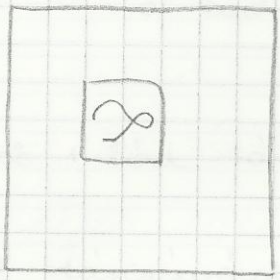
III. 31. 8. előadás

Adva van egy vonalas ábra. Hogyan tároljuk?



pixeleket kell tárolni

1 byte jól leírja az információkat. Ha kicsi az ábra ⇒ lehet kazétony.

3	2	1
4	X	0
5	6	7

8 különböző elmozdulás 1 byte-on megadható

$X\phi$, $Y\phi$

$r_1, r_2, r_3, \dots, r_n$ #→végjel

↳ relatív elmozdulások

$\overline{X\phi}$, $\overline{Y\phi}$

Hogyan lehet feldolgozni?

Kiolvasunk 1 pixelt, a következő pixel helyét r_i adja meg.

X, Y

r_i	x	y
ϕ	x+1	y
1	x+1	y+1
2	x	y−1
3	x−1	y−1
4	x−1	y
5	x−1	y+1
6	x	y+1
7	x+1	y+1

be: x,y pixel(x;y);

be: r

ciklus míg r ≠ végjel

elágazás r

0	x := x+1	
1	x := x+1	y := y-1
2	y := y-1	y := y-1
3	x := x-1	y := y-1
4	x := x-1	
5	x := x-1	y := y+1
6	y := y+1	
7	x := x+1	y := y+1

e vége

pixel(x; y);

be: r;

c vége;

ha (r=0) v (r=1) v (r=7) akkor x := x+1;

ha (r=3) v (r=4) v (r=5) akkor x := x-1;

ha (r=1) v (r=2) v (r=3) akkor y := y-1;

ha (r=5) v (r=6) v (r=7) akkor y := y+1;

A elsőnél 7 egyenesbe ágyazott ciklus szerepel, míg a másodiknál négy egymást követő feltétel.
A elsőnél átlagosan négy feltételt vizsgál meg, a másodiknál hatot.

hf: program 1: generál n db számot véletlenül 0 és 7 között (random(8)), ezeket kiírja 1 file-ba.

program 2: beolvassa a képernyő közepéből. Az egyes módon jelenítse meg (Gettime → eleje és vége)

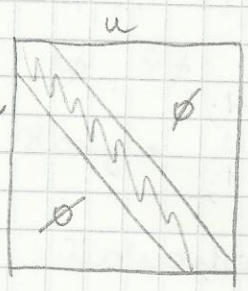
Majd a kettes módon is.

gedag@delfin.klte.hu

- Program tárigénye = Cód tárigénye + adatok tárigénye.
- Figyelni kell a tárigényre → ha az nagy ⇒ lassú a program.
  a real rövidebb (6 byte), de a longint lesz a gyorsabb
  a leggyorsabb: (műveletvégzés szempontjából): előjel nélküli
  előjeles
  valós → lebegőpontos

pl.: osztálypont tárolására a legmegfelelőbb a byte típus.

Válasszuk ki a lehető legkisebbre az elemek tárigényét!
lépésszám csökkentés → nem tároljuk el minden elemet.



diagonális mátrix: a főátlón kívül csak ϕ áll.

a_{ij} $i=j$

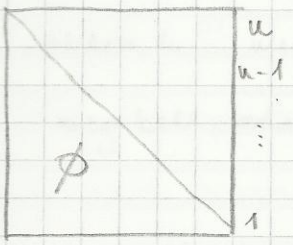
- ezt ábrázolhatjuk 1 vektorban, és írhatunk rá egy fgv-t:

fg. A(B:vektor; i,j : egész) : elemtip;

ha i=j akkor A := [i]

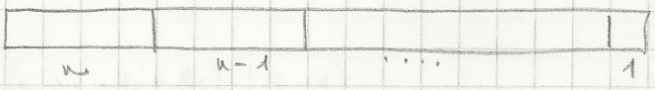
különben A := ∅.

- n x n helyett n elemű vektorban tároltuk



A főátlóban és a főátló felett nem csak ∅ áll ⇒ el kell tárolni.

