5. előadás

A lista

Lista, verem, sor

Adatszerkezetek és algoritmusok előadás
2018. március 6.

Kósa Márk, Pánovics János és Szathmáry László
Debreceni Egyetem
Informatikai Kar

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet

Dinamikus, homogén, szekvenciális adatszerkezet.

Jelölések:

lista:	$q = [x_1, x_2, \ldots, x_n]$
üres lista:	$q = [\,]$
a lista feje:	x_1
a lista farka:	$[x_2, \ldots, x_n]$
a lista vége:	x_n
a lista hossza, mérete:	n vagy $\lvert q \rvert$

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet

Alapműveletek

- Hozzáférés, elérés: közvetlen.

$$q[i] = x_i$$

- Részlistaképzés, allistaképzés:

$$q[i \ldots j] = [x_i, x_{i+1}, \ldots, x_{j-1}, x_j]$$

- Konkatenáció, egyesítés, összefűzés:

$$r = [y_1, \ldots, y_m]$$
$$q \,\&\, r = [x_1, x_2, \ldots, x_n, y_1, \ldots, y_m]$$

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet

Listával végezhető műveletek

- Létrehozás: explicit módon felsoroljuk az elemeit.
- Bővítés: bárhol bővíthető. Bővítéskor részlistákat képzünk, majd azokat konkatenáljuk az új elemből/elemekből álló részlistával. A k-adik elem mögé történő bővítés:

$$q[1 \ldots k] \,\&\, [\text{elem}] \,\&\, q[(k+1) \ldots n]$$

- Törlés: megvalósítható a fizikai törlés, melynek során részlistákat képzünk (melyekben már nem szerepel(nek) a törlendő elem(ek)), majd konkatenáljuk ezeket a részlistákat. A k-adik elem törlése:

$$q[1 \ldots (k-1)] \,\&\, q[(k+1) \ldots n]$$

- Csere: bármelyik elem cserélhető. Részlistákat képzünk, majd azokat konkatenáljuk az új értékből álló részlistával. A k-adik elem cseréje:

$$q[1 \ldots (k-1)] \,\&\, [\text{elem}] \,\&\, q[(k+1) \ldots n]$$

- Rendezés: értelmezhető, bármelyik rendezési algoritmus használható.
- Keresés: értelmezhető, bármelyik keresési algoritmus használható.
- Elérés: soros vagy közvetlen.
- Bejárás: értelmezhető.
- Feldolgozás: a lista alapműveletei segítségével.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet

A szélső elemekkel végezhető speciális műveletek

- ACCESS HEAD: az első elem elérése:

$$q[1] = x_1$$

- PUSH: bővítés az első elem előtt:

$$[\text{elem}] \,\&\, q$$

- POP: az első elem törlése:

$$q[2 \ldots n]$$

- ACCESS END: az utolsó elem elérése:

$$q[n] = x_n$$

- INJECT: bővítés az utolsó elem után:

$$q \,\&\, [\text{elem}]$$

- EJECT: az utolsó elem törlése:

$$q[1 \ldots n-1]$$

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet reprezentációja

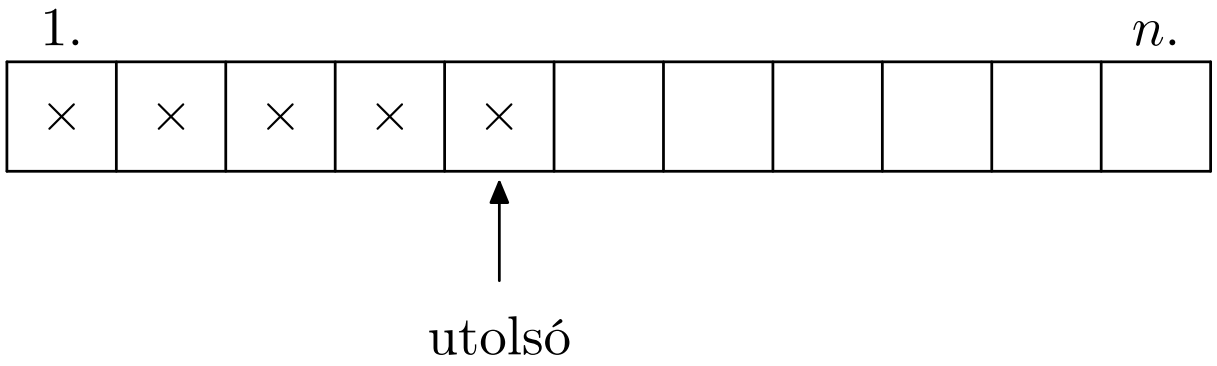
- Folytonos reprezentáció: vektorral.
- Szétszórt reprezentáció: láncolt listával.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet folytonos reprezentációja

