F: Az állapotter (n x m-es mátrix) elemeiről értése 1 v. 0. Hat megy az i. sor j- dik pozíciójan lévő elem 1-es szomszédainak a számát!

```
eljárás szomszédszám_1 (i, j, n, m, s: egész, t: tér)
  s := 0
  ciklus k := i-1 .. i+1
    ciklus l := j-1 .. j+1
      ha (k ≥ 1) és (k ≤ n) és (l ≥ 1) és (l ≤ m) és
                 megengedett sorpozíción van
        (k ≠ i) vagy (l ≠ j) akkor
              ne maga a vizsgált cella legyen
        ha t[k, l] = 1 akkor s := s+1;
    c vége;
  c vége;
e vége;
```

```
eljárás szomszédszám_2 (i, j, n, m, s: egész; t: tér);
  s := 0;
  ciklus k := i-1 .. i+1
    ciklus l := j-1 .. j+1;
      ha (k ≥ 1) és (k ≤ n) és (l ≥ 1) és (l ≤ m) és
        (k ≠ i) vagy (l ≠ j) akkor s := s + t[k, l];
    c vége;
  c vége;
e vége;
```

```
eljárás szomszédszám_3 (i,j, u,m, s: egész; t: tér);
  s := -t[i;j];
  ciklus r := i-1 .. i+1
    ciklus c := j-1 .. j+1
      ha (r ≥ 1) és (r ≤ u) és (c ≥ 1) és (c ≤ m) akkor
        s := s + t[r;c];
    c vége;
  c vége;
e vége;
```

→ ha az érték 0, nem változtatja semmit 0-is marad, a sorban lévő érték 0 ha az érték 1, -1-ben belőle, és az 1-ből 0-t csinál

	j		
	0	0	0
i	0	1	0
	0	0	0

-1+0+0+1.. +1+.. → 0 lesz az 1 értéke

Elhagyjuk azt a feltételt, amely megvizsgálja, h. az elem sorban lévő elem-e.

kivételesek az 1, és utolsó sor ill oszlopok elemei:

0	0	0	0	0	0	0
0	x	x	x	x	x	0
0	x	x	x	x	x	0
0	x	x	x	x	x	0
0	0	0	0	0	0	0

→ élettér

→ ennek az elemnek is van 8 szomszédja ⇒ Elhagyhatjuk a feltételvizsgálatot.

```
eljárás szomszédszám_4 (i,j, u,m, s: egész, t: tér);
  s := -t[i,j];
  ciklus r := i-1 .. i+1
    ciklus c := j-1 .. j+1
      s := s + t[r,c];
    c vége;
  c vége;
e vége;
```

Végrehajtási idő csökkentése – ciklusmag végrehajtási idejének csökkentése

Adatok elő-feldolgozása

A sokszor felhasznált adatokat nem akkor számítjuk ki, amikor szükség van rájuk, hanem még a feldolgozás megkezdése előtt.

F: Adott n x m-es mátrix sorait rendezzük a sorösszeg alapján növekvő sorrendbe!

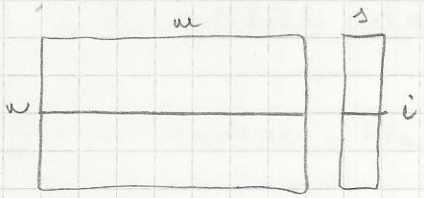
```
eljárás rendezés_1 (n, m : egész ; t : mátrix);
  s := Φ;
  ciklus i := 1 .. n-1;
    l := i;
    ciklus j := i+1 .. n
      sl := az l. sor elemeinek összege;
      sj := a j. sor elemeinek összege;
      ha sl > sj akkor l := j;
    ciklus vége;
    csere (i, l, t); => felcseréli a t mátrix i. és l. sorát
  ciklus vége;
e vége;
```

Hf: összegzés kiküszöbölését megoldani

n elemű sorozat esetében $\frac{n \cdot (n-1)}{2}$ összehasonlítást végzünk.

Mivel 2-nek hányja végre a ciklusmag a feltételvizsgálatot (i és l sora) $\Rightarrow$ összehas. száma: $\frac{2 \cdot n(n-1)}{2} = n \cdot (n-1)$

Előtte végezzük el az összegzést!



ciklus $i := 1..n$

$s[i] := \emptyset$;

ciklus $j := 1..m$

$s[i] := s[i] + t[i,j]$;

ciklus vége;

ciklus vége;

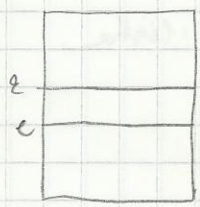
sc és sj sort elhasonlítjuk → első sor: ha a $s[c] > s[j]$ ...

