



Belépő a tudás közösségébe

Informatika szakköri segédanyag



Sor típus

Menyhárt László

Szerkesztő: Abonyi-Tóth Andor, Zsakó László

A kiadvány „A felsőoktatásba bekerülést elősegítő készle-
szítő és kommunikációs programok megvalósítása, val-
az MTMI szakok népszerűsítése a felsőoktatásban” (EFOP-
16-2017-006) című pályázat keretében készült 2018-ban.



Eötvös Loránd Tudományegyetem
Informatikai Kar

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

Ebben az anyagrészben egy speciális sorozat típusról olvashatunk.

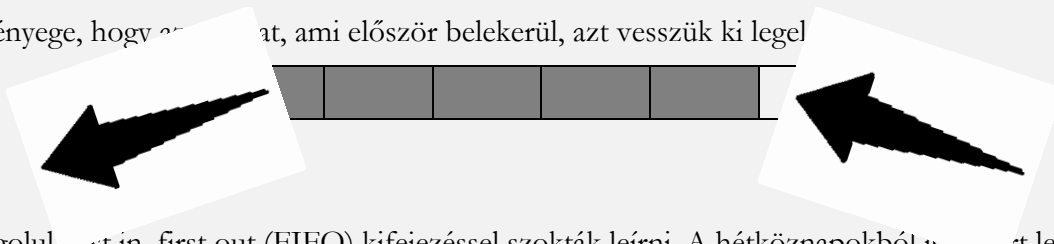
Ez az adatszerkezet nem csak bizonyos elemek tárolására lesz alkalmas, hanem műveletek is tartoznak hozzá, melyekkel lehet az elemeket elérni és így speciális logikát alkalmazhatunk rajta.

Az értékhalmaz: az alaphalmaz (amely az *Elemtípus* által van meghatározva) iteráltja („mely elemek lehetnek benne a halmazban”). Az *Elemtípus* általában valamely véges diszkrét típus lehet, legtöbbször még az elemszámát is korlátozzák (≤ 256). Ha nyelvi elemként nem létezik, akkor a megvalósításunkban lehet nagyobb elemszámú is.

Tegyük fel, hogy a probléma megismerése és a megoldás tervezése közben már tudjuk, hogy maximum mennyi adatot szeretnénk tárolni, így az egyszerűség kedvéért egy olyan statikus implementációt mutatunk be, ami tömböt használ az adatok tárolására.

Sor

A lényege, hogy az adat, ami először belekerül, azt vesszük ki legelőször.



Angolul *first in, first out* (FIFO) kifejezéssel szokták leírni. A hétköznapiakból is ismert lehet, amikor az emberek sorban állnak, egymás után érkeznek, majd lassan mindenkivel foglalkoznak és kikerülnek a sorból. Ugyanígy vannak speciális feladatok, amikor az adatokat is hasonlóképpen kell feldolgozni.

Műveletek

Üres (üres sor létrehozás eljárás), *Üres?* (logikai értékű függvény), *Sorba* (elem hozzávétele egy sorhoz), *Sorból* (elem kivétele egy sorból), *Első* (elem megtekintése a sor elejéről).

A műveletek tulajdonságai közül a következők a legfontosabbak. Az üres sorra a sorba rakás és az üresség vizsgálat kivételével minden művelet értelmetlen. A sorból kivétel a sorba legrégebben tett elemet veszi ki, azaz a sor alkalmas egy sorozat feldolgozására, ha a keletkezés és a feldolgozás üteme eltér egymástól. A sor elején mindig a sorba legrégebben betett (legelől lévő) elem látszik.

```
Sor (Elemtípus) = Rekord (elem: Tömb (1..MaxDb:Elemtípus); első, utolsó, db: Egész)
```

MaxDb elemszámú Elemtípus típusú elemet tárolunk egy tömbben. Két segédváltozónk mutatja, hogy melyik után kell felvenni új elemet, illetve melyik az első, amit ki lehet venni a sorból. Köztük az elemek folytonosan szerepelnek.

Ebben az esetben az alábbi módon valósíthatjuk meg a műveleteket:

```
Eljárás Üres (Változó s: Sor (Elemtípus)) :  
    s.utolsó:=0  
    s.elő:=1  
    s.db:=0;  
Eljárás vége.
```

Műveletigény számítása: nem függ a **sor elemszámától**.

```
Függvény Üres?(Konstans s: Sor(Elemtípus)): Logikai
    Üres? := (s.db=0)
```

Függvény vége.

