



# Algoritmusok és adatszerkezetek I.

## 3. Előadás

**Kétirányú listák**

# Tartalom

- Kétirányú listák
- Ciklikus kétirányú listák
- Listakezelő műveletek
- Példák C2L listákra
- Halmazműveletek szigorúan monoton növvő C2L listákra

# Ciklikus kétirányú listák (C2L) (Cyclic two-way list)

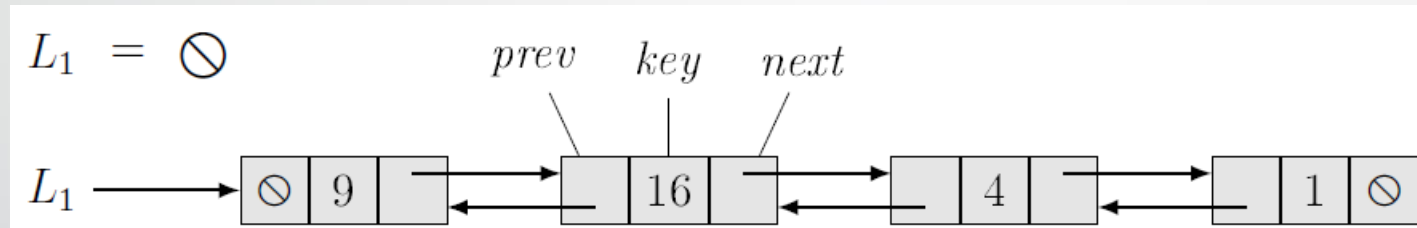
- Fejelemes vs. fejelem nélküli C2L
    - Megegyeznek:
      - a listák elemeinek osztálya ( $E2$ )
      - alapvető listakezelő műveleteik ( $\Theta(1)$ )
      - Sorrend:  $p, q, r$
    - Fejelem használatának előnye:
      - Nem kell külön kezelni:
        - az üres listába való beszúrást
        - az utolsó listaelem törlését
- Listakezelés tovább egyszerűsödik

$E2$
$+prev, next : E2^*$
$+key : \mathcal{T}$
$+ E2() \{ prev := next := \text{this} \}$

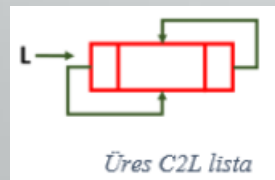
➤ C2L: Fejelemes

# Kétirányú listák (two-way or doubly linked lists)

- Egyszerű kétirányú listák (S2L = Simple 2-way List)
  - a lista módosításakor különbözőképpen kell kezelni a lista első, utolsó és közbülső elemeit
  - Hasító táblánál használjuk



- Fejelemes ciklikus kétirányú listák (C2L = Cyclic 2-way List with header)
  - Gyakoribb
  - listamódosító műveletek egyszerűbbek és hatékonyabbak



# Listakezelő műveletek

$\text{unlink}(q : E2^*)$

// remove ( $*q$ )

