$((a * b) + (c / (d \uparrow e)))$ $\xrightarrow{1.}$ $ab*cde\uparrow/+$

$\xleftarrow{2.}$

3. ↓

kiértékelése

$a = 3$

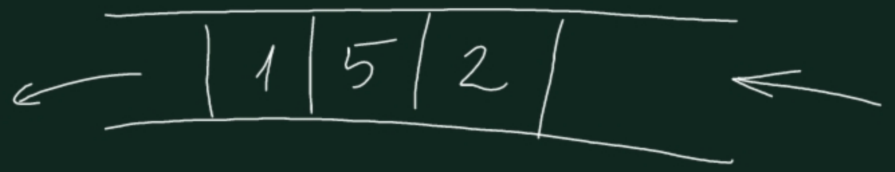
$b = 4$

c ...

veremmel

1, v. 2

Sort Queue FIFO



Q: Queue(n)

Q.add(x)
Q.rem()

Q.first()
Q.isEmpty()

Példa sor alkalmazásra: dadogós szöveg

Oldjuk meg *két sor* segítségével a következő feladatot:

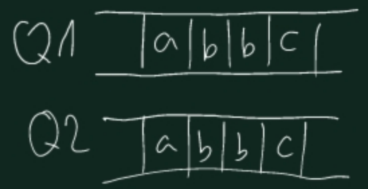
Olvassunk be karakterenként egy szöveget (hossza nem ismert), és döntsük el, hogy „dadogós" –e.

Pl.: *abcabc* dadogós, *abccbb* nem dadogós

2-szer ugyanaz a szöveg egymás után

abaaba

```
Dadogós(n:N):B
Q1,Q2:Queue(n); h:=0
read(x)
  Q1.add(x); h++
h≡0 (mod 2)
SKIP | return FALSE
i=1 to h/2
  Q2.add(Q1.rem())
i=1 to h/2
  Q1.rem()=Q2.rem()
  SKIP | return FALSE
return TRUE
```



További sorokkal, vermekkel megoldható feladatok:

→ abc#abcdde#ddeee#e##

↪ abbabb#ad#

→ abba#cdefedc#...

gyakorlásképpen: ↖ ilyeneket ellenőrző algoritmus?

Két sor összefésülése

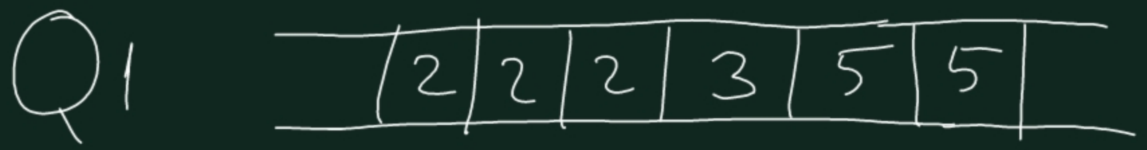
Q1 és Q2 sorokban számok rendezetten
$\longrightarrow$ fésüljük össze Q3-ba rendezetten

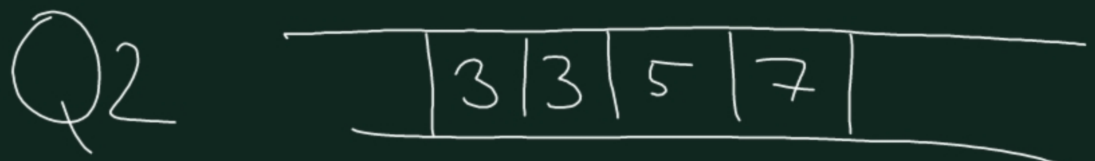
esetek: – lehet-e több példány egy elemből Q1, Q2-ben?
Q3-ban?

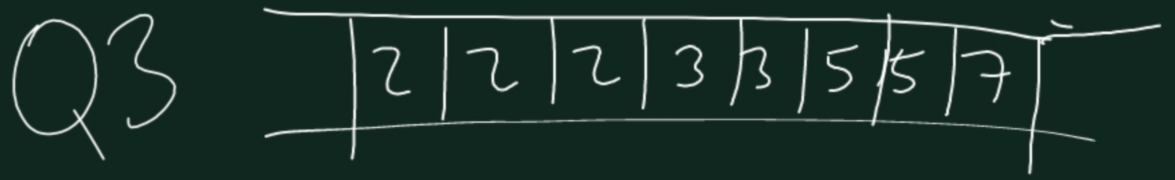
Ezeket gyakorlásként érdemes meggondolni!