így $\frac{n(n+1)}{2}$ elemet tároltunk nekünk az n^2-tel



$\frac{n(n+1)}{2}$

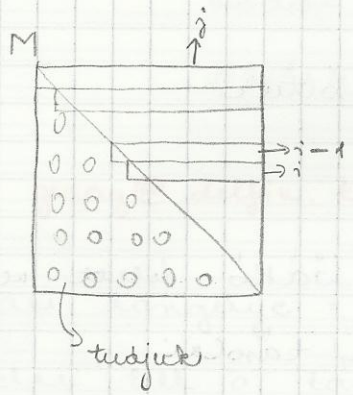
- ha i>j ⇒ főátló alatti elem

i: sor j: oszlop

i	j	k
1		j
2	3	$n+(j-1) = n+(j-1+1)$
3	5	$n+(n-1)+(j-i+1)$

10.21. 9. előadás

máj 28. (péntek) 9^{00} vizsgaidőpont



ha ez a n. $n \times n$ típusú mátrix, a tárigénye n^2.

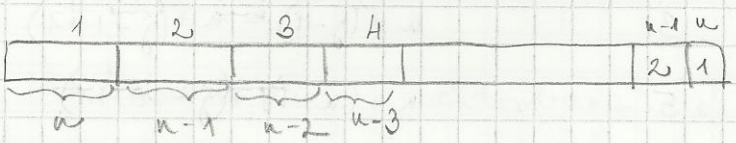
ha $j < i$ $\Rightarrow$ biztosan $\emptyset$ értékű elemet adnak meg.

1. sor	n	db
2. sor	$n-1$	db
3. sor	$n-2$	db
⋮		
n. sor	1	db

↳ megrajzjuk azon elemek tárigényét, amiről nem tudjuk, h mit tárolnak

$\frac{n(n+1)}{2}$ $\Rightarrow$ megközelítőleg a fele.

Próbáljuk meg az elemeket leképezni egy vektorra!



függv _M (i, j) : elemtip;

ha $i > j$ akkor _M := $\emptyset$.

különben _M := v[f(i,j)]

i és j függvényeként adható meg.

Ha i 2 lenne $\Rightarrow$ j értékét lehetne értelmezni.

Példa:

$M[2;4]$

↓ 2. blokk 3. elemét kell megnézniünk

$M[3;4]$ 3. blokk 2. elemét kell megvizsgálni

Ahhoz, h. meg tudjam mondani, h. kol van (pl.: 3. blokk) $\Rightarrow$ meg kell adni, mennyit lépünk. $n+(n-1)+\underbrace{\dots}_{\text{ahány elem van még}}$

$M[i,j]$

$$n+(n-1)+(n-2)+\dots+\overset{i}{(n-i+1)}$$

Nekünk az $n-(i-1)$ kell $\Rightarrow$ $n-(i-2) \Rightarrow \underline{n-i+2}$

$$\frac{(n+n-i+2)(i-1)}{2}$$ ← { az i-edik sor fölött tárolt elemek számát határoztuk meg

1. sor	n elem	
2. sor	$n-1$ elem	
⋮		
i. sor	$n-(i-1)$ elem ↕	Az i sorban tárolt elemek számára nincs szükségünk
$(i-1)$. sor	$n-(i-2)$ elem ↕	

Arra van szükségünk, hogy hány elem van az $i-1$. sorig, ha ezt tudjuk, csak az i-dik elemhez kell számolni. Így a vektorban könnyen megtalálható.

Az aktuális sorun belül $j-(i-1)$ kell előre lépni!

$$f(i,j) := \frac{(n+n-i+2)(i-1)}{2} + j - i + 1$$

M[2,2]

$$\frac{2n \cdot 1}{2} + 1 = n + 1 \quad \checkmark$$

Hf.: Levezetni arra az esetre, ha a főátlóban és a főátló fölött Ø-k vannak.

f(i,j) → címfüggvény, i és j alapján számol egy indexet.

v[f(i,j)]

Ha $i \leq j$ akkor

_ M[f(i;j)] = érték;

Ha $i \not\leq j \Rightarrow$ nem kell csinálni semmit, mert a főátló alatti Φ-kat nem tároljuk.

