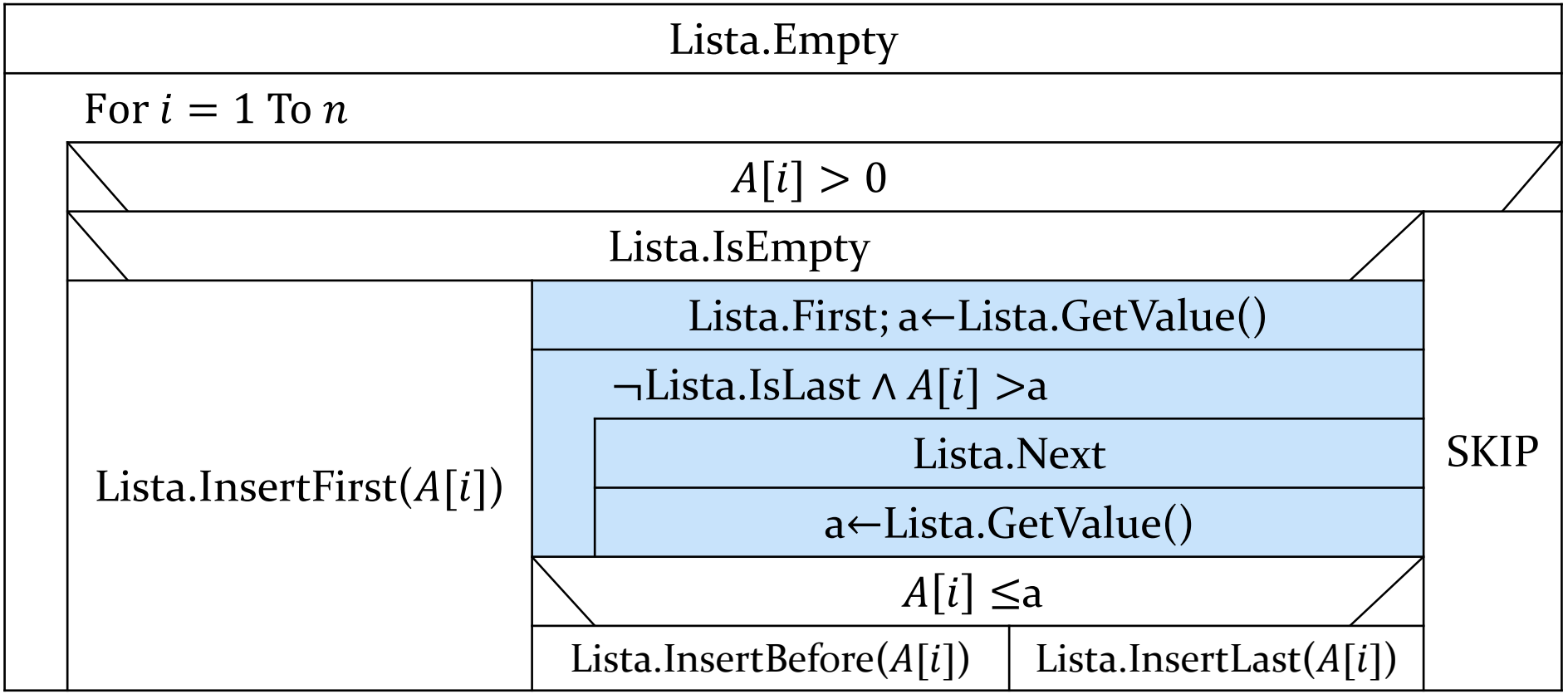
Az algoritmus



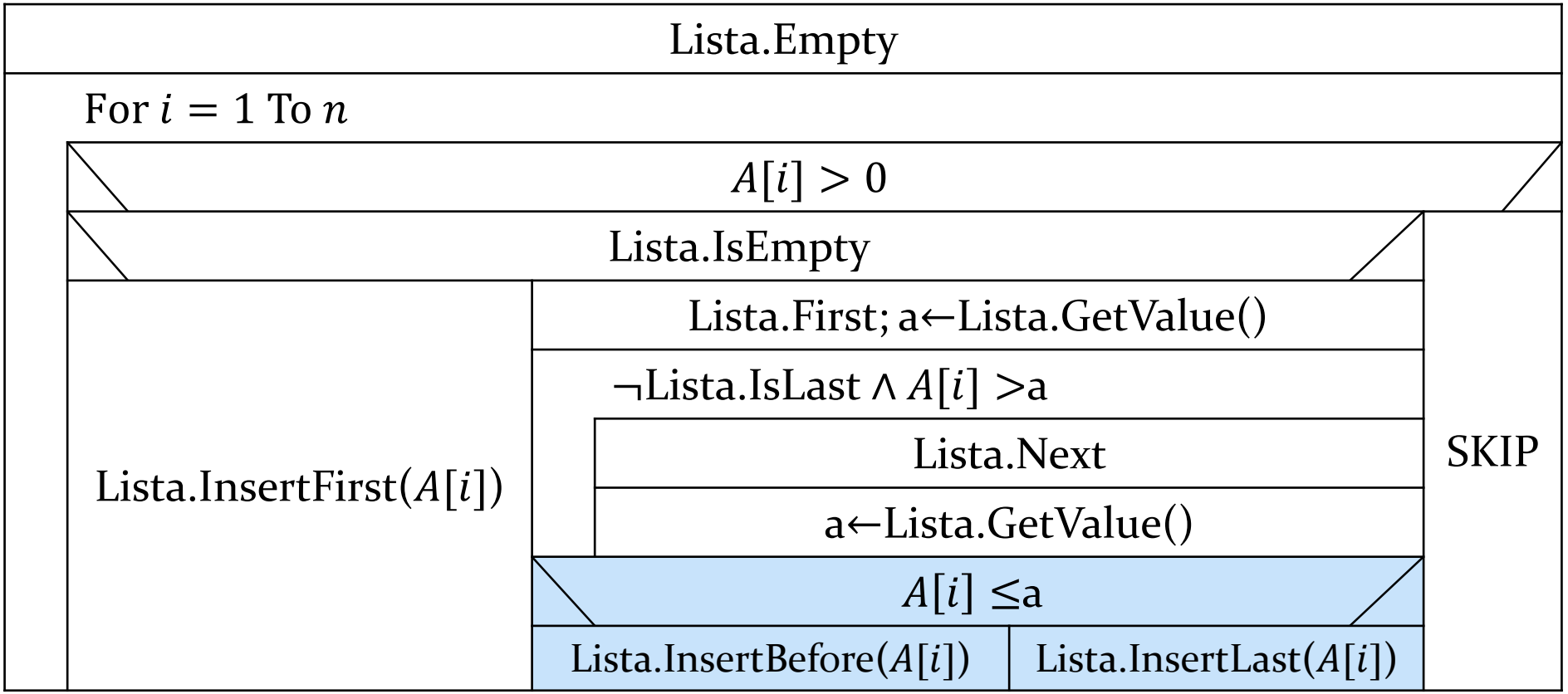
Az algoritmus



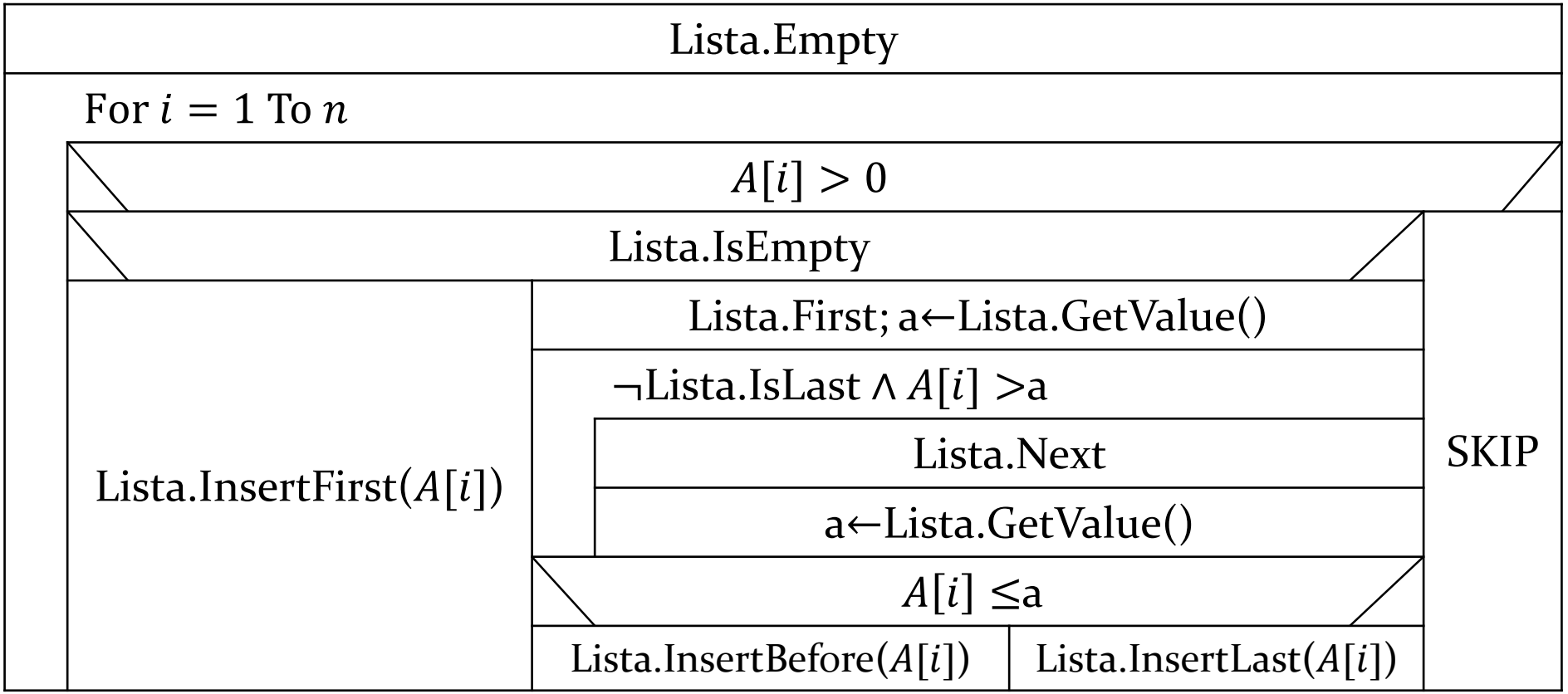
Az algoritmus



Az algoritmus



Az algoritmus



Hiearchikus adatszerkezetek
Fák

Hierarchikus adatszerkezetek

- A hierarchikus adatszerkezet olyan $\langle A, R \rangle$ rendezett pár, amelynél van egy kitüntetett r elem, ez a **gyökérelem**, úgy, hogy:
  1. r nem lehet végpont
     $\forall a \in A$ esetén $\neg R(a, r)$
  2. $\forall a \in \{A \backslash \{r\}\}$ elem egyszer és csak egyszer lehet végpont, azaz $\forall a \in \{A \backslash \{r\}\}$-hez $\exists! \, b \neq a, b \in A$: $R(b, a)$
  3. $\forall a \in \{A \backslash \{r\}\}$ elem r-ből elérhető, azaz
     $\exists a_1, a_2, \ldots, a_n \in A, a_n = a$:
     $R(r, a_1), R(a_1, a_2), \ldots, R(a_{n-1}, a_n))$
- A hierarchikus adatszerkezetek bizonyos értelemben a lista általánosításai