F: Főátló feletti elemek alapján rendezzük az egész mátrixot!

7. előadás

11.24.

Adatmozgatások számának csökkentése:



sorok mozgatása: helyben rendezés

beszúrás elé v. mögé: beszúró rendezés

rendezhetjük min- és max. kiválasztás alapján.

Legalább 2 elemhez hozzá kell nyúlni mivel így sokat számolunk ⇒ szükség van „elő-feldolgozásra".

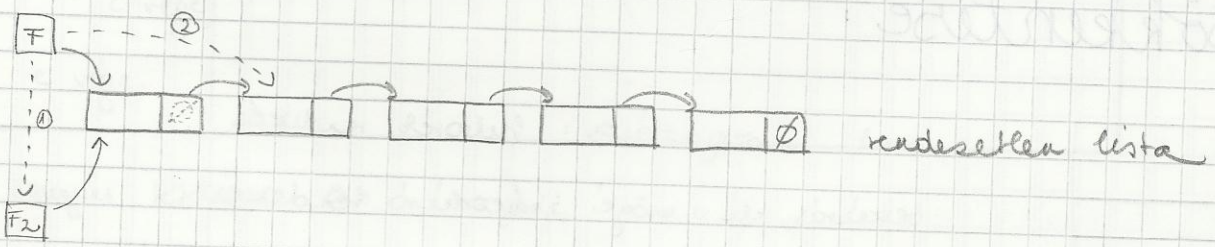
Sok adatmozgatással járó algoritmusok:

- rendezés
- kiválogatás
- nitválogatás

nagy mennyiségű adat	kulcs	hely
:		

F: A n rekordból álló sorozatot rendezzük a „kulcs" szerint növekvő sorrendbe.

- cserélgetéssel lehet, de túl sok adatot kell hozzá mozgatni.
- Elég: ha az első elemből kiindulva a MEGFELELŐ SORRENDBEN írjuk ki az adatokat.
- A táblázatot a „hely" oszloppal kiegészítjük, amelynek i-dik eleme mutatja, hogy hol az i. rekordot valójában hol kell keresni ⇒ többlet tárigény (így végülis listát kapunk.)



- $F_2 := F$ értékével
- F a második elemre mutat
- A első elem mutatójába végjelet írunk.
- Csak a mutatóknak kell értéket adni.

· újraírtuk a mutatókat, ezután F-nek F_2-t kell értékül adni.

```
Eljárás rendezés_Min (n: egész; v: sorozat);
  ciklus i:=1..n
    v[i].hely:=i
  c vége
  ciklus i:=1..n-1          t:=i;
  ciklus j:=i+1..n
    ha v[v[t].hely].kulcs < v[v[j].hely].kulcs akkor t:=j;
  c vége;
  csere (v[t].hely, v[i].hely);
  c vége;
e vége;
```

- A „hely" mező nem járjuk végig a sorozat elemeit.

v. → vektor

valahányadik sor adatmezőjére vagyunk kíváncsiak

v[v[1].hely].adat

1. sor hely mezője ⇒ itt a legkisebb

```
ciklus i:=1..n
ki: v[v[i].hely].adat
```

↓ i. sor hely mezőjén eltárolt adatot írja ki.

- Rendezés beállítjuk a hely mezőt. Ha a sorszámok megegyeznek, nem kell változtatni.

Adatmozgatások számának csökkentése:

Mutatók használata

A min. és max. kiválasztásnál a maximális/minimális kulcsú elemnek elegendő a helyét megjegyezni.

Felesleges műveletek kiküszöbölése:

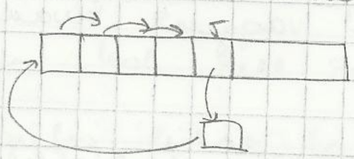
- Közvetlen kiválasztásos rendezés → minimum-kiválasztásos rendezés $\frac{2\cdot(n-1)}{2}$ míg másutt $3(n-1)$

Beszúró rendezés:

így ez 12 csere.

rendezett sorozat

- megnézzük a másodikat is, h. az elsővel egy rendezett sorozatot adjon
- azt a bizonyos elemet kimásoljuk egy átmeneti változóba → onnan hasonlítottuk össze.