$p := q \rightarrow \text{prev} ; r := q \rightarrow \text{next}$

$p \rightarrow \text{next} := r ; r \rightarrow \text{prev} := p$

$q \rightarrow \text{prev} := q \rightarrow \text{next} := q$

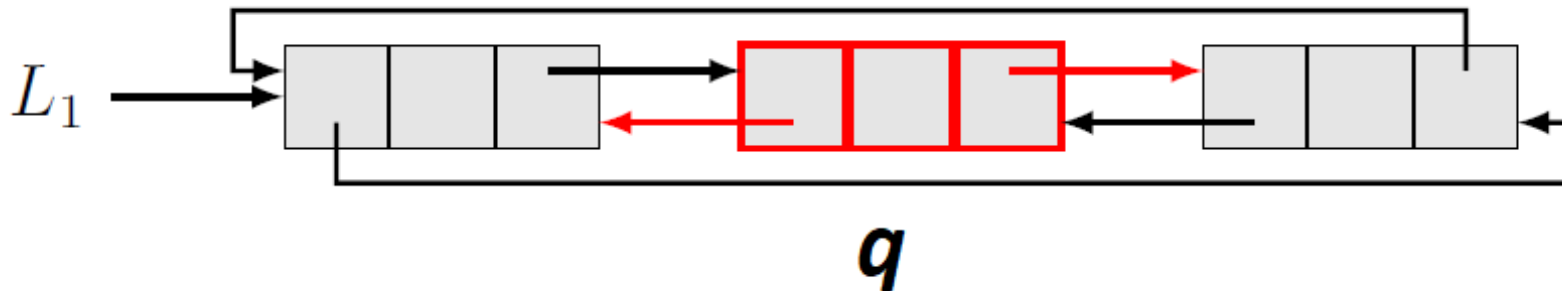
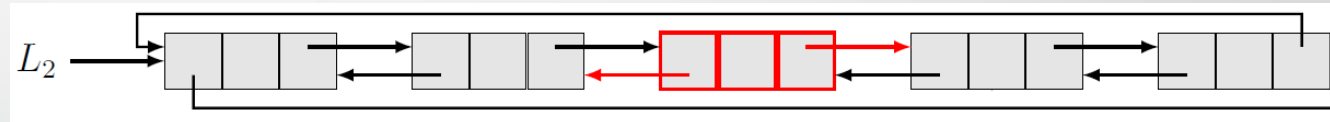
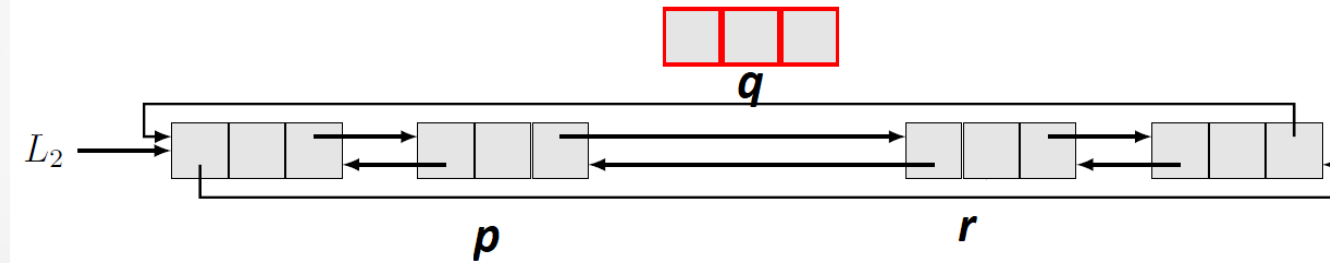
$\text{precede}(q, r : E2^*)$

// ( $*q$ ) will precede ( $*r$ )

$p := r \rightarrow \text{prev}$

$q \rightarrow \text{prev} := p ; q \rightarrow \text{next} := r$

$p \rightarrow \text{next} := r \rightarrow \text{prev} := q$



$\text{follow}(p, q : E2^*)$

// ( $*q$ ) will follow ( $*p$ )