Hierarchikus adatszerkezetek

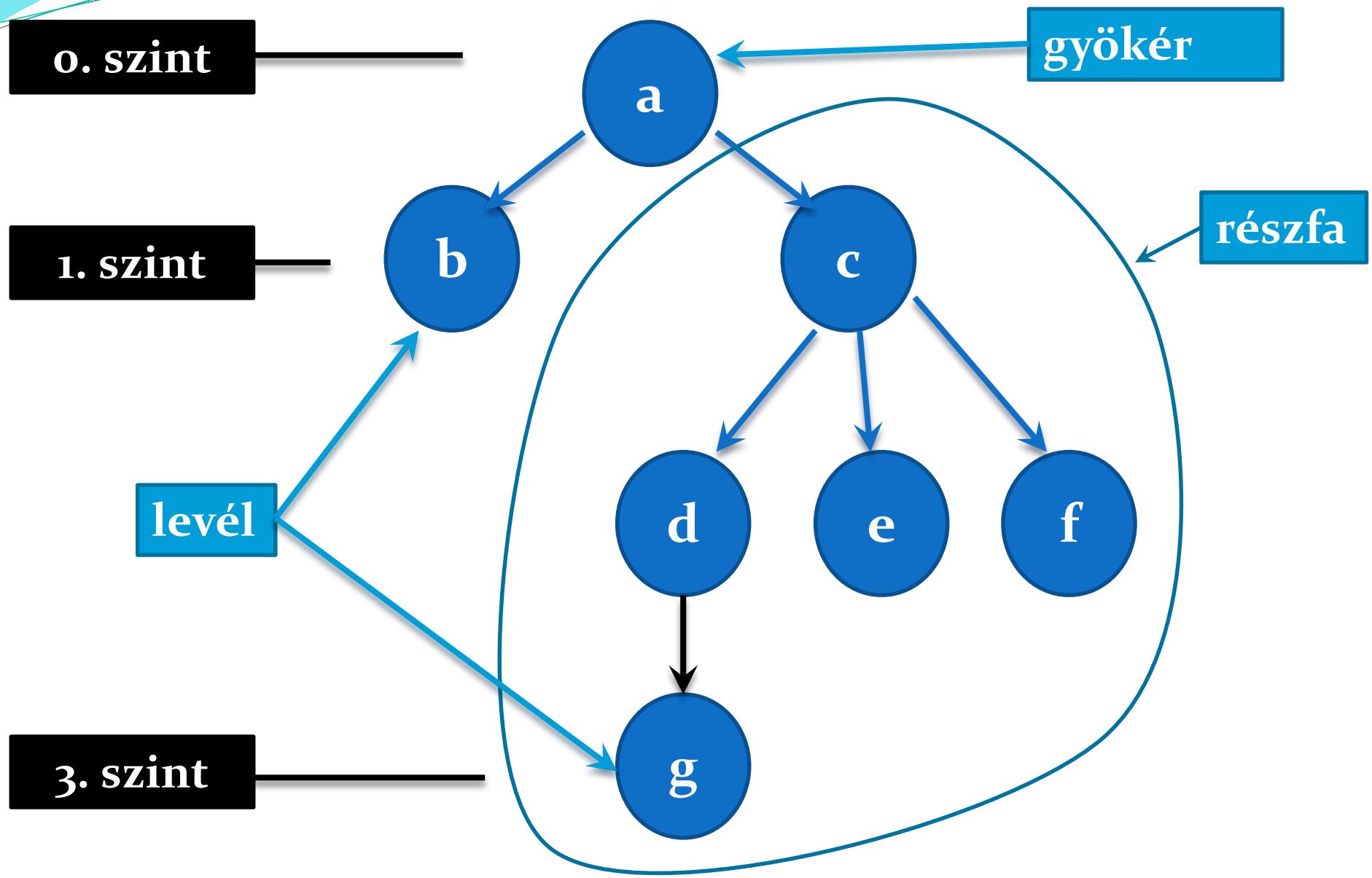
- Egy elemnek akárhány rákövetkezője lehet, de minden elemnek csak egyetlen megelőző eleme van, azaz az adatelemek között egy-sok jellegű kapcsolat áll fenn
- Minden adatelem csak egy helyről érhető el, de egy adott elemből tetszés szerinti számú adatelem látható
  - Például
    - Fa
    - összetett lista
    - B-fa

Fák

- A fa egy hierarchikus adatszerkezet, mely véges számú csomópontból áll, és igazak a következők:
  - Két csomópont között a kapcsolat egyirányú, az egyik a kezdőpont, a másik a végpont
  - Van a fának egy kitüntetett csomópontja, ami nem lehet végpont
    - Ez a fa gyökere
  - Az összes többi csomópont pontosan egyszer végpont

Fák

- A fa rekurzív definíciója:
  - A fa vagy üres, vagy
  - Van egy kitüntetett csomópontja, ez a gyökér.
  - A gyökérhez 0 vagy több diszjunkt fa kapcsolódik
    - Ezek a gyökérhez tartozó részfák
- A fával kapcsolatos algoritmusok gyakran rekurzívak


0. szint
1. szint
3. szint
gyökér
részfa
levél
a
b
c
d
e
f
g

Fák

- Az adatszerkezetben
  - A fa csúcsai az adatelemeknek felelnek meg,
  - Az élek az adatelemek egymás utáni sorrendjét határozzák meg – egy csomópontból az azt követőbe húzott vonal egy él
  - A gyökérelem a fa első eleme, amelynek nincs megelőzője
  - Levélelem a fa azon eleme, amelynek nincs rákövetkezője
  - Közbenső elem az összes többi adatelem

Fák

- Az adatszerkezetben
  - Minden közbenső elem egy részfa gyökereként tekinthető, így a fa részfákra bontható:
    - részfa: „t” részfája "a" -nak, ha
      - "a" a gyökere, azaz közvetlen megelőző eleme „t”-nek, vagy
      - „t” részfája "a" valamely részfájának
  - elágazásszám: közvetlen részfák száma
  - a fa szintje a gyökértől való távolságot mutatja.
    - A gyökérelem a 0. szinten van.
    - A gyökérelem rákövetkezői az 1. szinten. stb.
  - a fa szintjeinek száma a fa magassága

Fák

- További definíciók:
  - Csomópont foka: a csomóponthoz kapcsolt részfák száma
  - Fa foka: a fában található legnagyobb fokszám
  - Levél: 0 fokú csomópont
  - Elágazás (közbenső v. átmenő csomópont): > 0 fokú csomópont
  - Szülő (ős): kapcsolat kezdőpontja
    - csak a levelek nem szülők
  - Gyerek (leszármazott): kapcsolat végpontja
    - csak a gyökér nem gyerek
    - ugyanazon csomópont leszármazottai egymásnak testvérei

Fák

- További definíciók
  - Szintszám: gyökértől mért távolság.
    - A gyökér szintszáma 0.
    - Ha egy csomópont szintszáma n, akkor a hozzá kapcsolódó csomópontok szintszáma $n + 1$.
  - Útvonal: az egymást követő élek sorozata
    - Minden levélelem a gyökértől pontosan egy úton érhető el.
  - Ág: az az útvonal, amely levélben végződik
  - Üresfa az a fa, amelyiknek egyetlen eleme sincs. (Ω)
  - Fa magassága: a levelekhez vezető utak közül a leghosszabb
    - Mindig eggyel nagyobb, mint a legnagyobb szintszám

Fák

- További definíciók:
  - Minimális magasságú az a fa, amelynek a magassága az adott elemszám esetén a lehető legkisebb.
    - Valójában ilyenkor minden szintre a maximális elemszámú elemet építjük be.
  - Egy fát kiegyensúlyozottnak nevezünk, ha csomópontjai azonos fokúak, és minden szintjén az egyes részfák magassága nem ingadozik többet egy szintnél.
  - Rendezett fa: ha az egy szülőhöz tartozó részfák sorrendje lényeges, azok rendezettek.