Folytonos reprezentáció esetén a lista elemei egy tömbben vannak letárolva egymás után. A lista könnyen bejárható, illetve egy új elemet azonnal be lehet tenni a lista végére.



Egy új elem beszúrása a lista közepére viszont azzal jár, hogy az őt követő elemeket eggyel jobbra kell mozgatni. Hasonlóképpen, egy elem törlése (kivéve ha az utolsó elemet töröljük) szintén elemmozgatással jár.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet szétszórt reprezentációja

Láncolt listák esetén az egymás utáni elemeket valamilyen módon meg kell címezni. Erre valók a mutatók.

Mutató

A mutató (pointer) egy olyan változó, amelynek az értéke egy memóriacím. Magas szintű programozási nyelvekben ehhez hozzárendelődik még az a típus is, amelyre a mutató mutat.

Műveletek mutatókkal

- Mutató hozzárendelése egy objektumhoz:

```
tipus *p;
p = (tipus *)malloc(sizeof(tipus));
```

- A mutató nincs semmilyen objektumhoz sem rendelve:

```
p = NULL;
```


Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet szétszórt reprezentációja

Műveletek mutatókkal (folyt.)

- A mutató által címzett objektum megszüntetése:

```
free(p);
```

- A mutató átállítása egy másik mutató által címzett objektumra:

```
q = p;
```

1. Mi történik a `q`-val korábban címzett objektummal?
2. Legyen a `p` mutató egy lokális változó egy eljárásban. Mi történik az általa címzett objektummal?

- Értékadás mutatóval:

```
valtozo = *p;
*p = ertek;
```

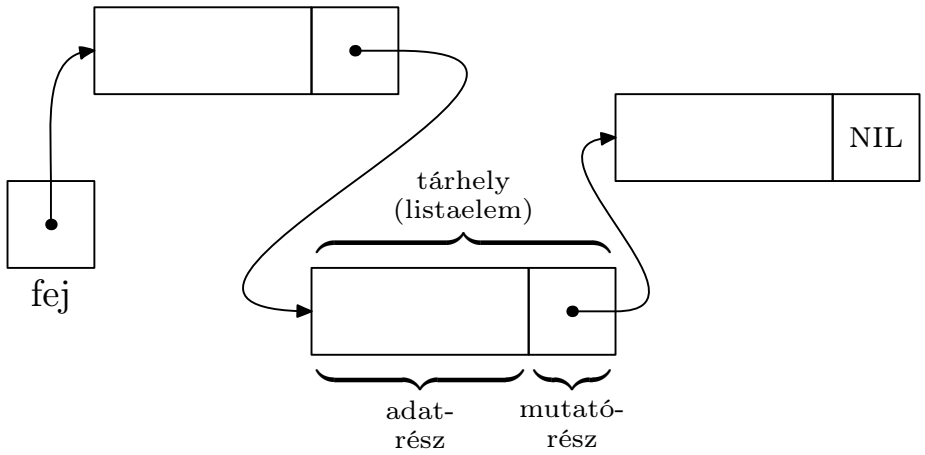
- Összehasonlítás:

```
if (p == q) ...     /* Ugyanoda mutatnak? */
if (*p == *q) ...   /* A mutatott objektumok
                       azonos ertekuek? */
```


Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

Egyszerű láncolt lista



A listaelem egy rekord. Egy üres láncolt listát a következőképpen tudunk létrehozni C-ben:

hosszabb változat:	rövidebb változat:

struct listaelem {
  típus adat;
  struct listaelem *kov;
};
typedef struct listaelem LISTAELEM;
LISTAELEM *fej;
fej = NULL;
	typedef struct listaelem {
	  típus adat;
	  struct listaelem *kov;
	} LISTAELEM;
	
	LISTAELEM *fej = NULL;

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet szétszórt reprezentációja

Műveletek egyszerű láncolt listákon

- Üres lánc inicializálása:

```
fej = NULL;
```

- Elem beszúrása:
  - lista elejére
  - lista végére
  - aktuális elem után
  - aktuális elem elé
- Elem törlése:
  - lista elejéről
  - lista végéről
  - aktuális elem
- Pozicionálás:
  - lista elejére
  - lista végére
  - aktuális elem után

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet szétszórt reprezentációja

Műveletek egyszerű láncolt listákon (folyt.)