$r := p \rightarrow \text{next}$

$q \rightarrow \text{prev} := p ; q \rightarrow \text{next} := r$

$p \rightarrow \text{next} := r \rightarrow \text{prev} := q$

# Példák C2L listákra: splice listaművelet

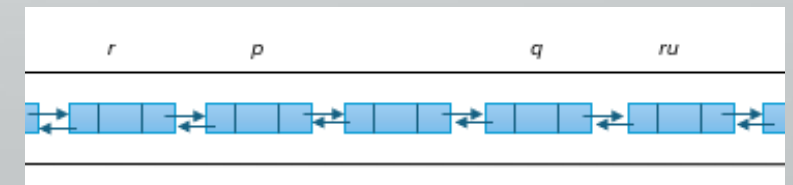
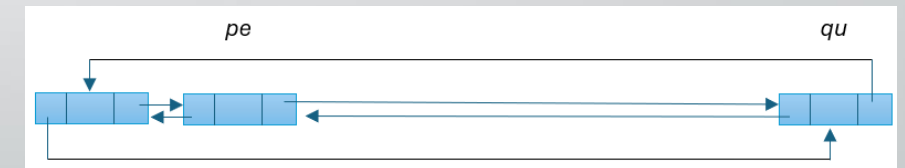
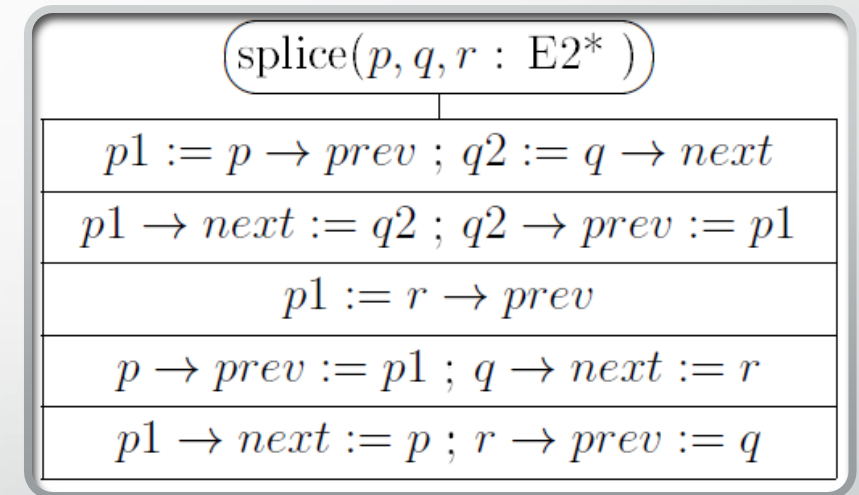
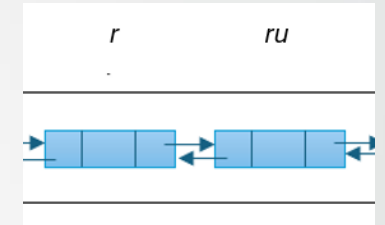
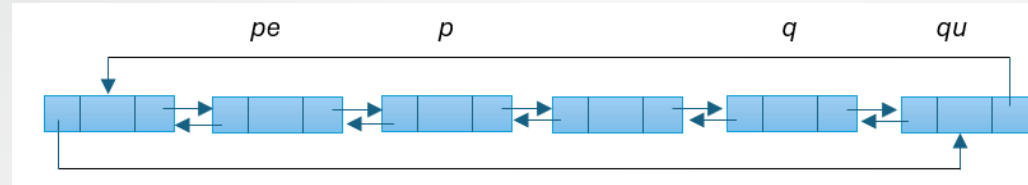
- Előfeltétel:

- $*p$  és  $*q$  ugyanannak a C2L-nek az elemei
- $*p$  után jön valahol  $*q$  ( $p = q$  is lehet)
- a C2L  $[p, \dots, q]$  szakasza nem tartalmaz:
  - sem fejelemet
  - sem a  $*r$  elemet
- $*r$  elem: ennek, vagy egy másik C2L-nek is lehet eleme

- $\text{splice}(p, q, r)$  elemi listaművelet**

- A fenti előfeltétellel
- tetszőleges C2L egy adott  $[p, \dots, q]$  szakaszát eltávolítja
- majd az eltávolított szakaszt egy C2L adott  $*r$  eleme elé fűzi a listába!