Feladat

- Maximum hány csomópont helyezhető el egy f fokú, m magasságú fában?

$$1 + f + f^2 + f^3 + \cdots$$

$$\frac{(f^{m+1} - 1)}{(f - 1)}$$

Fák műveletei

- Lekérdező
  - Üres_e – logikai értéket ad vissza
  - Gyökérelem – visszaadja a gyökér adatelemet
  - Keres(e) – adott e adatelemet keres, egy ilyen elem mutatóját adja vissza

Fák műveletei

- Módosító
  - Üres
    - létrehoz egy üres fát
  - Beszúr(e)
    - adott e adatelemet beszúr
  - MódosítGyökér(e)
    - adott e adatelem lesz a gyökér
  - Töröl(e)
    - törli az e adatelemet
      - egy előfordulást
      - összes előfordulást
  - TörölFa
    - törli az összes elemet

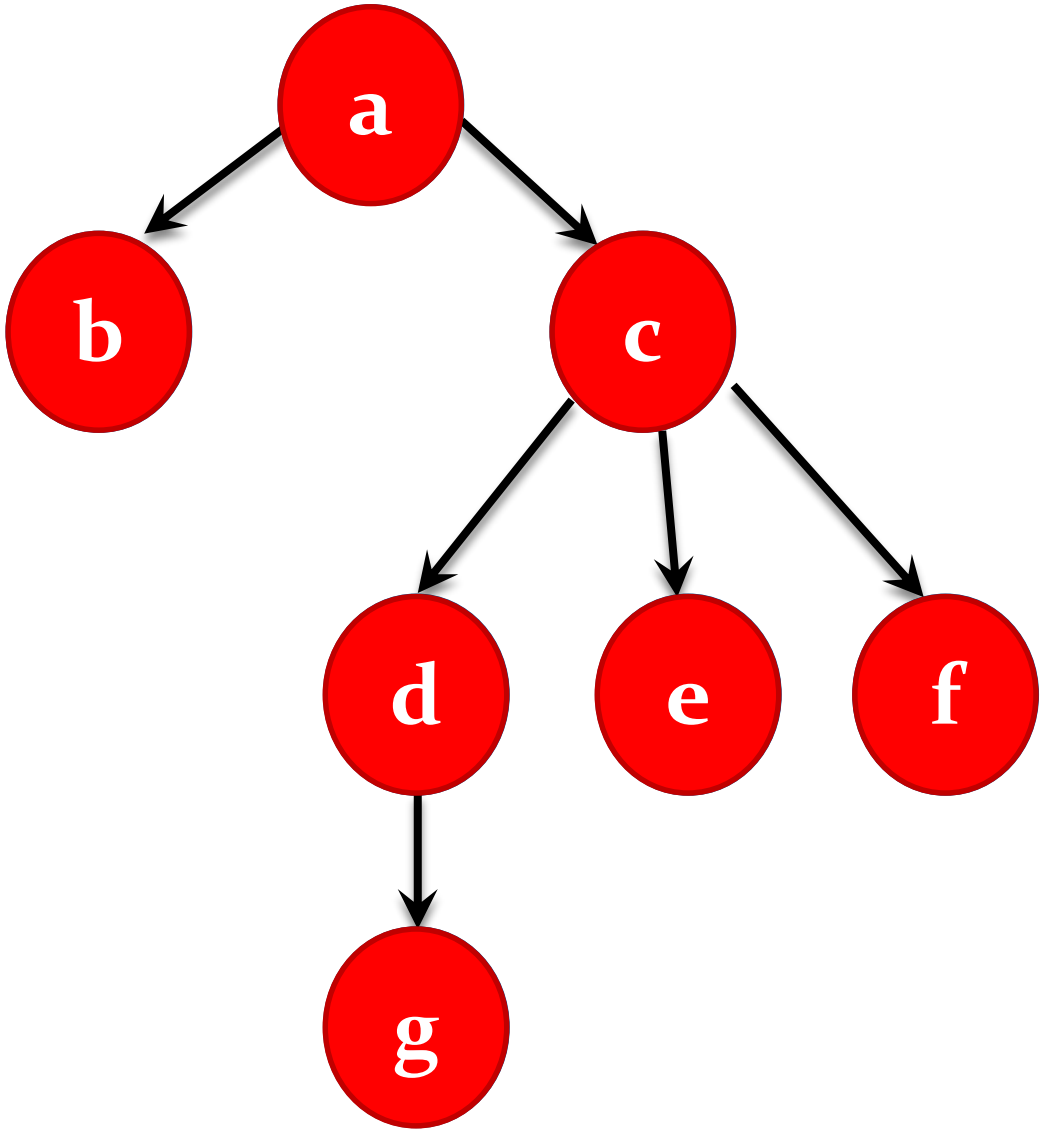
Fák műveletei

- Fák bejárása
  - A fa csomópontjaiban általában adatokat tárolunk. Ezeket valamilyen sorrendben szeretnénk egymás után elérni.
- Általános fa esetén a bejárási stratégiák
  - Gyökérkezdő (preorder)
    - gyökér, majd a részfák bejárása sorban
      - például balról jobbra
  - Gyökérvégző (postorder)
    - részfák bejárása sorban, majd a gyökér

Preorder bejárás

Gyökér, majd a részfák bejárása sorban (balról jobbra)