6 értékadás

- k esetben $3(k-1)$ értékadást kell végezni
- csúsztatás esetén: $k-1+1+1 = k+1$

$3(k-1) = k+1$

$k=2$ → itt egyezik meg az értékadások száma

beszúró rendezés cserével → beszúró rendezés csúsztatással

buborékrendezés javításai

Alapelvében más megoldások keresése:

Matematikai ismeretek alkalmazása

1, . . . n-1

számtani forma, ha léteznek, alkalmazhatjuk az Összegzés tétele helyett. NEM ALKALMAZZUK AZ ÖSSZEGZÉS TÉTELÉT

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

N alatt k := fakt(n) / (fakt(k) * fakt(n-k));

probléma: vannak olyan részletek, amik kilsordulhatnak, de a hányados értéke nem olyan nagy. (az nem csordul túl)

$$\frac{\overbrace{n(n-1)(n-2)\dots(k+1)k(k-1)\dots 2\cdot 1}^{n \text{ db}}}{\underbrace{k(k-1)\dots 2\cdot 1}_{k \text{ db}} \cdot \underbrace{(n-k)(n-k-1)\dots 2\cdot 1}_{(n-k) \text{ db}}} \quad \begin{matrix} \to n \text{ db} \\ \to n-k+k = n \text{ db} \end{matrix} \Rightarrow$$

$$\Rightarrow \frac{n\cdot(n-1)\dots(k+1)}{(n-k)(n-k-1)\dots 2\cdot 1} \quad \begin{matrix} \to (n-k) \text{ szorzótényező} \\ \to (n-k) \text{ szorzótényező} \end{matrix}$$

„Megspóroltunk" 2k szorzást.

Alakítsuk át hányadosok szorzatává: $\left(\frac{ac}{bd} = \frac{a}{b}\cdot\frac{c}{d}\right)$

$$\frac{n}{n-k}\cdot\frac{n-1}{n-k-1}\cdot \dots \cdot\frac{k+2}{2}\cdot\frac{k+1}{1}$$

Így a viszonylag nagy számok kiszámolásától megmenekülünk (és hányadosokat szorzunk össze), de van 1 probléma.

$n, k \in \mathbb{Z} \Rightarrow \binom{n}{k} \in \mathbb{Z}$, de nehéz garantálni,

h. pl.: $\left(\frac{n}{n-k}\right) \in \mathbb{Z}$

Csökkentjük az elvégzendő szorzások (multiplikatív) műveletek számát, de kitolhatjuk a határt. (nagyobb számokkal lehet számolni, és gyorsabb is lesz.)

Tárigény, helyfoglalás csökkentése:

Neumann: ne csak az adatot tegyenek a memóriában, hanem a kód (processor működéséhez szükséges dolgot)

Adatot + Kód tárigénye

adatot tárigényének csökkentése:

(A végrehajtási időt a ciklusok módosításával változtathatjuk)

- Ott tudunk eredményt elérni, ahol kevesebb adatszerkezettel dolgozunk.
- Sorozatot elemszámát kell csökkenteni, v. megnövelni a sorozatot (ne tároljuk el őket)

```
F:=1;
ciklus i:=1..n
    f:=f*i;
c vége;
```

Nem kell letárolni n! elemeit

Fibonacci számok:

$$f_n := \begin{cases} n & , \text{ ha } n \leq 1 \\ f_{n-1} + f_{n-2} & , \text{ különben} \end{cases}$$

A f sorozat 0. eleme = 0, 1. eleme = 1.

```
f[0]:=0;
f[1]:=1;
ciklus i:=2..n
    f[i]:=f[i-1]+f[i-2]
c vége;
```

Mindig 2 elemre van szükség.

```
f0 := 0;        f1 := 1;
i := 2;
ciklus  i := 1 .. n
    fi := f0 + f1
    f0 := f1;
    f1 := fi;
ciklus vége
```

Egyszerre max. 3 elemet tárolunk el.

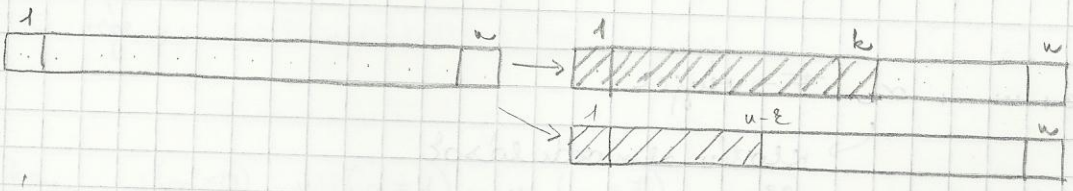
$$f_n = \frac{1}{\sqrt{5}} \cdot \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right]$$

Közvetlenül számolható a Fib. sorozat n-dik eleme.

Probléma: Valamely szempont szerint válogassuk szét a sorozat elemeit. (Alkossunk 2 részsorozatot.)

feltétel: 2 helyen nem lehet 1 elem egyidőben ⇒

⇒ szétválogatás



Így ennek a tárigénye $3n$

Azt kell elérni, hogy: $1-k$. helyen a $+$ tulajdonságúak } egyenes tárolva

$k-n$. -"- a nem -"-

Ez a szétválogatás helyben.