**Műveletigény:  $\Theta(1)$**

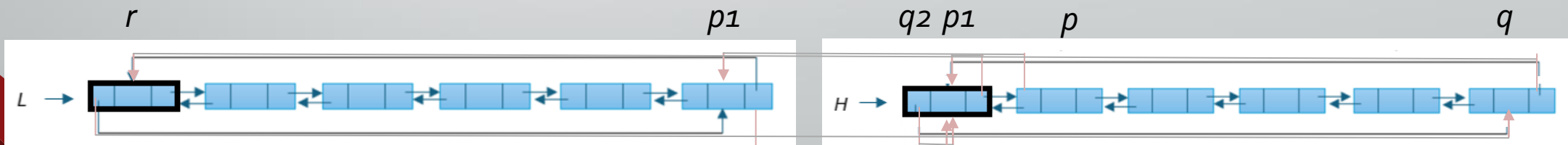


# Példák C2L listákra: append eljárás

- Előfeltétel:
  - $*p$  és  $*q$  ugyanannak a C2L-nek az elemei
  - $*p$  után jön valahol  $*q$  ( $p = q$  is lehet)
  - a C2L  $[p, \dots, q]$  szakasza nem tartalmazza sem a fejeleket, sem a  $*r$  elemet
  - $*r$  elem: ennek, vagy egy másik C2L-nek is lehet eleme
- **append( $L, H$ ) eljárás**
  - átfűzi az  $L$  C2L végére a  $H$  C2L elemeit
  - az eredeti sorrendben
  - Használja a splice( $p, q, r$ )
- **Műveletigény:  $\Theta(1)$**

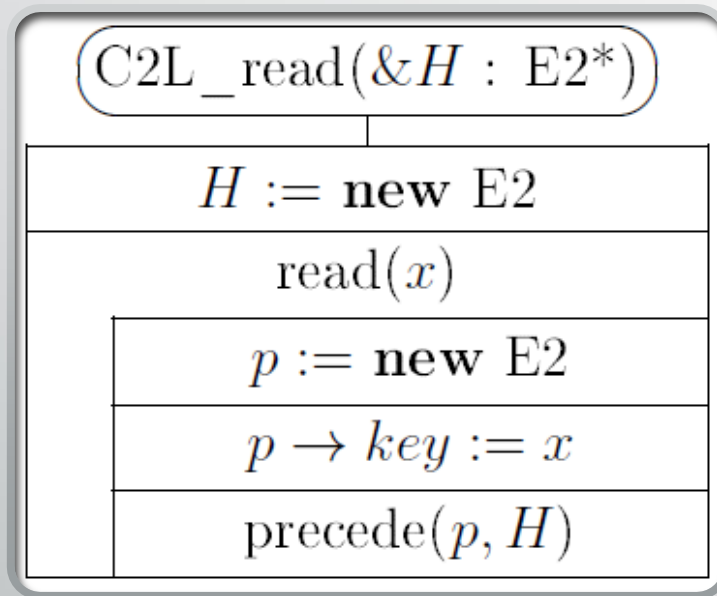
splice( $p, q, r : E2^*$ )	
$p1 := p \rightarrow prev ; q2 := q \rightarrow next$	
$p1 \rightarrow next := q2 ; q2 \rightarrow prev := p1$	
$p1 := r \rightarrow prev$	
$p \rightarrow prev := p1 ; q \rightarrow next := r$	
$p1 \rightarrow next := p ; r \rightarrow prev := q$	

append( $L, H : E2^*$ )	
$H \rightarrow next \neq H$	
splice( $H \rightarrow next, H \rightarrow prev, L$ )	SKIP