- aktuális elem értékének kiolvasása
- jelezzük, ha a lánc üres
- jelezzük, ha az aktuális elem a lánc utolsó eleme

A továbbiakban nézzük meg néhány művelet implementációját. Az egyszerűség kedvéért a fejmutatót globális változóként kezeljük.

beszúrás a lánc elejére

```
void elejere_beszur(tipus adat)
{
   LISTAELEM *uj = (LISTAELEM *)malloc(sizeof(LISTAELEM));

   uj->adat = adat;
   uj->kov = fej;
   fej = uj;
}
```


Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet szétszórt reprezentációja

beszúrás a lánc végére

Ha alkalmazunk egy „utolsó” mutatót is (mely minden esetben a lánc utolsó elemére mutat), akkor könnyű dolgunk van. Ellenkező esetben meg kell keresni a lánc végét:

```
void vegere_beszur(tipus adat)
{
   LISTAELEM *akt = fej;

   LISTAELEM *uj = (LISTAELEM *)malloc(sizeof(LISTAELEM));
   uj->adat = adat;
   uj->kov = NULL;

   if (fej == NULL) {
      fej = uj;
   }
   else
   {
      while (akt->kov != NULL) {
         akt = akt->kov;
      }
      akt->kov = uj;
   }
}
```


Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet szétszórt reprezentációja

aktuális elem *után* való beszúrás

lásd ÁBRA

aktuális elem *elé* való beszúrás

- Két mutatót használunk: „előző” és „aktuális”. Az „előző” mutató az „aktuális” elem előtti elemre mutat. A beszúrás az „előző” elem után történik.
- Egy mutató használata esetén: beszúrás az aktuális elem után, majd adatrészek cseréje.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet szétszórt reprezentációja

Elem törlése

első elem törlése

lásd ÁBRA

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet szétszórt reprezentációja

utolsó elem törlése

Meg kell jegyezni, hogy melyik az utolsó előtti elem, s annak a következő mutatóját NIL-re állítjuk.

```
void utolso_torles()
{
   LISTAELEM *elozo = NULL;
   LISTAELEM *akt = fej;

   if (fej == NULL) {
      hiba("a lanc ures");
      return;
   }

   /* kulonben, ha a lanc nem ures */
   while (akt->kov != NULL) {
      elozo = akt;
      akt = akt->kov;
   }
   if (elozo == NULL) {  /* egyelemu */
      free(akt);
      fej = NULL;
   }
   else {   /* egynel tobb elemu */
      free(akt);
      elozo->kov = NULL;
   }
}
```


Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A lista adatszerkezet szétszórt reprezentációja

Keresés

adott elem keresése a láncban

```
int benne_van(LISTAELEM *fej, tipus keresett_ertek)
{
   LISTAELEM *akt = fej;

   while ((akt != NULL) && (akt->adat != keresett_ertek))
   {
      akt = akt->kov;
   }

   if (akt != NULL) return 1;
   else return 0;
}
```


Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A verem adatszerkezet

Speciális lista adatszerkezet. Csak a verem **tetejére** lehet betenni, illetve csak onnan lehet kivenni.

- Az utoljára betett elem a verem tetejére kerül.
- Az elsőnek betett elem a verem aljára kerül.

Veremmel végezhető műveletek

- Létrehozás: üres verem inicializálása.
- Bővítés: új elem bevitele az utoljára betett elem fölé (PUSH).
- Csere: nincs.
- Törlés: fizikai, a verem tetején lévő elemet (POP).
- Rendezés, keresés és bejárás: nem értelmezett.
- Elérés: a fölső elemet közvetlenül, a többit sehogyan sem (TOP).
- Feldolgozás: Last In First Out (LIFO) adatszerkezet, az utolsóként érkező elemet dolgozzuk fel először.