Műveletigény számítása: nem függ a **sor elemszámától**.

```
Eljárás Sorba(Változó s: Sor(Elemtípus),
              Konstans e: elemtípus):
    s.utolsó:=s.utolsó+1; Ha s.utolsó>MaxDb akkor s.utolsó:=1;
    s.db:=s.db+1; s.elem[s.utolsó]:=e
```

Eljárás vége.

Műveletigény számítása: nem függ a **sor elemszámától**.

```
Függvény Sorból(Változó s: Sor(Elemtípus): Elemtípus
    e:=s.elem[s.első]
    s.első:=s.első+1; Ha s.első>MaxDb akkor s.első:=1;
    s.db:=s.db-1;
    Sorból:=e
```

Eljárás vége.

Műveletigény számítása: nem függ a **sor elemszámától**.

```
Függvény Első(Konstans s: Sor(Elemtípus)): Elemtípus
    Első:=s.elem[s.első]
```

Függvény vége.

Műveletigény számítása: nem függ a **sor elemszámától**.

Minta kódok.

C++	cpp/feladat.cpp
C#	cs/feladat.cs
Java	java/feladat.java
Pascal	pas/feladat.pas
Python	py/feladat.py



A megoldás alapvető problémája, hogy MaxDb-t előre kell ismernünk és természetesen bele kell férnie a memóriába.

Az így ábrázolt sorban lévő elemek típusára semmilyen megkötést nem kell tennünk, hiszen egy tömbben bármilyen elem elhelyezhető.

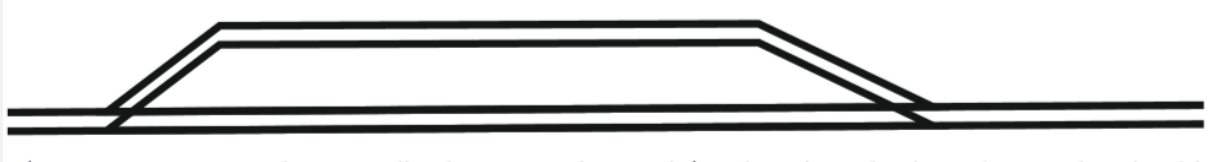
A fenti algoritmusban tehát a MaxDb nem a feladathoz szükséges összes adat darabszáma kell, hogy legyen, hanem az egyszerre tárolandó elemek darabszáma. Ellenőrizzük, hogy amikor az egyik index, azaz segédváltozó eléri a MaxDb értéket, akkor ciklikus permutáció mintájára ugorjon vissza az első indexre. Így előfordulhat, hogy az első elem a MaxDb-edik és az utolsó (például a második) elem az első lesz.

Feladatok sorra

a Nemes Tihamér NITV-ről és az Informatika OKTV-ről

1. feladat

Egy vasútállomáson a fő vágányról kitérő vágányt építettek, mindegyiken csak jobbról balra lehet haladni. A kitérőn akárhány kocsit elhelyezhetünk, a fő vágányon azonban nem állhat meg kocsi. Egy szerelvény érkezik jobbról, sorszámozott, összekevert kocsikkal. A kocsikat 1 és K közötti különböző sorszámokkal azonosítjuk.



Készíts programot, amely megadja, hogy maximum hány kocsit tudunk sorba rendezni a kitérő segítségével és elküldeni balra úgy, hogy a kitérőn végül nem marad kocsi!

A standard bemenet első sorában a kocsik száma van ($1 \leq K \leq 100\,000$). A következő K sorban a kocsik sorszámai következnek az érkező szerelvényen belüli sorrendben ($1 \leq S_i \leq K$).

A standard kimenet egyetlen sorába azt kell kiírni, hogy maximum mennyi kocsit lehet így sorba rendezni!

Példa:

Bemenet

6
4
5
2
6
3
1

Kimenet

5

Magyarázat: a kocsik 2,3,4,5,6 sorrendben mehetnek ki balra.

Megoldas:

```

Olvas(k); ut:=0; jo:=igaz; Ures(s); i:=1
Ciklus amig i≤k és jo
  Olvas(szam)
  Ha szam<út akkor jo:=hamis
  különben ha Üres?(s)
    akkor sorba(s,szam)
  különben ha szam>Utolsó(s)
    akkor sorba(s,szam)
  különben Ciklus amig szam>Első(s)
    Sorból(s,ut)
  Ciklus vége
  ut:=szam
Elágazás vége
  i:=i+1
Ciklus vége
Ha jo akkor Kiír(k) különben Kiír(i-2)
Eljárás vége.

```

Néha előfordul, mint ahogy itt is, hogy nem csak az első, hanem az utolsó elemet is szeretnénk megtekinteni. Ennek algoritmusáértélemszerűen az első mintájára készíthető el.