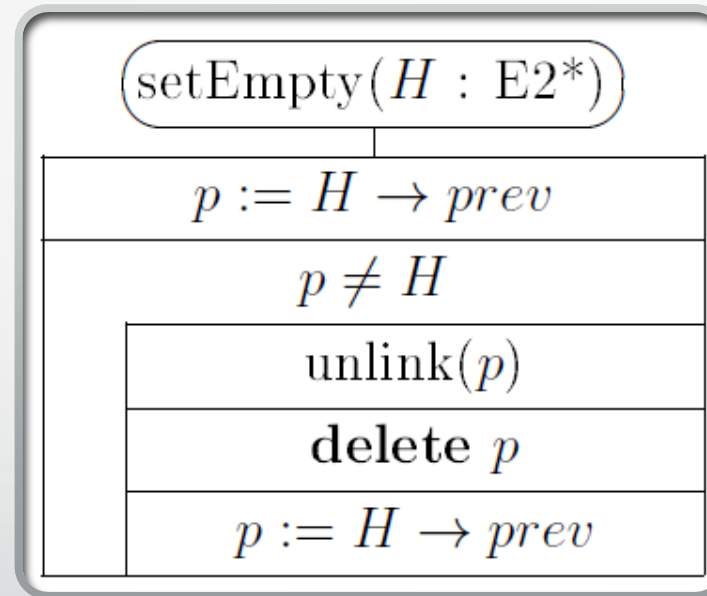




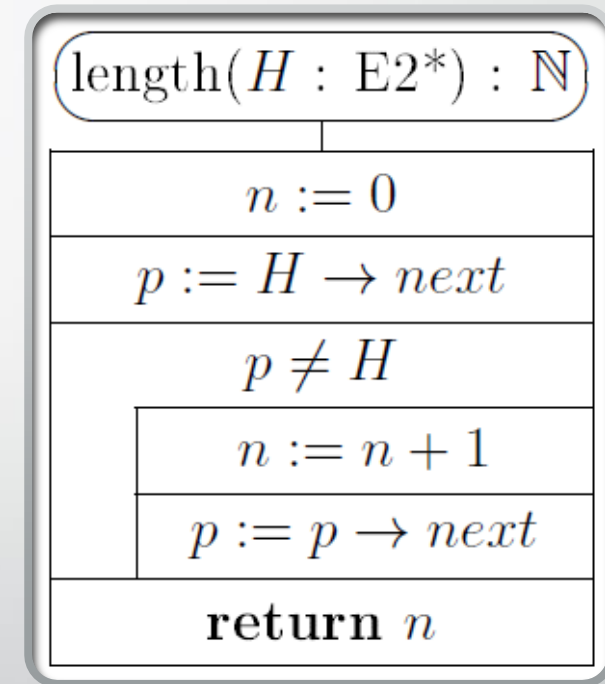
# Példaprogramok C2L listákra



$$T_{\text{C2L\_read}}(n)$$



$$T_{\text{setEmpty}}(n) \in \Theta(n)$$



$$T_{\text{lenght}}(n) \in \Theta(n)$$



# Példaprogramok C2L listákra

s



insertionSort( $H : E2^*$ )

$r := H \rightarrow \text{next} ; s := r \rightarrow \text{next}$

$s \neq H$

$r \rightarrow \text{key} \leq s \rightarrow \text{key}$

unlink( $s$ )