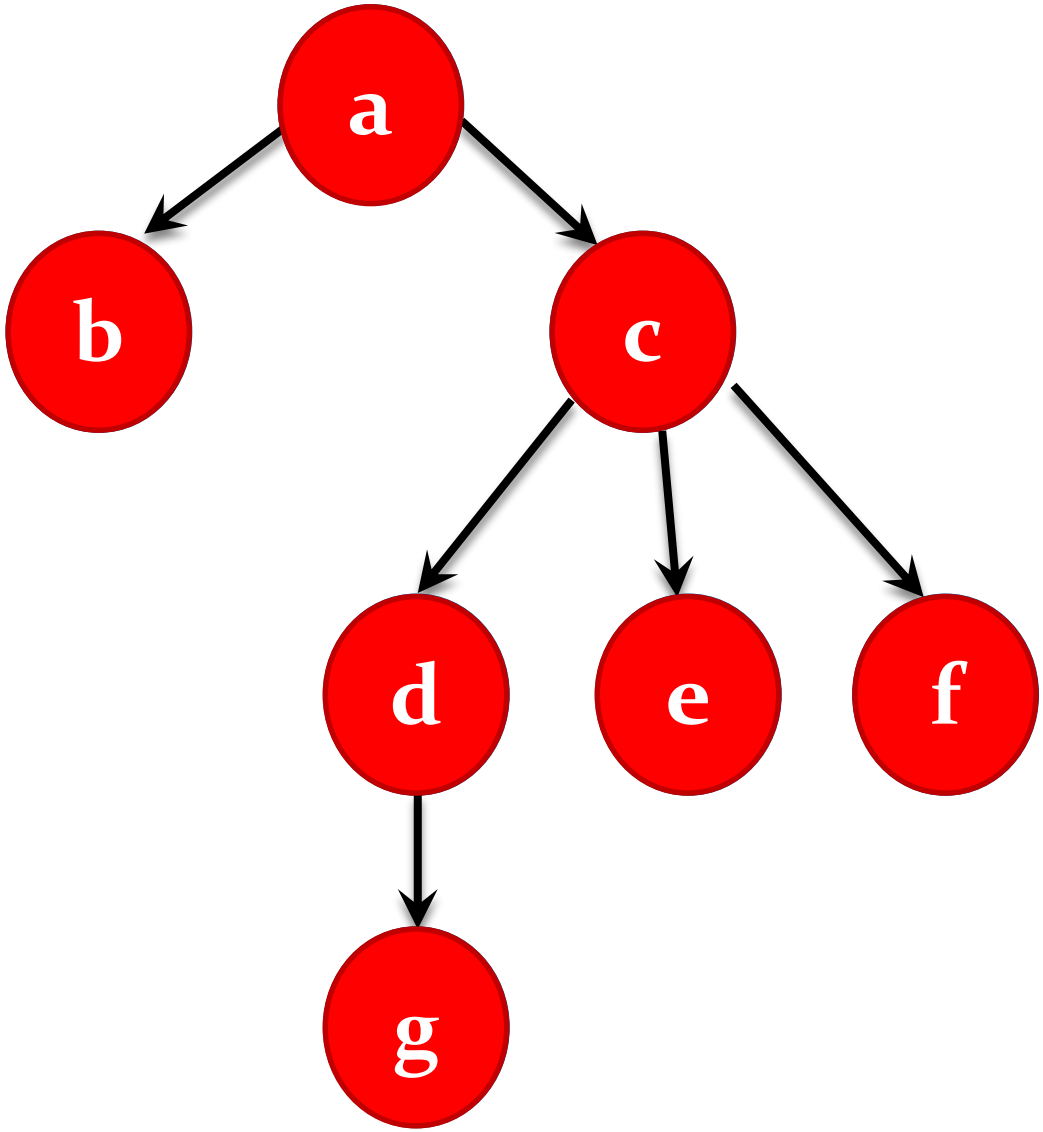
a b c d g e f



Postorder bejárás

Részfák bejárása
sorban, majd a gyökér

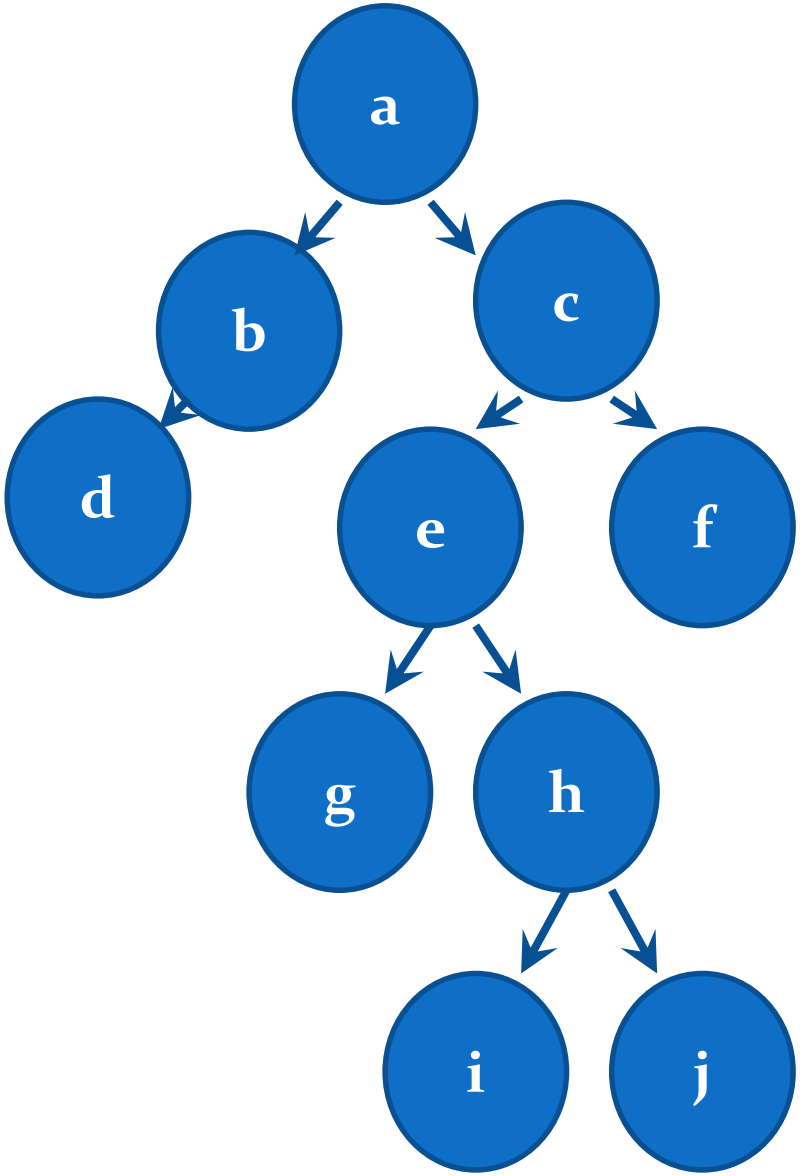
b g d e f c a



Bináris fák

- A bináris fa olyan fa, amelynek csúcspontjaiból ***maximum*** 2 részfa nyílik
  - Azaz fokszáma 2
- A szülő mindig a gyerekek között helyezkedik el
  - Van értelme a „gyökérközepű” (inorder) bejárásnak

Bináris fa



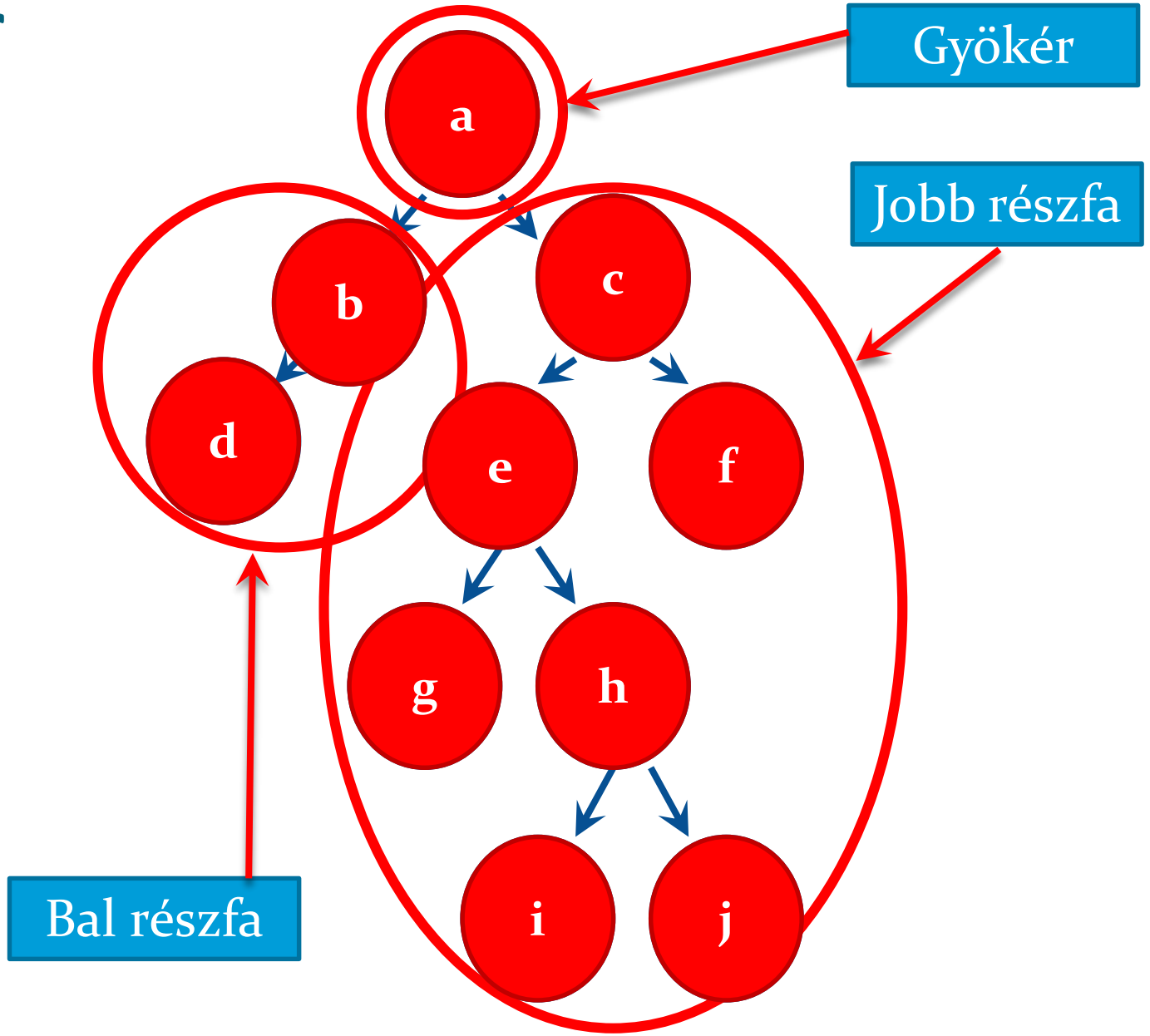
Bináris fák bejárása

- A bejárási stratégiák
  - Gyökérkezdő (preorder)
    - gyökér, bal részfa, jobb részfa
  - Gyökérközepű (inorder)
    - bal részfa, gyökér, jobb részfa
  - Gyökérvégző (postorder)
    - bal részfa, jobb részfa, gyökér