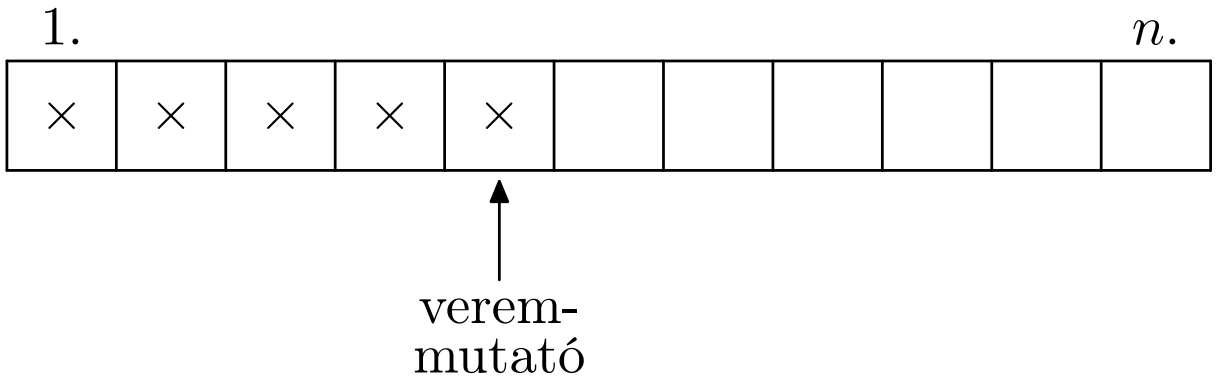
Az utolsóként érkezett elemhez történő hozzáférést illetve ezen elem törlésének a műveletét egy műveletként is definiálhatjuk.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

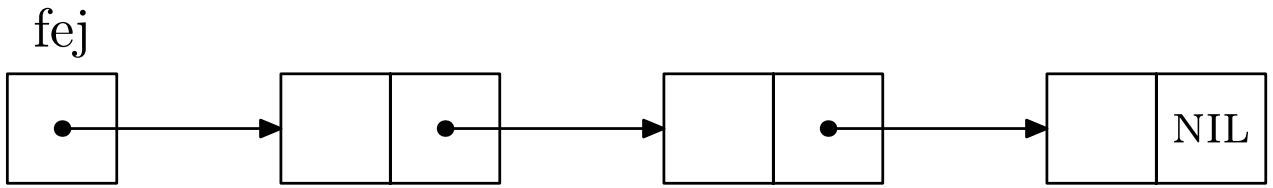
A verem adatszerkezet reprezentációi

Folytonos reprezentáció:



A veremmutató mindig a verem tetején lévő elemet indexeli. Ha a veremmutató értéke 0, a verem üres. Ha a veremmutató értéke *n*, a verem tele van.

Szétszórt reprezentáció: egyirányban láncolt listával.



A fej mutató mindig a verem tetején lévő elemre mutat. Ha a fejnek NIL az értéke, a verem üres.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A sor adatszerkezet

Speciális lista adatszerkezet, melynek alapműveletei a speciális listaműveletek közül a következők:

- az első elemhez történő hozzáférés (FRONT)
- bővítés az utolsó elem mögé (ENQUEUE)
- az első elem törlése (DEQUEUE)

Az első elemhez történő hozzáférés és az első elem törlésének műveletét egy műveletként is definiálhatjuk.