$p := r \rightarrow \text{prev}$

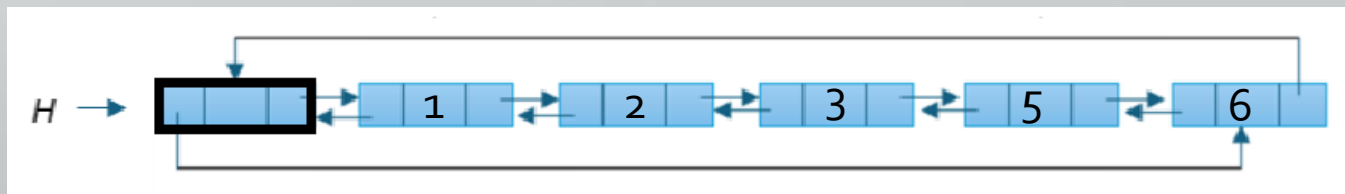
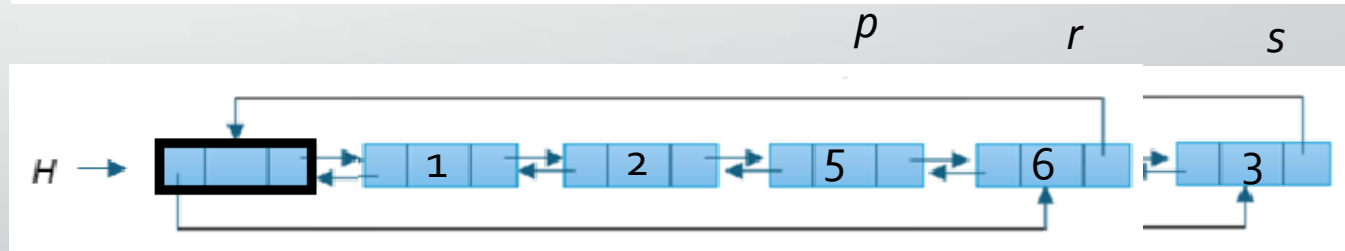
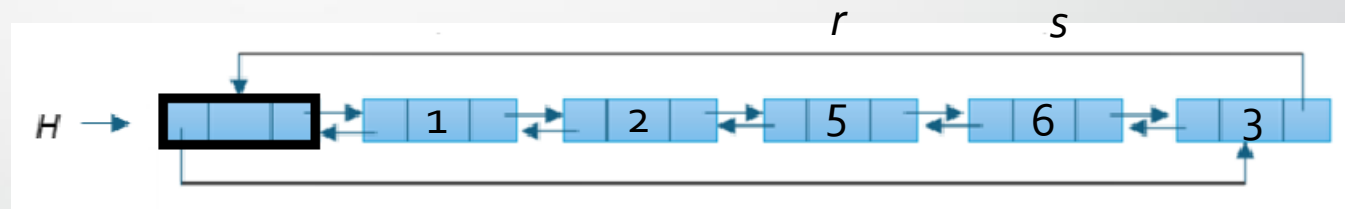
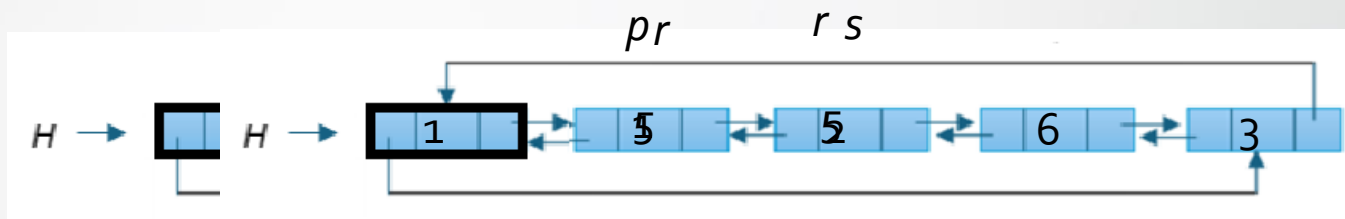
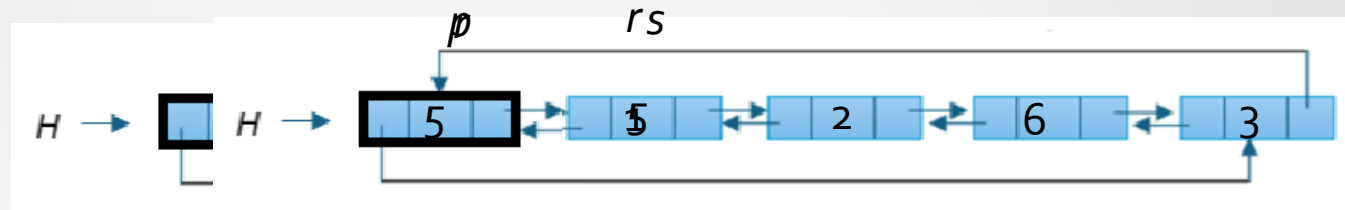
$r := s$

$p \neq H \wedge p \rightarrow \text{key} > s \rightarrow \text{key}$

$p := p \rightarrow \text{prev}$

follow( $p, s$ )

$s := r \rightarrow \text{next}$



$mT_{IS}(n) \in \Theta(n)$

$AT_{IS}(n), MT_{IS}(n) \in \Theta(n^2)$

# Halmazműveletek szig. mon. növő C2L listákra

- Legyenek  $H_u$  és  $H_i$  szigorúan monoton növekvően rendezett C2L-ek! Írjuk meg a **unionIntersection( $H_u, H_i : E2^*$ )** eljárást, ami a  $H_u$  listába  $H_i$  megfelelő elemeit átfűzve, **a  $H_u$  listában** az eredeti listák mint halmazok **unióját** állítja elő, míg **a  $H_i$  listában a metszetük** marad!
- Ne allokáljunk és ne is deallokáljunk listaelemeket, csak az listaelemek átfűzésével oldjuk meg a feladatot!
- **$MT(n_u, n_i) \in \Theta(n_u + n_i)$** , ahol a  $H_u$  C2L hossza  $n_u$ , a  $H_i$  C2L hossza pedig  $n_i$ .
- Minkét lista maradjon szigorúan monoton növekvően rendezett C2L!