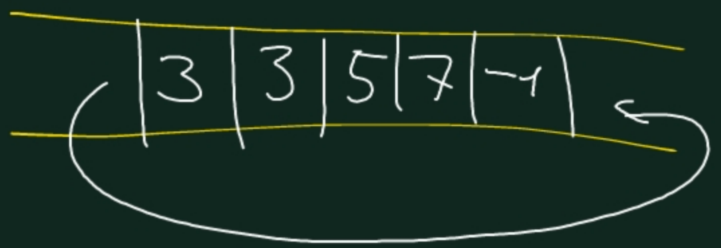
Q1, Q2 sorokban egy-egy egynél nagyobb szám prímtényezős felbontása található növekvő sorrendben. Készítsünk egy függvényt, ami egy új sorba előállítja a legkisebb közös többszörös prímtényezős felbontását. Q1 sor lebontható, Q2 maradjon meg!

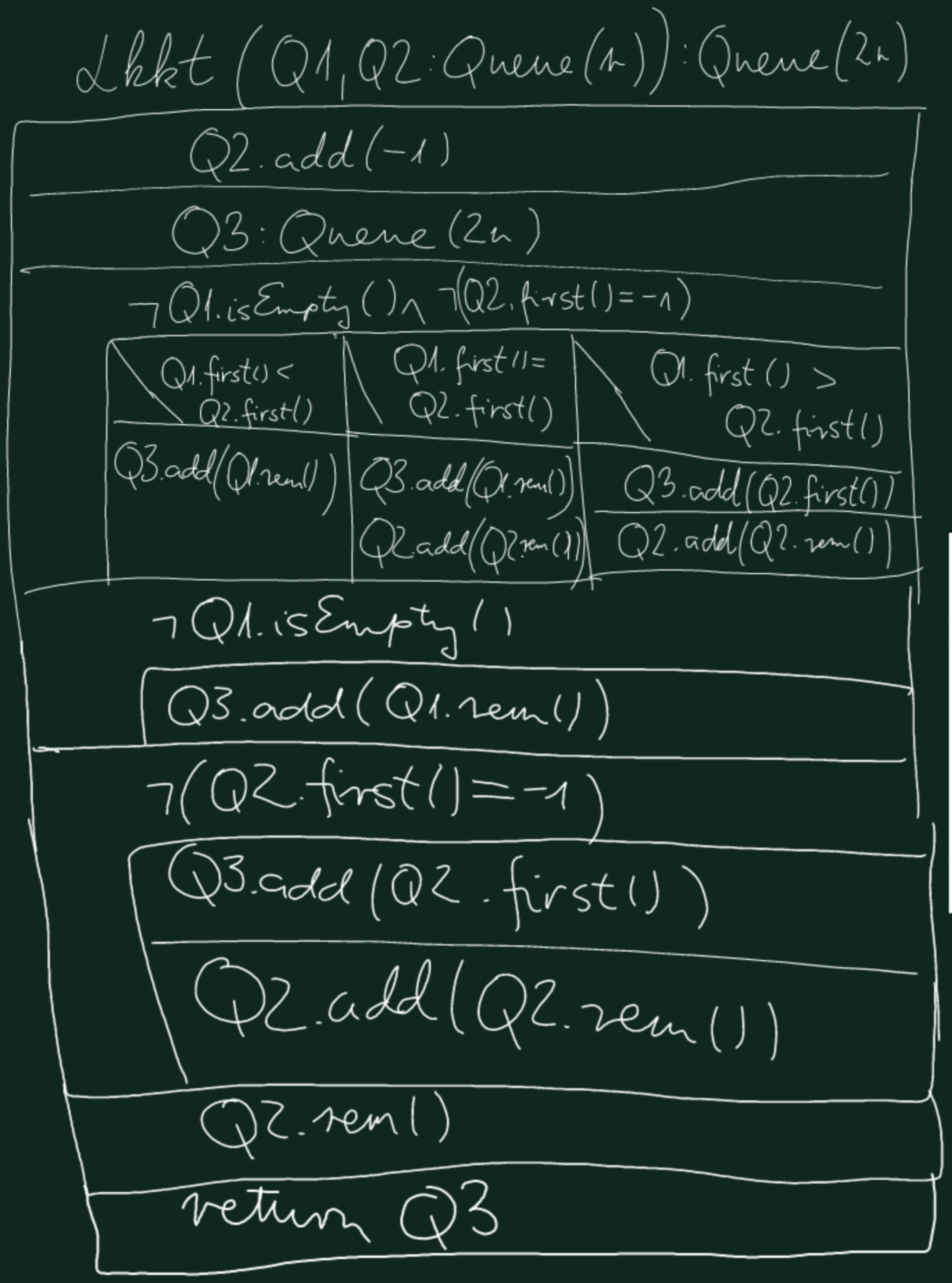
Trükk: Q2 végére szúrjunk egy ideiglenes „végjelet”, pl. -1-et, ezzel tudjuk vizsgálni, hogy hol van a sor vége.