Sorral végezhető műveletek

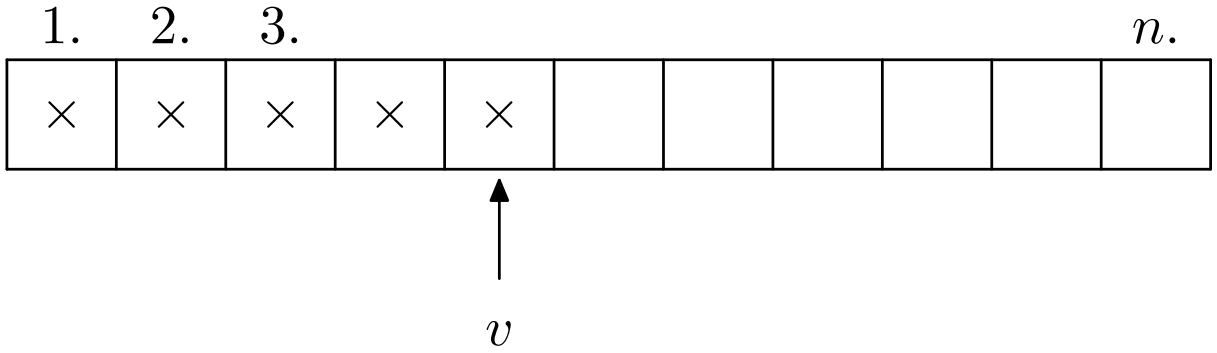
- Létrehozás: üres sor.
- Bővítés: az utolsó elem mögé.
- Csere: nincs.
- Törlés: fizikai, az első elemet.
- Rendezés, keresés és bejárás: nem értelmezett.
- Elérés: az első elemet közvetlenül, a többit sehogyan sem.
- Feldolgozás: First In First Out (FIFO) adatszerkezet, az elsőként érkező elemet dolgozzuk fel először.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A sor adatszerkezet folytonos reprezentációi

Fix kezdetű sor:



A sor első elemének a helye rögzített, mindig az 1. indexű tárhely a vektorban. A v (vége) mutató a sor utolsó elemét indexeli. Üres a sor, ha $v = 0$. Tele van a sor, ha $v = n$.

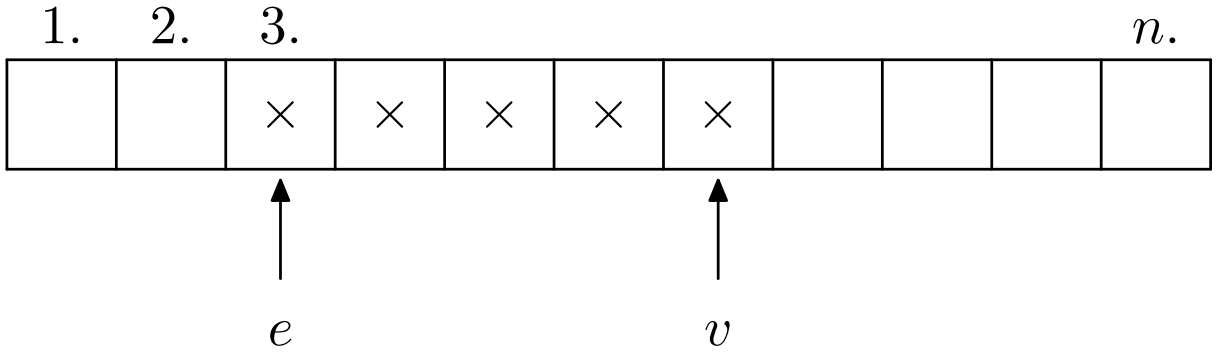
Az új elemet a $(v + 1)$-edik pozícióra helyezzük el, ha a sor nincs tele. Törölni az 1. pozíción lévő elemet tudjuk, ha a sor nem üres. Törléskor a sor megmaradó elemeit egy tárhellyel előrébb csúsztatjuk.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A sor adatszerkezet folytonos reprezentációi

Vándorló sor:



A sor első elemét az *e* (eleje) mutató, az utolsót a *v* (vége) mutató indexeli. Üres a sor, ha $e = 0$ és $v = 0$. Tele van a sor, ha $e = 1$ és $v = n$.