Preorder

Gyökér,
Bal részfa,
Jobb részfa

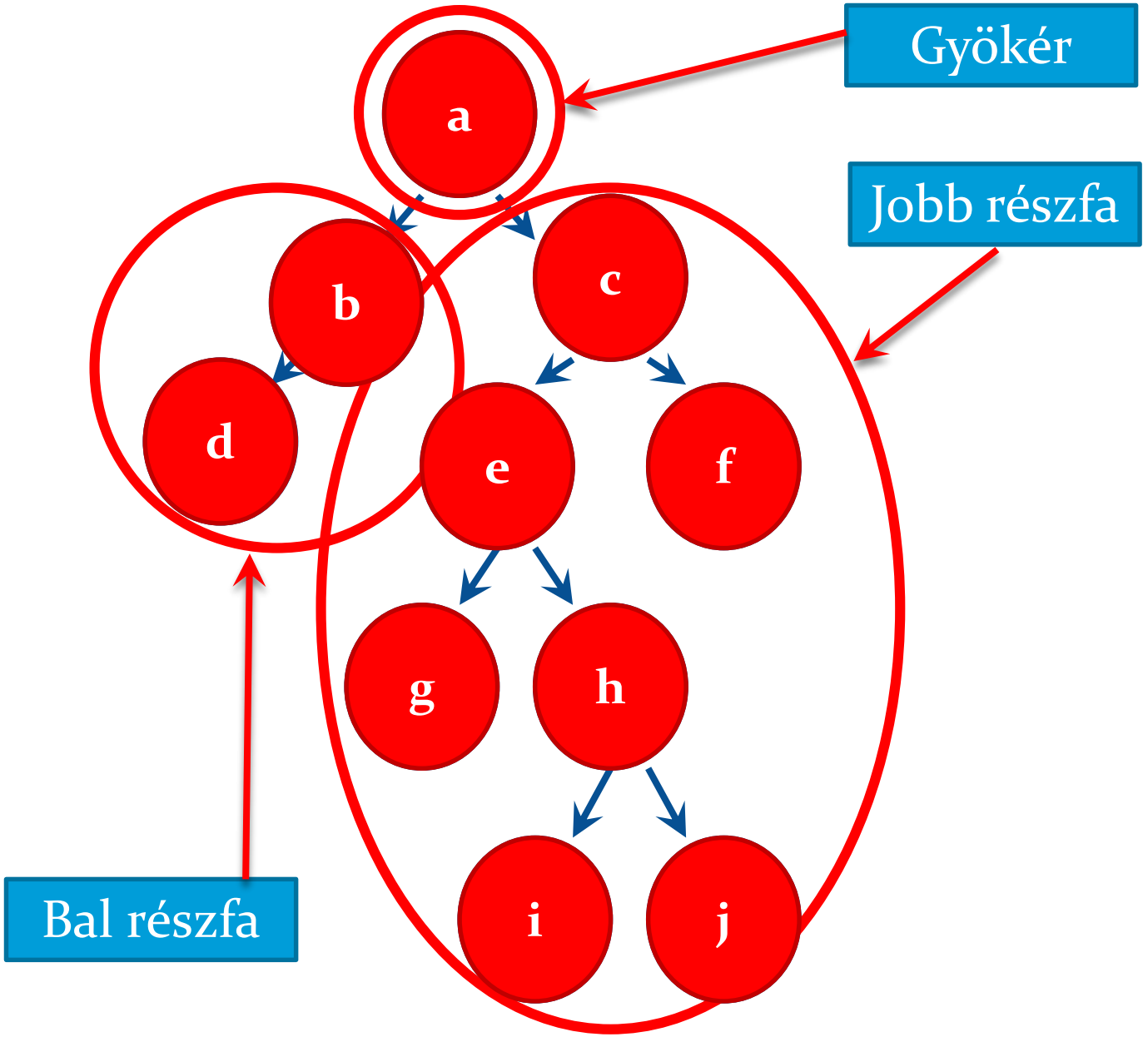
a b d c e g h i j f



Inorder

Bal részfa,
Gyökér,
Jobb részfa

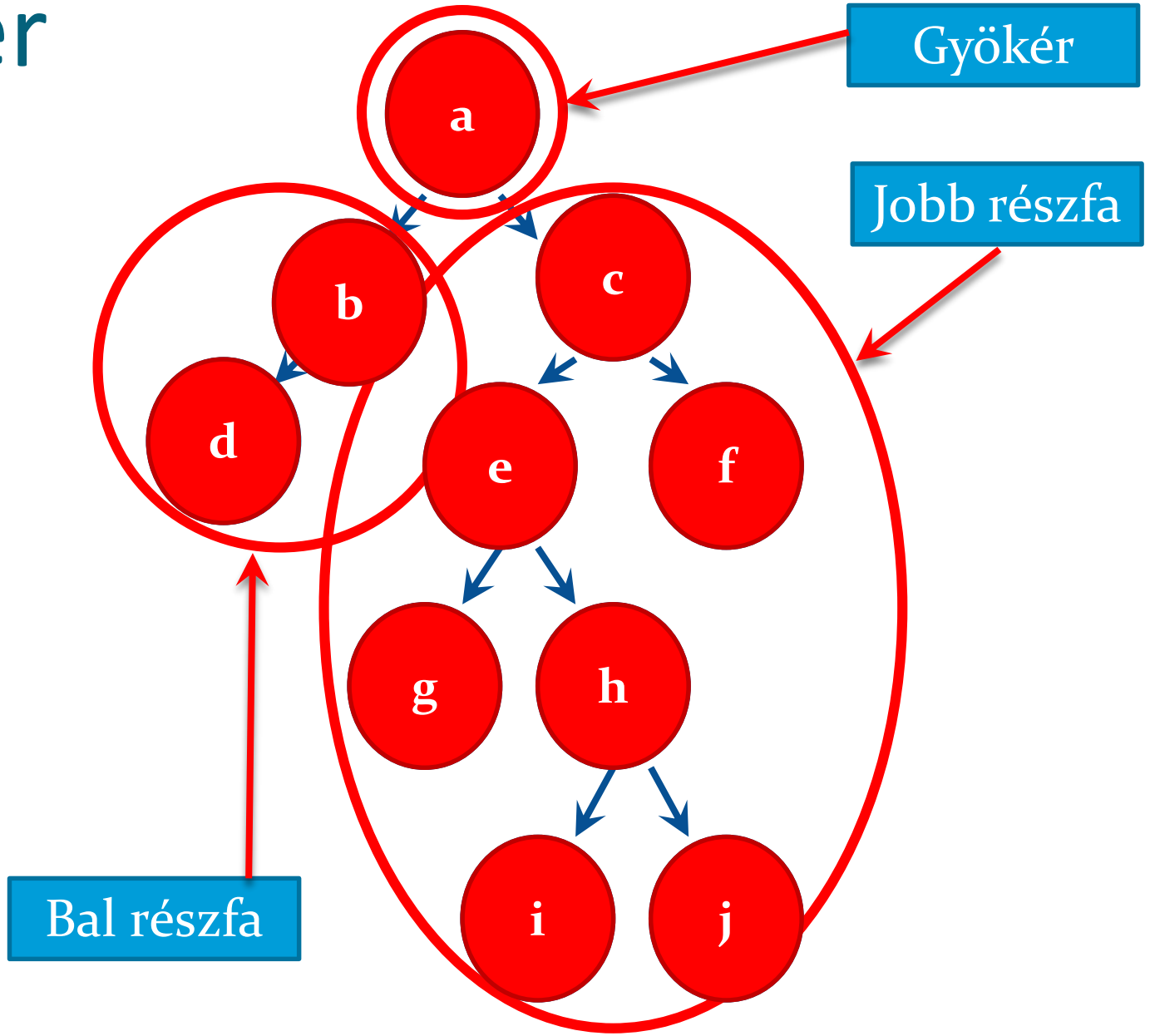
d b a g e i h j c f



Postorder

Bal részfa,
Jobb részfa,
Gyökér

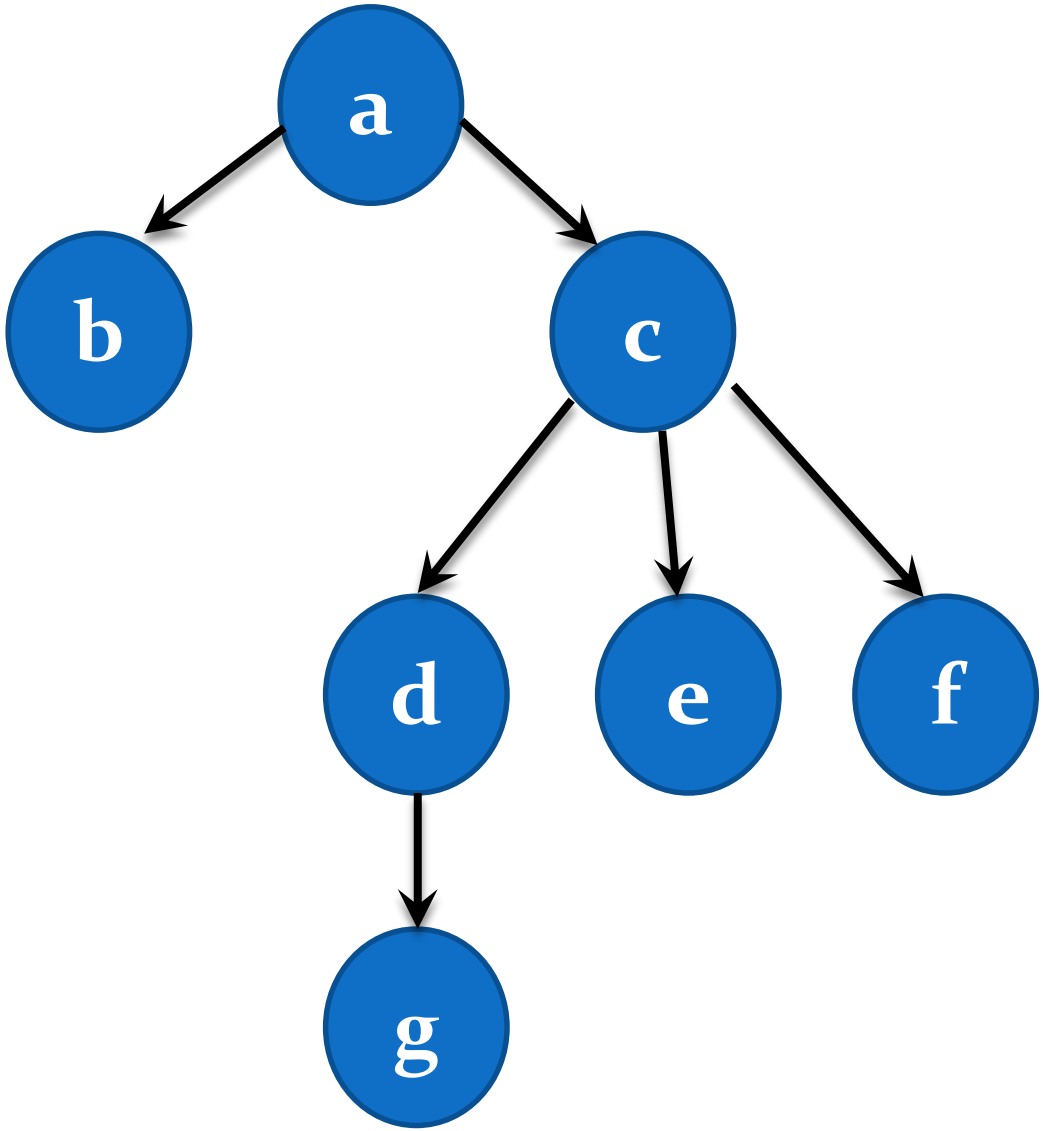
d b g i j h e f c a



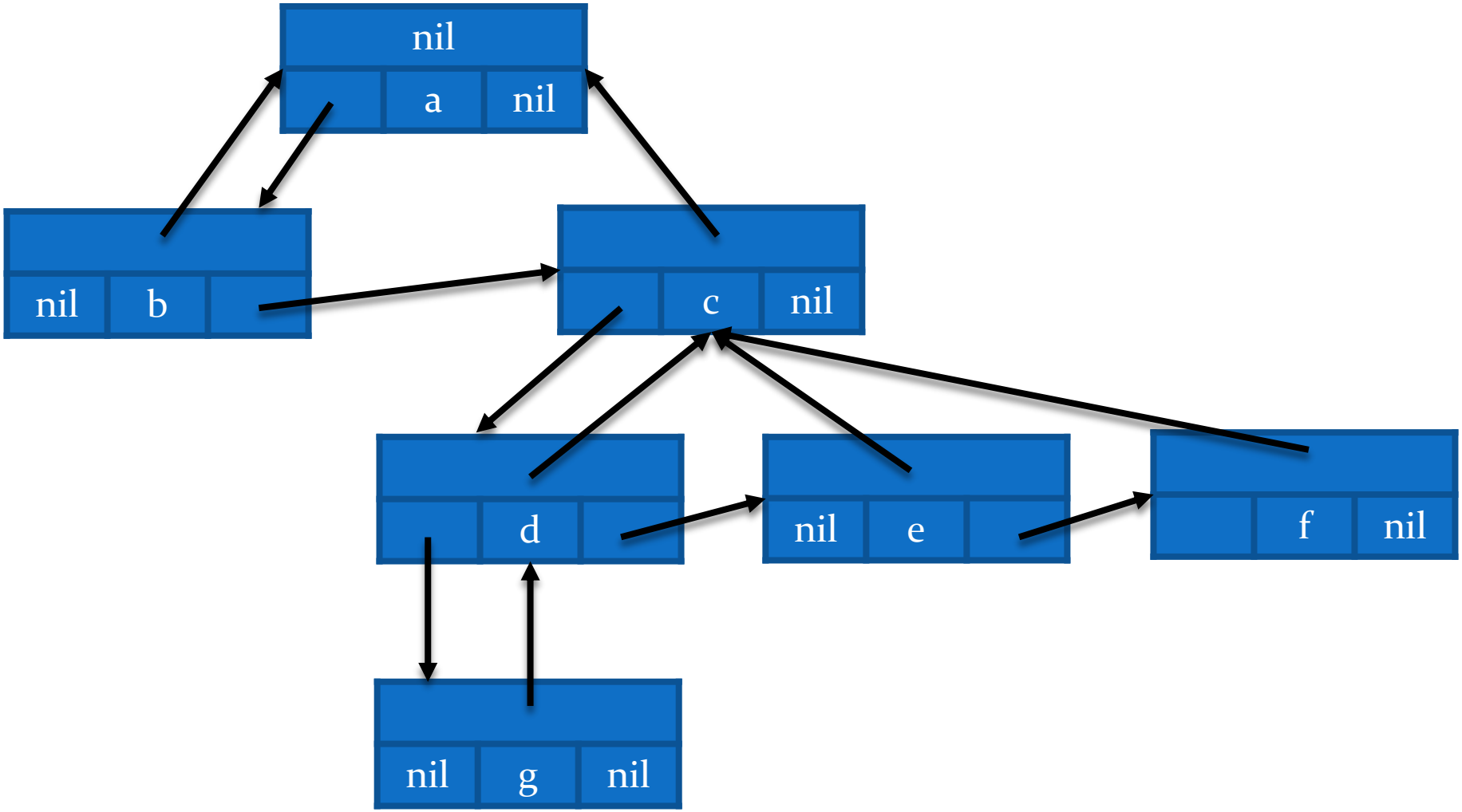
Reprezentáció

- Általános fa esetén
  - „bal-gyermek, jobb-testvér”
  - Minden csomóponthoz tartozik három mutató
    - bal-gyermek – a csúcs gyermekei közül a bal szélsőre mutat