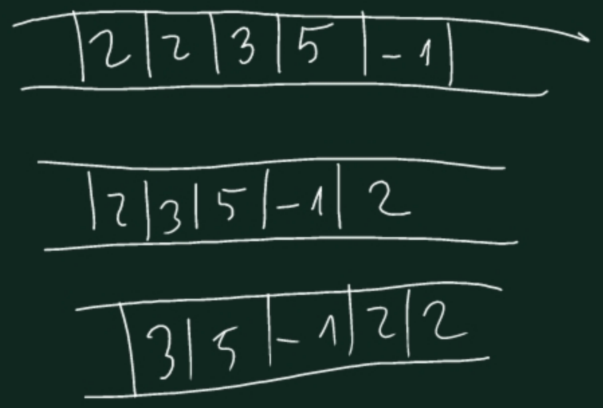










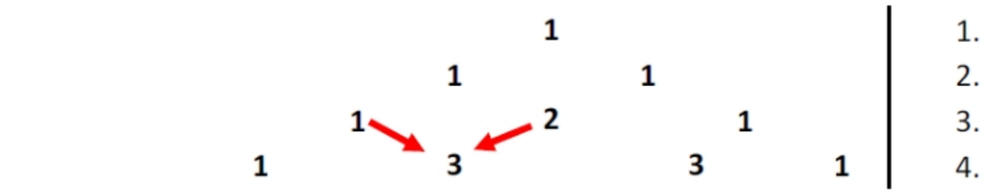


LKKT(Q1: Queue, Q2: Queue): Queue

Q3: Queue		
Q2.add(-1)		
¬ Q1.isEmpty() ∨ Q2.first() ≠ -1		
Q2.first() = -1 ∨ (¬ Q1.isEmpty() ∧ Q1.first() < Q2.first())	¬ Q1.isEmpty() ∧ Q2.first() ≠ -1 ∧ Q1.first() = Q2.first()	Q1.isEmpty() ∨ (Q2.first() ≠ -1 ∧ Q1.first() > Q2.first())
Q3.add(Q1.rem())	Q3.add(Q1.rem())	Q3.add(Q2.first()
	Q2.add(Q2.rem())	Q2.add(Q2.rem())
Q2.rem() *//-1 végjel eltüntetése*		
return Q3		

Szorgalmi Hf:

1 db sor és az összeadás művelet segítségével állítsuk elő a Pascal-háromszög k-adik sorát (feltehető, hogy $k \geq 1$)!



Láncolt listák → S1L, H1L → jegyzetben!