Fix az az érték, ill. hány Φ vagy attól különböző elem van.

Ha az elemek (Φ és nem Φ) szabálytalanul helyezkednek el:

	1	2	3	4	5	6
1	2	3				7
2						
3		5		7		
4						
5		7				11
6						
7			10			13

érték	2	3	7	5	7	7	11	10	13
sor	1	1	1	3	3	5	5	7	7
oszlop	1	2	6	2	4	2	6	3	6

A sorindexek rendezett sorozatot alkotnak ⇒ amelyben könnyű keresni. Sorfolytonosan bejárható!

Pl.: Írassuk a 3. sor elemeit!

1	2	3	4	5	6
∅	5	∅	7	∅	∅

Ha oszlopindexek szerint keresünk, már az nem alkot rendezett sorozatot, ezért más megoldási mód szükséges.
Vajon bejárható-e oszlopfolytonosan?
Vezessünk be egy újabb sort.

	1	2	3	4	5	6	7	8	9
érték				5					
sor				3					
oszlop				0					
következő elem ← K	0	4	7	6	0	0	9	0	0

oszlopos bejárás eltárolja az adott táblázatban lévő a következő elem helyét.
a mátrixban

A számokat úgy írjuk be az új sorba, hogy a mátrix 1. sor 1. oszlopától haladunk előbb az első oszlopon, majd a másodikon ... stb. végig.

Így találhatjuk meg a keresett elemet, hogy megkeressük az első adott oszlopindexűt, és az mutatni fogja a következő elemet, ha pedig nincs több elem az oszlopban ∅-t tartalmaz a táblázat.

Minél nagyobb egy elem tárigénye, annál biztosabb, hogy jobb így eltárolni.

Bevezethetünk még egy táblázatot!

	1	2	3	4	5	6	7
SE	1	0	4	0	6	0	8

↑ Hányadik oszlopban van eltárolva a táblázatban.

Tároljuk el azt a pozíciót, ahol az adott sorban az első 0-tól különböző eleme található!

	1	2	3	4	5	6
SO	1	2	8	5	0	3

Ha be akarjuk járni pl.: oszlopfolytonosan a mátrixot, kiolvasható, h. a táblázatban hol található az első 0-tól különböző elem, majd a táblázatból a többi elemét az adott sorban.

A 2 új „táblázattal" gyorsítottuk az algoritmust, de plusz adatok tárolása vált szükségessé.

Gráfok ábrázolása

- a gráf élekből és csomópontokból áll
- fa: összefüggő körmentes gráf
- teljes gráf: minden pontból minden pontba él vezet

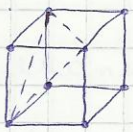
A csomópontokban adatokat tudunk tárolni, az él pedig azt adja meg, h. melyik csomópont melyikkel van kapcsolatban

10. előadás

10.28.

Útiváznodell:

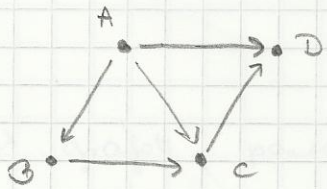
pl.: kocka

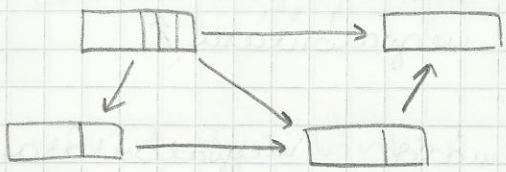


Ha az élet másképpen rajzoljuk le, arról testeket kapunk.

- A drótvázmodell egy GRÁF
- Az élek megmutatják, hogy a csúcspontok milyen kapcsolatban vannak egymással.
- A illeszkedéseket által vannak összekötve, ezek megfeleltethetők a gráf éleinek, a csúcsok.
- Az élhez lehet súlyszámot rendelni.
- A gráfot eddig 2 adatstruktúrával írtuk le:
  1., lista
  2., fa: minimálisan összefüggő, körülmentes
  (A csúcspontjába sor egyféle úton lehet eljutni)

Lehet vegyes irányított gráfok.





Ennek nem kell útmutató.

Egy elemnek annyi útmutató éle, ahány él vezet belőle.