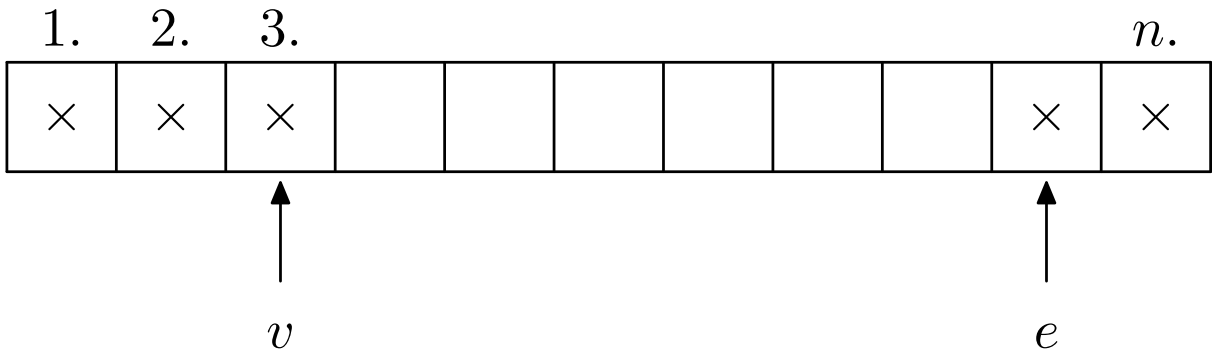
Ha bővítéskor $v = n$, de a sor nincs tele, akkor először a sor minden elemét $e - 1$ pozícióval előrébb csúsztatjuk, majd végrehajtjuk a bővítést az új elemmel. Az új elemet a $(v + 1)$-edik pozícióra helyezzük el, ha a sor nincs tele. Törölni az *e*-edik pozíción lévő elemet tudjuk, ha a sor nem üres.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A sor adatszerkezet folytonos reprezentációi

Ciklikus sor:



A sor első elemét az e (eleje) mutató, az utolsót a v (vége) mutató indexeli. Üres a sor, ha $e = 0$ és $v = 0$. Tele van a sor, ha $e = v \bmod n + 1$.

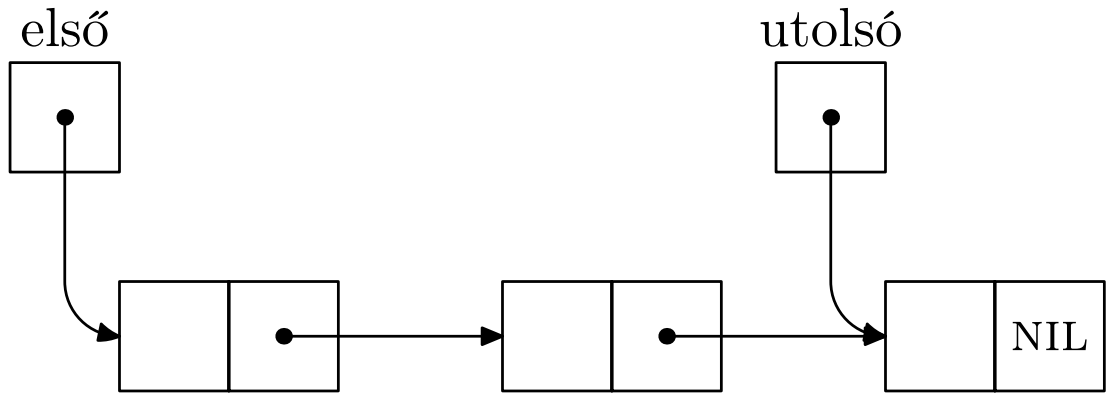
Az új elemet a $(v \bmod n + 1)$-edik pozícióra helyezzük el, ha a sor nincs tele. Törölni az e-edik pozíción lévő elemet tudjuk, ha a sor nem üres.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

A sor adatszerkezet szétszórt reprezentációja

Szétszórt reprezentáció egyirányban láncolt listával, két segédmutatóval:



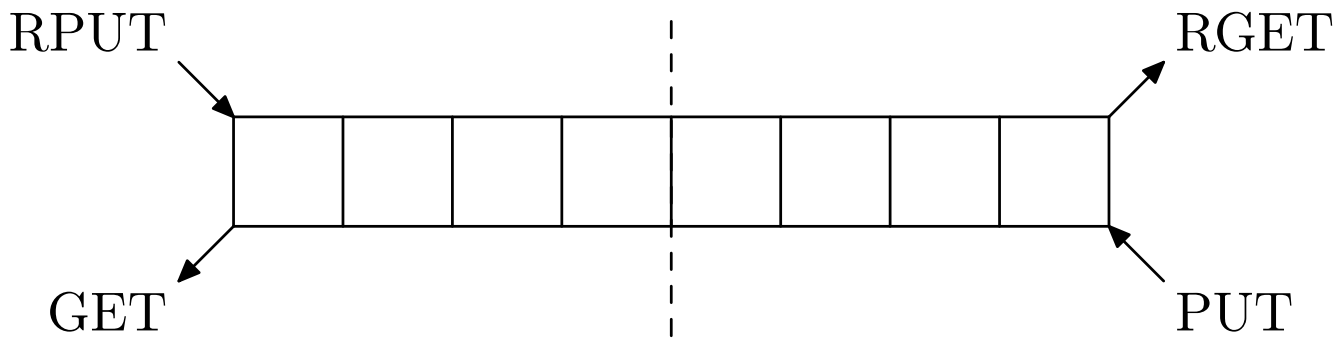
Elérni és feldolgozni az „első” mutató által hivatkozott elemet tudjuk, bővíteni pedig az „utolsó” mutató által hivatkozott elem mögé tudunk. A sor üres, ha mindkét segédmutató értéke NIL.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