"Halmaz műveletek" S1L listában

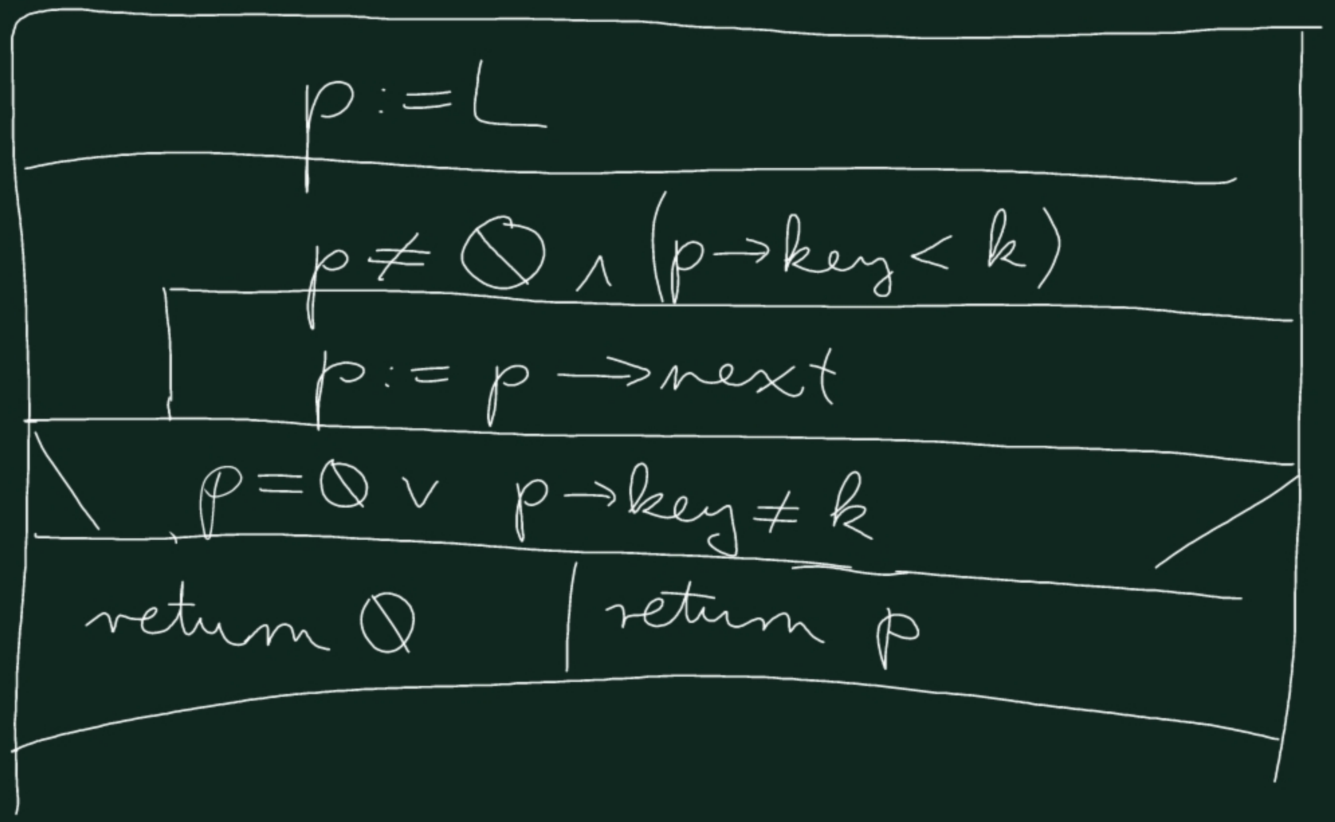
Feladat: adott L egy S1L lista, melyben egy halmazt "tárolunk" rendezetten (növ. sorrendben)

minden kulcs különböző

→ keresés
→ beszúrás (rendezetten)
→ törlés

} algoritmusát megírni!