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Sorozatok
Feladat	77. Vasúti kocsik mellékvágányokon

2. feladat

Egy 1 processzoros számítógép memóriájában programok vannak, mindegyik K időegységig használná a processzort. Az egyes programok a processzor idejéből 1-1 időegységet kapnak a memóriába kerülésük sorrendjében, az utolsónak bekerült után újra az első jön. Ha a program futási ideje lejárt, akkor kikerül a memóriából. Közben újabb programok kerülhetnek be a memóriába, az i . időpontban memóriába került program a futásra várók közül az utolsó helyre kerül!

Készíts programot, amely megadja, hogy a processzor időegységeit milyen sorrendben használják az egyes programok!

A standard bemenet első sorában a programok száma ($1 \leq N \leq 50\,000$) és az egyes programok futási ideje ($1 \leq F \leq 100$) van. A következő sorban az egyes programok kezdési ideje szerepel ($1 \leq Kezd_i \leq 1\,000\,000$), növekvő sorrendben.

A standard kimenet első sor i . száma az i . program befejeződési ideje legyen!

Példa:

Bemenet

4 3
2 3 3 7

Kimenet

6 10 11 13

Magyarázat: a processzor idejét a programok a következőképpen használják:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
-	1	1	2	3	1	2	3	4	2	3	4	4	-

Megoldas:

```

Olvas(n,k)
Ciklus i:=1-től n-ig
    Olvas(kezd[i])
Ciklus vége
ido:=kezd[1]
i:=1; se:=1; su:=0; sdb:=0
Ciklus amíg kezd[i]=ido
    Sorba(k); i:=i+1;
Ciklus vége
Ciklus amíg sdb>0 vagy i≤n
    Ha sdb>0 akkor
        Ha sor[se]=1 akkor Kiír(ido,' '); Sorbol(j);
        különben Sorbol(j); Sorba(j-1);
    Elágazás vége
    ido:=ido+1
    Ciklus amíg i≤n és kezd[i]=ido
        Sorba(k); i:=i+1
    Ciklus vége
Ciklus vége

```

Eljárás vége.

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Szimuláció
Feladat	25. Ütemezés azonos futási idővel

3. feladat

Egy 1 processzoros számítógép memóriájában programok vannak, mindegyik adott időegységig használná a processzort. Az egyes programok a processzor idejéből 1-1 időegységet kapnak a memóriába kerülésük sorrendjében, az utolsónak bekerült után újra az első jön. Ha a program futási ideje lejárt, akkor kikerül a memóriából. Közben újabb programok kerülhetnek be a memóriába, az i . időpontban memóriába került program a futásra várók közül az utolsó helyre kerül!

Készíts programot, amely megadja, hogy a processzor időegységeit milyen sorrendben használják az egyes programok!

A standard bemenet első sorában a programok száma van ($1 \leq N \leq 100\,000$). A következő N sorban az egyes programok kezdési ideje ($1 \leq \text{Kezd}_i \leq 1\,000\,000$) és a futásához szükséges processzoridő ($1 \leq P_i \leq 1000$) szerepel kezdés szerint növekvő sorrendben.

A standard kimenet első sor i . száma az i . program befejeződési ideje legyen!

Példa:

Bemenet

4
2 3
3 3
3 2
7 4

Kimenet

6 10 8 13

Magyarázat: a processzor idejét a programok a következőképpen használják:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
-	1	1	2	3	1	2	3	4	2	4	4	4	-

Megoldas:

```

Olvas(n)
Ciklus i:=1-től n-ig
    Olvas(kezd[i],hossz[i])
Ciklus vége
ido:=kezd[1]; i:=1; se:=1; su:=0; sdb:=0
Ciklus amíg kezd[i]=ido
    Sorba(i,hossz[i]); i:=i+1
Ciklus vége
Ciklus amíg sdb>0 vagy i≤n
    Ha sdb>0 akkor
        Ha sor[se,2]=1 akkor Sorbol(j,k); veg[j]:=ido
        különben Sorbol(j,k); Sorba(j,k-1)
    Elágazás vége
    ido:=ido+1
    Ciklus amíg i≤n és kezd[i]=ido
        Sorba(i,hossz[i]); i:=i+1
    Ciklus vége
Ciklus vége
Ciklus i:=1-től n-1-ig
    Kiír(veg[i], ' ')
Ciklus vége
    Kiír(veg[n])

```

Eljárás vége.

A feladat megoldása tesztelhető az elkészült forráskód feltöltésével itt:

Weboldal	https://mester.inf.elte.hu/
Szint	Haladó
Téma	Szimuláció
Feladat	27. Ütemezés különböző futási idővel