    - jobb-testvér – a csúcsnak arra a testvérére mutat, amelyik közvetlenül jobbra mellette található (azonos szinten ugyanahhoz az őshöz tartozó következő szomszédos elemre)
    - szülő

Általános fa



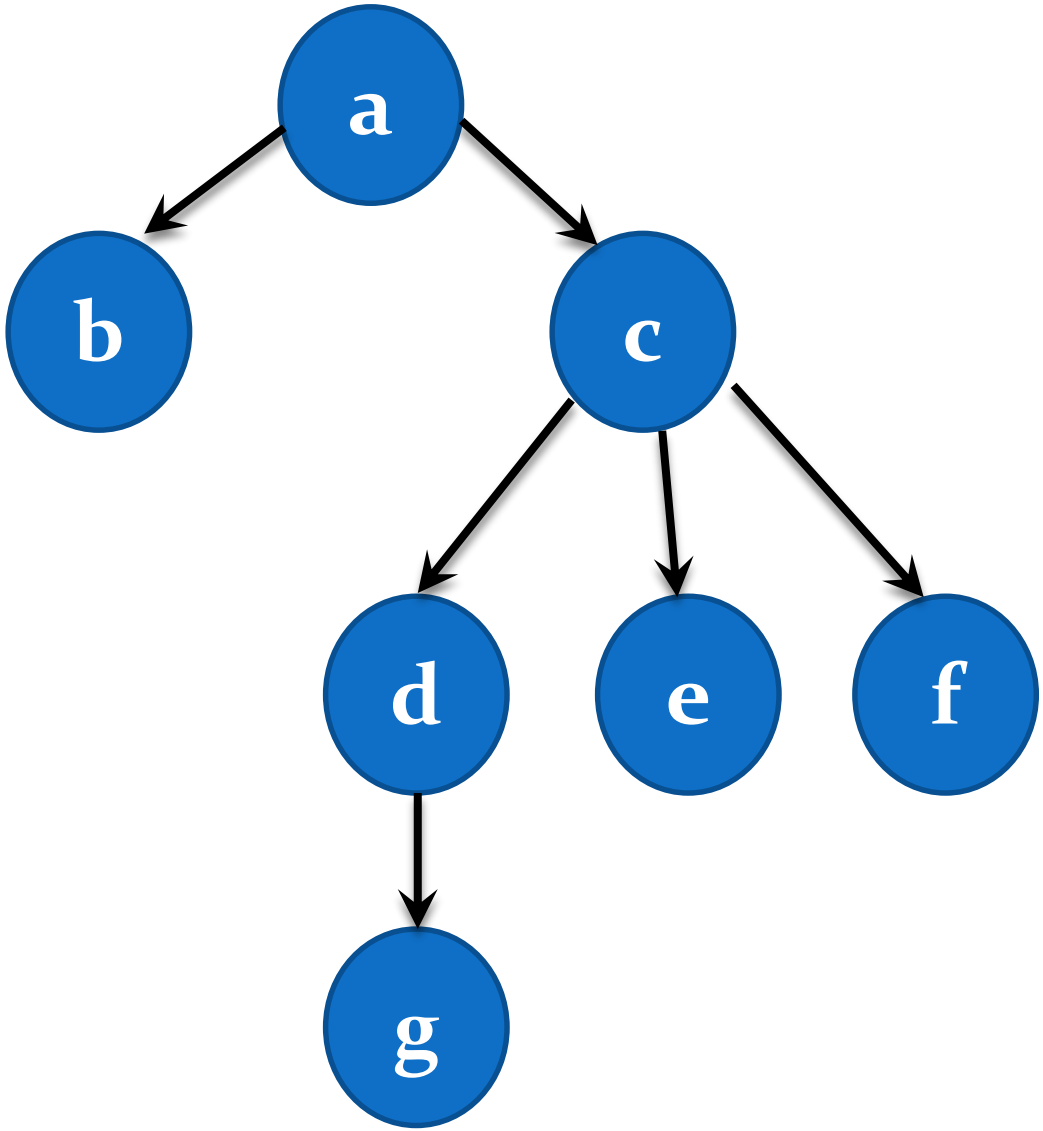
Bal-gyermek, jobb-testvér



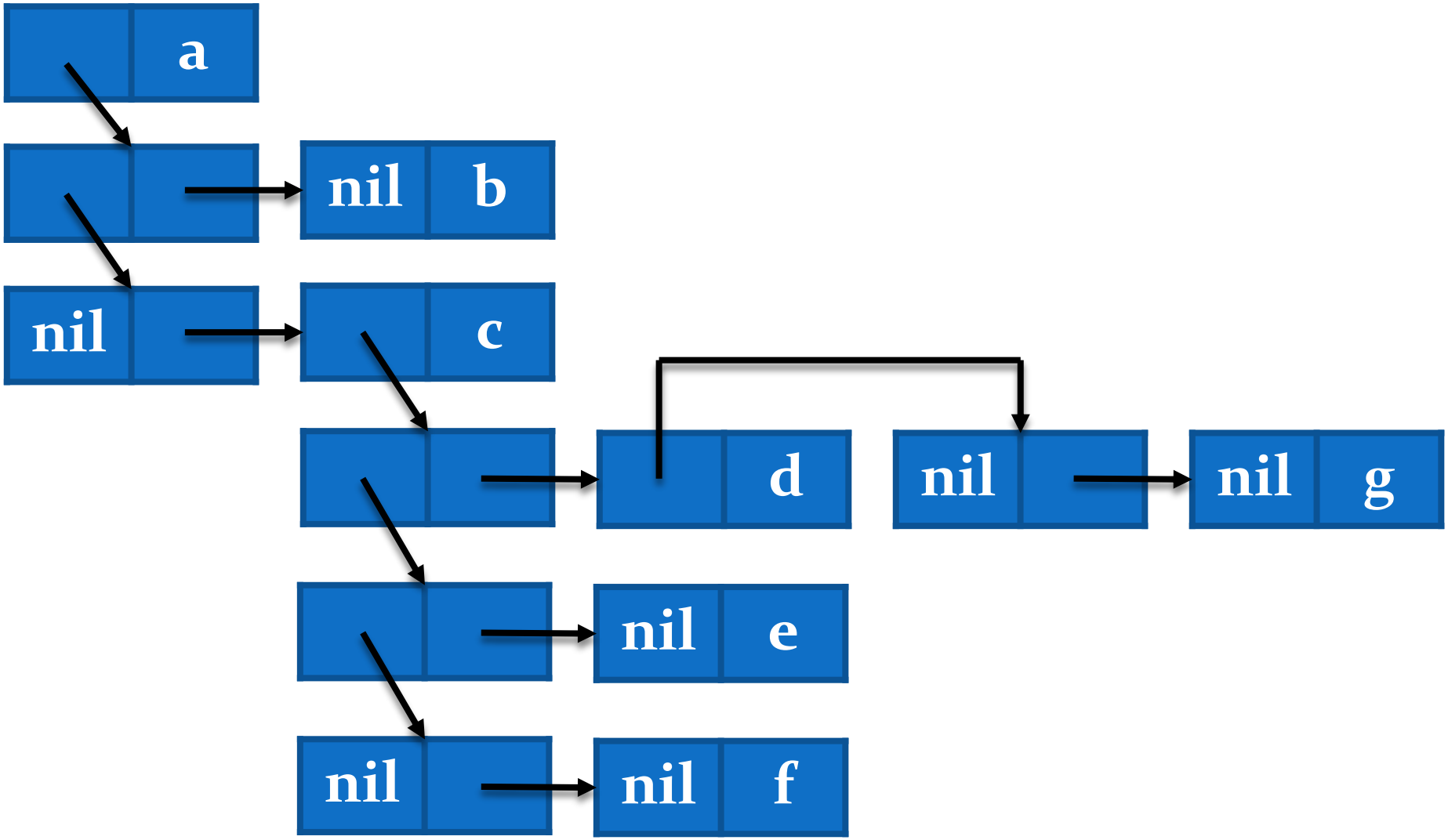
Reprezentáció

- Általános fa esetén például multilista:
  - Minden csomóponthoz tartozik egy lineáris lista, amelynek első eleme az adat, a többi a kapcsolatok listája
  - Annyi kapcsolati elem, ahány fokú a csomópont.
  - A kapcsolatok újabb csomópontokra, illetve lineáris listákra mutatnak.