unionIntersection( $H_u, H_i : E2*$ )			
$q := H_u \rightarrow next ; r := H_i \rightarrow next$			
$q \neq H_u \wedge r \neq H_i$			
$\backslash q \rightarrow key < r \rightarrow key$	$\backslash q \rightarrow key > r \rightarrow key$	$\backslash q \rightarrow key = r \rightarrow key$	
$q := q \rightarrow next$	$p := r$	$q := q \rightarrow next$	
	$r := r \rightarrow next$		$r := r \rightarrow next$
	unlink( $p$ )		
	precede( $p, q$ )		
$r \neq H_i$			
$p := r ; r := r \rightarrow next ; unlink(p)$			
precede( $p, H_u$ )			

# Halmazműveletek szig. mon. növő C2L listákra

- **unionIntersection( $H_u, H_i : E2^*$ )**

- $H_u$  és  $H_i$  szig. Mon. növő rendezett C2L-ek
- $H_i$  megfelelő elemeit átfűzi a  $H_u$  listába



- **Eredmény:**

- **$H_u$  listában:** az eredeti listák  $\sim$  halmazok **uniója**
- **$H_i$  listában:** a **metszetük** marad

- **Feltételek:**


- Csak az listaelemek átfűzése
  - Nincs allokáció/deallokáció
- **$MT(n_u, n_i) \in \Theta(n_u + n_i)$** 
  - $H_u$  C2L hossza:  $n_u$
  - $H_i$  C2L hossza:  $n_i$
- Minkét lista maradjon szigorúan monoton növekvően rendezett C2L

unionIntersection( $H_u, H_i : E2*$ )		
$q := H_u \rightarrow next ; r := H_i \rightarrow next$		
$q \neq H_u \wedge r \neq H_i$		
$\backslash q \rightarrow key < r \rightarrow key$	$\backslash q \rightarrow key > r \rightarrow key$	$\backslash q \rightarrow key = r \rightarrow key$
$q := q \rightarrow next$	$p := r$	$q := q \rightarrow next$
	$r := r \rightarrow next$	
	unlink( $p$ )	$r := r \rightarrow next$
	precede( $p, q$ )	
$r \neq H_i$		
$p := r ; r := r \rightarrow next ; unlink(p)$		
precede( $p, H_u$ )		

# További feladatok:

- Legyenek  $H_u$  és  $H_i$  szigorúan monoton növekvően rendezett C2L-ek! Írjuk meg a **unionIntersection( $H_u, H_i : E2^*$ )** eljárást, ami a  $H_u$  listába  $H_i$  megfelelő elemeit átfűzve, **a  $H_u$  listában** az eredeti listák mint halmazok **unióját** állítja elő, míg **a  $H_i$  listában a metszetük** marad!
- Ne allokáljunk és ne is deallokáljunk listaelemeket, csak az listaelemek átfűzésével oldjuk meg a feladatot!
- **$MT(n_u, n_i) \in \Theta(n_u + n_i)$** , ahol a  $H_u$  C2L hossza  $n_u$ , a  $H_i$  C2L hossza pedig  $n_i$ .
- Minkét lista maradjon szigorúan monoton növekvően rendezett C2L!

unionIntersection( $H_u, H_i : E2*$ )		
$q := H_u \rightarrow next ; r := H_i \rightarrow next$		
$q \neq H_u \wedge r \neq H_i$		
$\backslash q \rightarrow key < r \rightarrow key$	$\backslash q \rightarrow key > r \rightarrow key$	$\backslash q \rightarrow key = r \rightarrow key$
$q := q \rightarrow next$	$p := r$	$q := q \rightarrow next$
	$r := r \rightarrow next$	
	unlink( $p$ )	$r := r \rightarrow next$
	precede( $p, q$ )	
$r \neq H_i$		
$p := r ; r := r \rightarrow next ; unlink(p)$		
precede( $p, H_u$ )		



# Köszönöm a figyelmet!

**Pusztai Kinga**

A bemutató Ásványi Tibor: Algoritmusok és adatszerkezetek I.  
Előadásjegyzete alapján készült.