KeresésSLLben (L : E1* ; k : T) : E1*

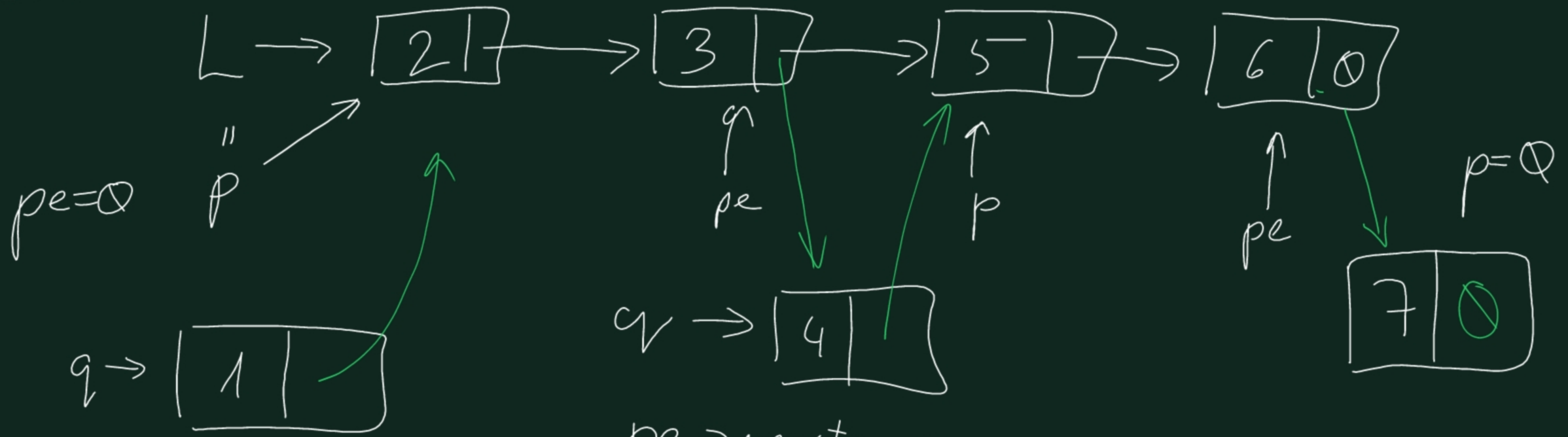


megtalált k kulcsú elem címe, ∅ ha nincs benne

→ és ha L nem rendezett?

→ és ha L HLL típusú

Beszúrás: 2 pointer p és pe



L:=q

q->next:=p

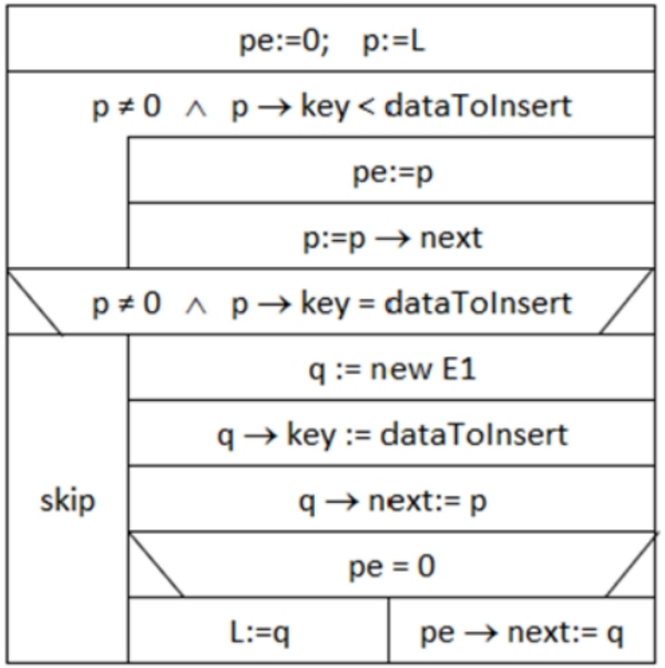
pe->next:=q

q->next:=p

Üres listára?

Beszúrás:

insertIntoS1L(&L: E1*, dataToInsert: T)



dataToInsert = k

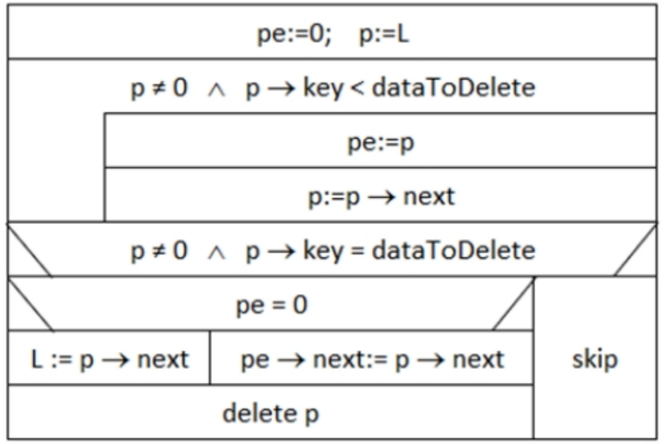
lista "közepére" vagy végére

ha már benne van

üres listába szúrom, vagy lista elejére

Törlés:

deleteFromS1L(&L:E1*, dataToDelete: T)



Szorg. HF.

Egy rendezetlen egyirányú fejelemes listában (H1L) keressük meg az (egyik) legnagyobb kulcsú elemet, egyszeri bejárással! Üres lista esetén a NIL pointert adja vissza az algoritmus, nem üres lista esetén az elemet fűzze ki a listából, és címét adja vissza az algoritmus.