Kétvégű sorok

- Kétvégű sor: a sor műveletei mellett további két művelet jelenik meg: RGET és RPUT (reverse GET és reverse PUT).



  Tekinthető két, az aljuknál összeragasztott veremnek.

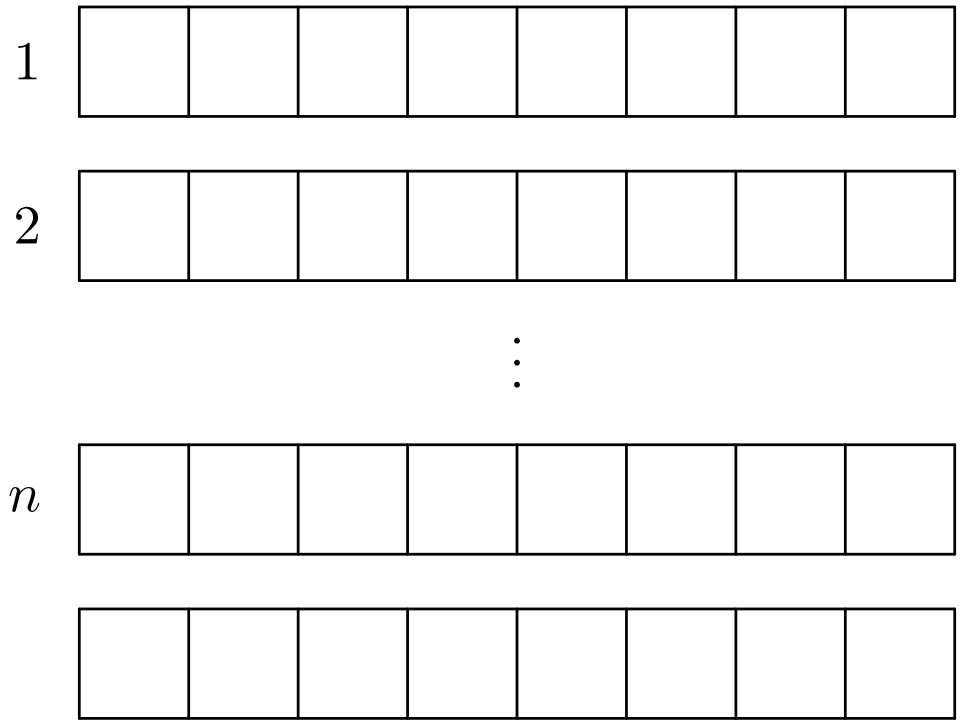
- Inputkorlátozott kétvégű sor: nincs RPUT művelet.
- Outputkorlátozott kétvégű sor: nincs RGET művelet.

Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor

Prioritásos sor

Olyan sor, amelyben az adatelemekhez prioritásértéket rendelünk, majd ezen értékek sorrendjében (azonos prioritású elemek esetén pedig továbbra is a bekerülés sorrendjében) dolgozzuk fel őket. Megvalósítása n különböző prioritásérték esetén $n+1$ (hagyományos) sorral történhet: minden prioritásértékhez tartozik egy-egy sor, a prioritás nélküli elemeket pedig külön sorban tároljuk:



Kósa Márk
Pánovics János
Szathmáry László

Lista
folytonos reprezentáció
szétszórt reprezentáció
Verem
Sor
Kétvégű sorok
Prioritásos sor