Általános fa



Multilista

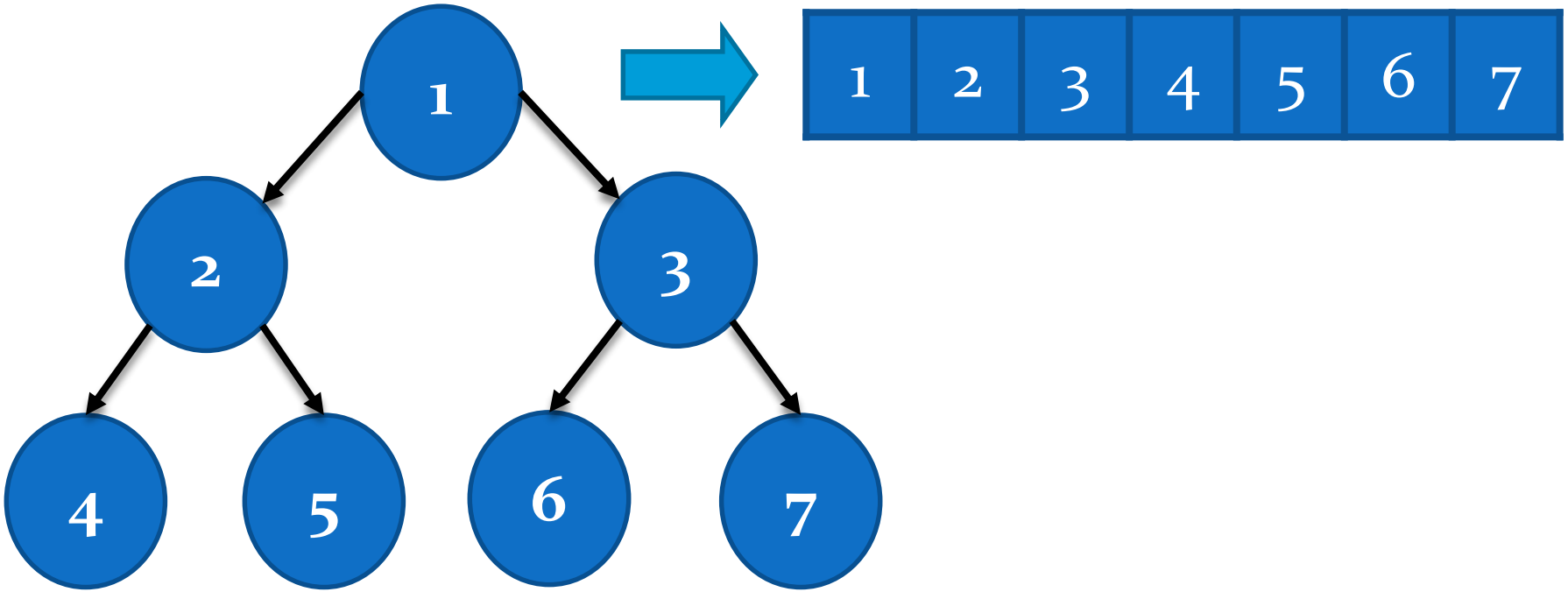


Reprezentáció

- Korlátos általános fa esetén további lehetőség
  - Aritmetikai ábrázolás
  - Láncolt, ahol minden csomópontnak van pontosan k db mutatója a maximum k gyerekre

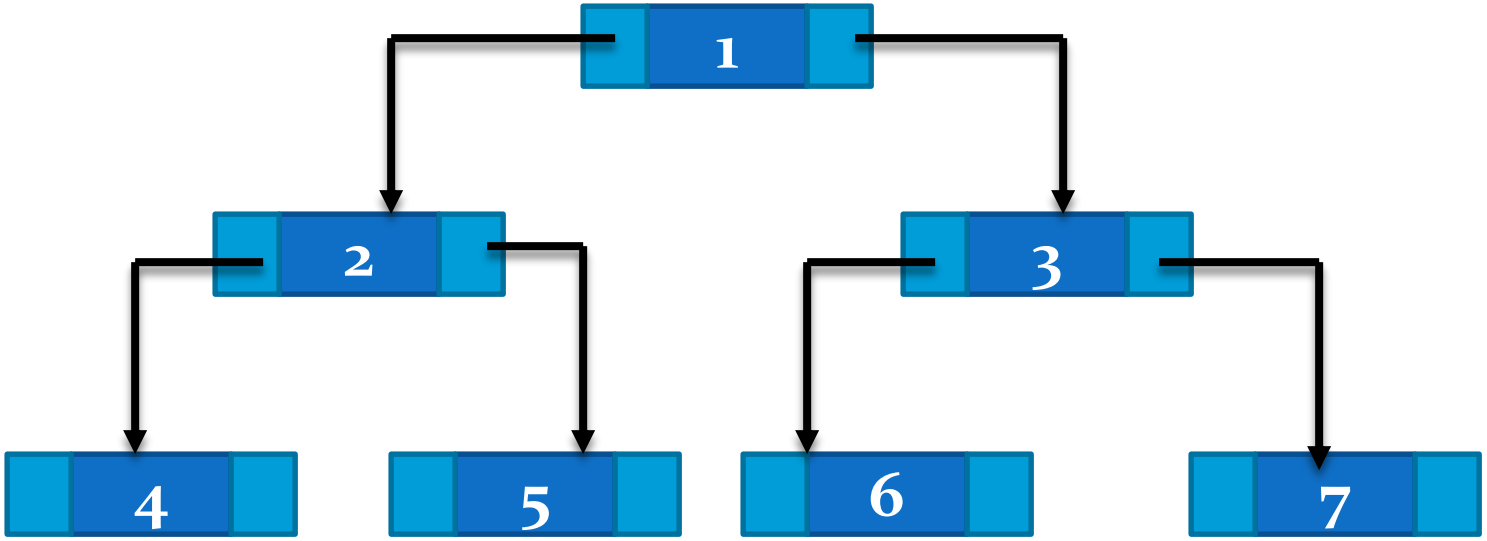
Reprezentáció

- Bináris fa
  - Aritmetikai ábrázolás: szintfolytonosan egy tömbben
  - $\text{ind}(\text{bal}(c)) = 2 * \text{ind}(c)$
  - $\text{ind}(\text{jobb}(c)) = 2 * \text{ind}(c) + 1$



Reprezentáció

- Bináris fa
  - Láncolt – mutató a bal és a jobb gyerekre
    - esetenként a szülőre is



Bináris fák

- Definíciók
  - Egy bináris fa akkor tökéletesen kiegyensúlyozott, ha minden elem bal-, illetve jobboldali részfájában az elemek száma legfeljebb eggyel tér el
  - Teljesnek nevezünk egy bináris fát, ha minden közbenső elemének pontosan két leágazása van
  - Majdnem teljes: ha csak a levelek szintjén van esetleg hiány

Bináris fák

- Lehetséges műveletek
  - üres fa inicializálása
  - az üres fa gyökéreleméneknek definiálása
  - a gyökér és a két részfa csoportosítása (az egyik részfa lehet üres)
  - egy elem hozzáadása egy olyan elem bal (jobb) oldalához, amelynek nincs bal (jobb) oldali leágazása
  - jelezzük, ha a fa üres
  - jelezzük, ha nincs bal (jobb) oldali leágazása az aktuális elemnek

Bináris fák

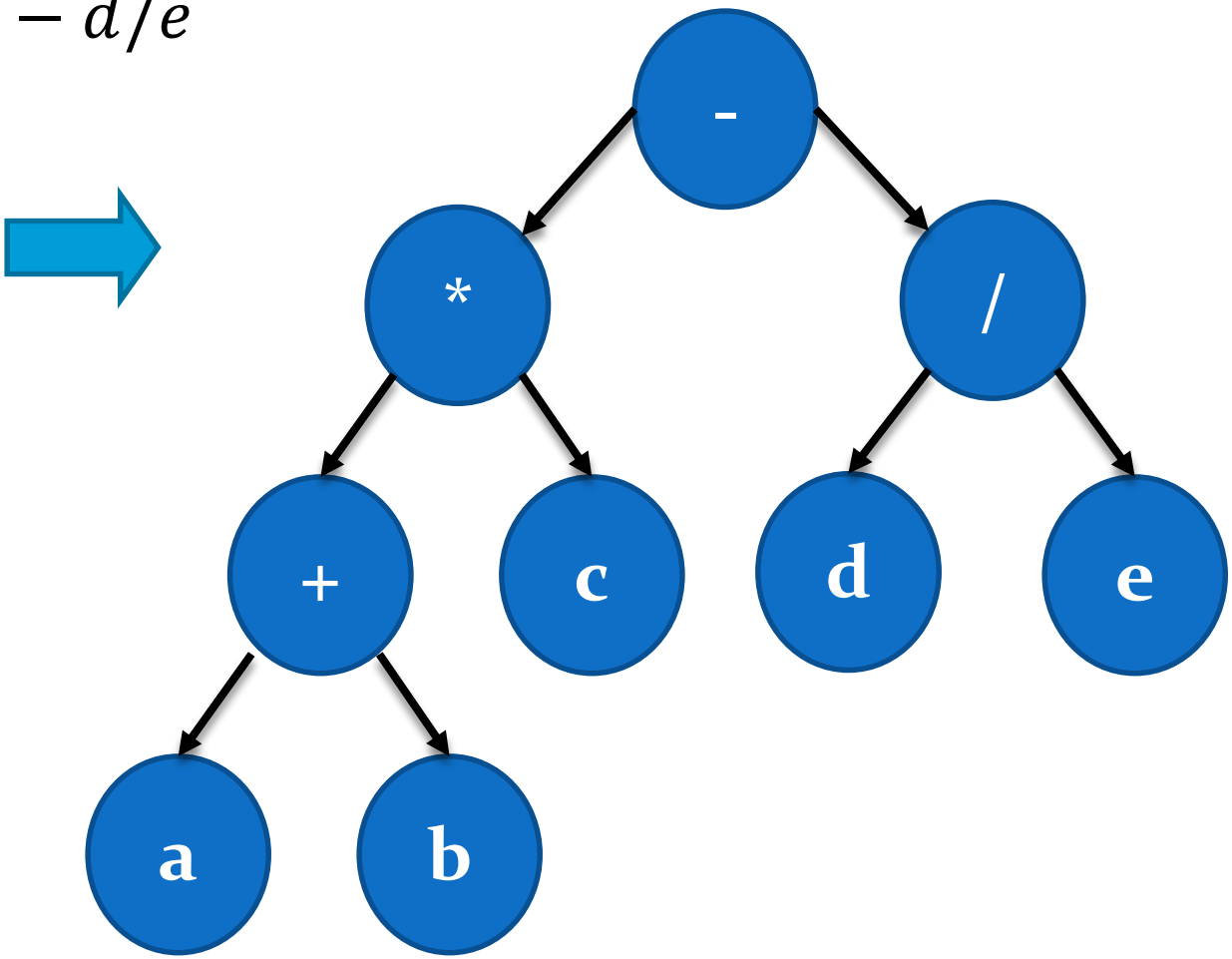
- Lehetséges műveletek (folytatás)
  - a gyökérelem elérése
  - egy adott elem elérése, egy elem bal (jobb) oldali részfájának az elérése a gyökérből
  - egy fa kettéválasztása egy elemre (régi gyökér) és egy vagy két részfára
    - attól függően, hogy a gyökérnek egy vagy két leágazása volt
  - egy (rész-)fa törlése
    - a fa lehet egyetlen elem?
  - egy részfa helyettesítése egy másik részfával …

Bináris fák

- Kiszámítási- vagy kifejezésfa
  - Az a struktúra, amely egy nyelv szimbólumai és különböző műveletei közötti precedenciát jeleníti meg.
  - Aritmetikai kifejezések ábrázolására használják.
  - Minden elágazási pont valamilyen operátort,
  - A levélelemek operandusokat tartalmaznak.
  - A részfák közötti hierarchia fejezi ki az operátorok precedenciáját, illetve a zárójelezést.

Kifejezés fák

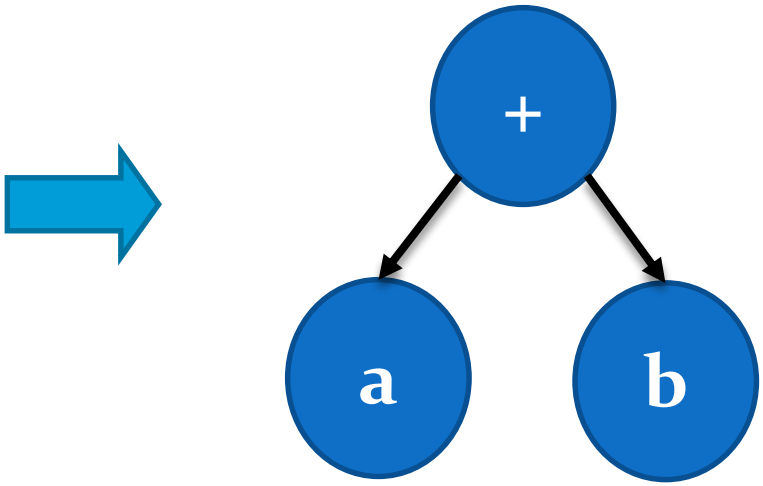
- $(a + b) * c - d/e$



Példa

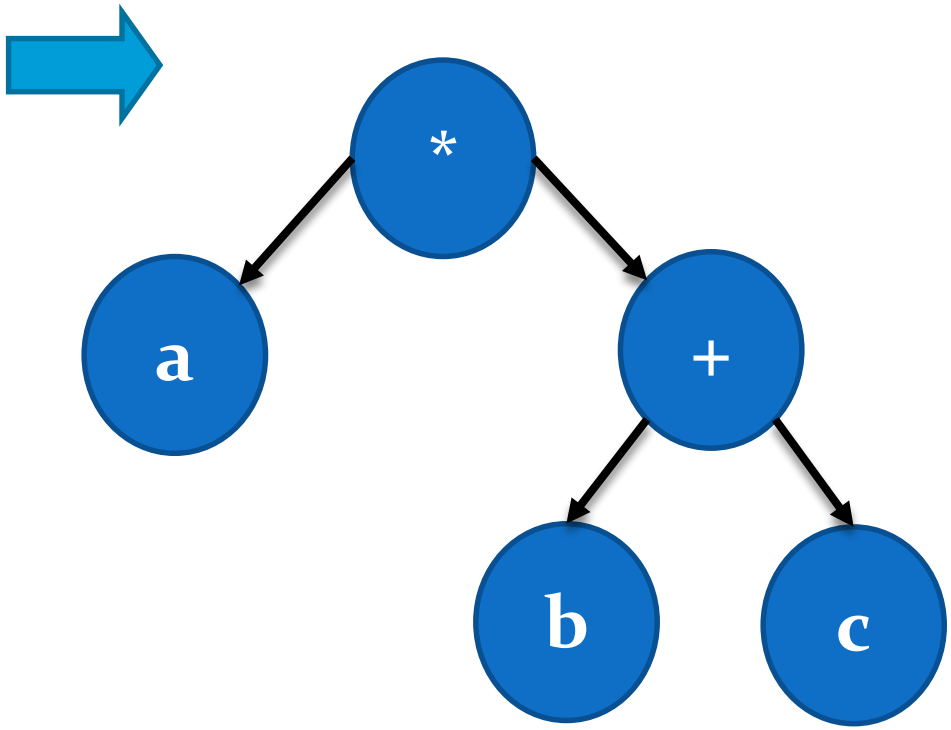
- Legyen adott egy kifejezés lengyel formája az lf sorban, állítsuk elő az f kifejezésfát belőle!

- $a\ b\ +$



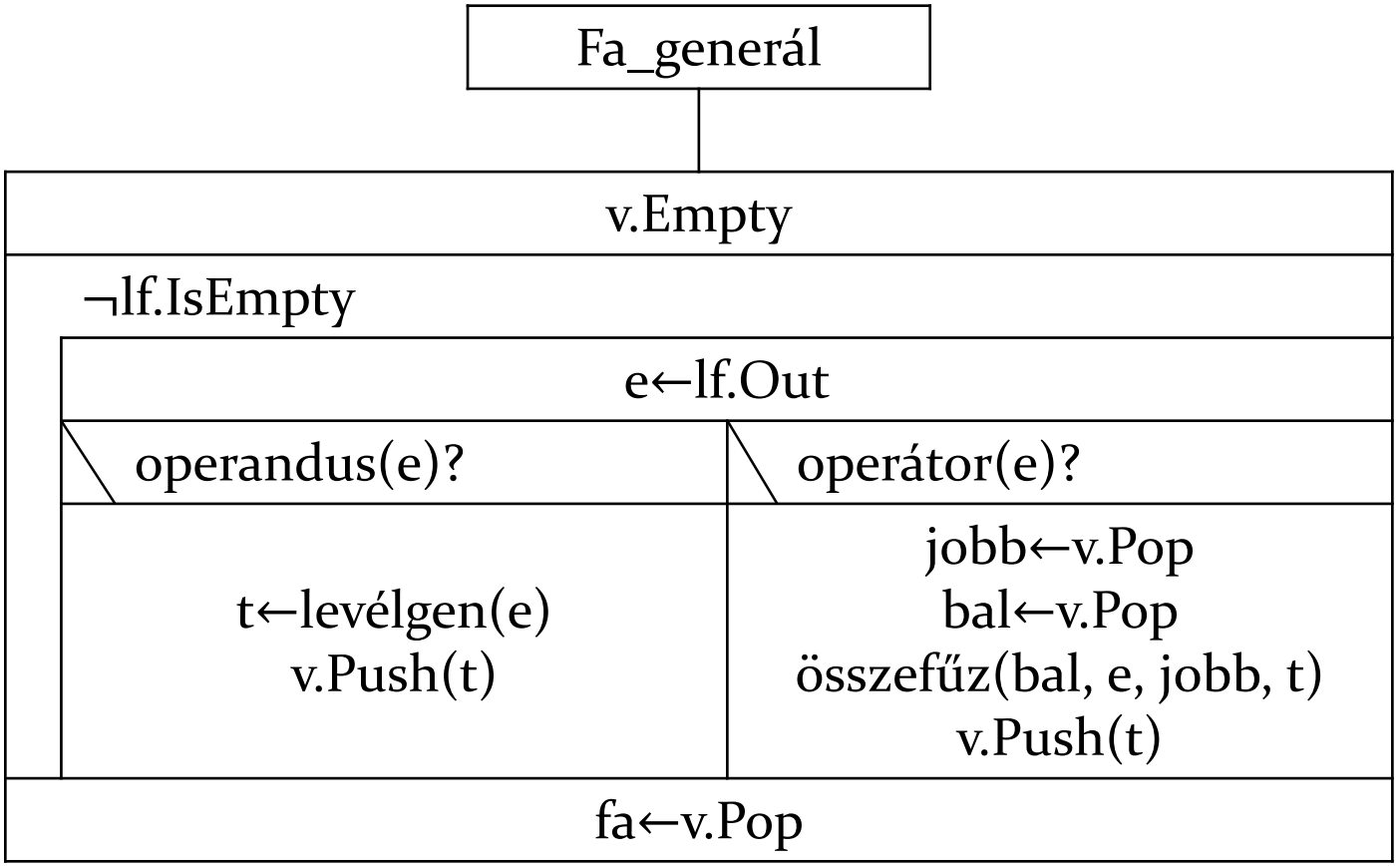
Példa

- $a\ b\ c\ +*$



Algoritmus

- Használjuk a fák vermét!



Algoritmus folytatása