

Algoritmusok és adatstruktúrák 9. előadás

Pusztai Pál
pusztai@sze.hu

Tartalom

- Rekurzív algoritmusok
 - Megvalósítás eljárással, függvénnyel, több eljárással
 - Megvalósítás saját veremkezeléssel, iterációval
 - Visszalépéses algoritmusok
 - Összes megoldás megkeresése
 - Egy megoldás megkeresése
- Kidolgozott feladatok
 - Faktoriális számítás
 - Gyorsrendezés rekurzív eljárással
 - Kínai gyűrűk
 - A 8 királynő probléma
 - Sakktábla bejárása egy huszárral
 - Gyorsrendezés nem rekurzív eljárással



Rekurzív algoritmusok

- Azokat az algoritmusokat nevezzük **rekurzívnak**, amelyek kifejtésében, definiálásában maga a definiálandó algoritmus is szerepel.
- Az ilyen algoritmusok csak **szubrutinokkal** definiálhatók, hiszen csak ők képesek saját magukat közvetlenül, vagy más szubrutinokon keresztül közvetve meghívni.
- Minden rekurzív algoritmus megvalósítható **rekurzió nélkül** is, rekurzióval azonban rövidebb, áttekinthetőbb, a lényegét jobban kifejező megoldást adhatunk.

Faktoriális

- **Feladat:** Határozzuk meg egy pozitív egész szám faktoriálisát!

Pl: $5! \rightarrow 120$



Faktoriális

■ Megoldás:

$$n! = 1 * 2 * 3 * \dots * (n-1) * n$$

$$n! = (n-1)! * n$$

$$0! = 1$$

Funkció	Azonosító	Típus	Jelleg
A szám, amelynek a faktoriálisát számoljuk	N	Egész	I
Az eredmény	ER	Egész	O



Faktoriális

```
/* Faktoriális kiszámítása rekurzív függvénynel */  
FAKT(N)  
if N=0  
    ER  $\leftarrow$  1  
else  
    ER  $\leftarrow$  FAKT(N-1)*N  
return ER
```



Gyorsrendezés

- A gyorsrendezés egy, az „**oszd meg és uralkodj**” elven alapuló algoritmus.
- Ezek az algoritmusok a feldolgozandó adatokat több, kisebb részre osztják, majd ezeket feldolgozva (rajtuk uralkodva) állítják elő a feladat megoldását.
- Az ilyen algoritmusokat többnyire **rekurzív szubrutinokkal** valósítják meg, amelyek a kisebb adatcsoportok feldolgozására saját magukat hívják meg rekurzívan.



Gyorsrendezés

- A gyorsrendezés rekurzív algoritmusának **paramétere** a rendezendő adatokat tartalmazó egydimenziós tömb és a feldolgozandó rész kezdő- és végindexe.
- Az eljárás lépései:
 1. Ha a feldolgozandó rész nem tartalmaz legalább két elemet, akkor készen vagyunk, hiszen az adott rész már rendezett, különben folytassuk a 2. ponttal.
 2. Elemcserékkel **válasszuk szét** az elemeket két részre úgy, hogy az első rész összes eleme legyen kisebb vagy egyenlő, mint a másik rész összes eleme.
 3. Rendezzük az **első részt**, azaz hívjuk meg az algoritmust az első részre.
 4. Rendezzük a **második részt**, azaz hívjuk meg az algoritmust a második részre.



Gyorsrendezés

- A **szétválasztást** az alábbiak szerint végezzük:
 - Válasszunk egy **középértéket** (más néven strázsa elemet), legyen ez a fizikailag (index szerint) középső elem értéke.
 - A tömbben előlről haladva keressük meg az első olyan elemet, amelyik **nem kisebb**, mint a középérték.
 - A tömbben hátulról haladva keressük meg az első olyan elemet, amelyik **nem nagyobb**, mint a középérték.
 - A két elemet **cseréljük fel**.
 - A kereséseket és cseréket a cserélt elemektől az eredeti irányokban haladva mindaddig **ismételjük**, amíg a két oldal nem találkozik.
- A **találkozási pont** két részre osztja a tömböt, elől a középértéknél nem nagyobb, hátul a középértéknél nem kisebb értékű elemek találhatók.

Gyorsrendezés

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

A szétválasztás elemcseréi (strázsa:5)

- A gyorsrendezés műveletigénye:
 - Legrosszabb esetben n^2 nagyságrendű.
 - Átlagosan $n \cdot \log_2 n$ nagyságrendű.



Gyorsrendezés

■ Típus

ELEM Egész

/* A rendezendő elemek típusa */

TOMB Egydimenziós ELEM tömb

/* Az elemeket tároló tömb típusa */

Funkció	Azonosító	Típus	Jelleg
A rendezendő elemek	A	TOMB	I, M, O
A rendezendő rész kezdete	K	Egész	I
A rendezendő rész vége	V	Egész	I
Az előlről induló pásztázó index	I	Egész	M
A hátulról induló pásztázó index	J	Egész	M
A strázsa elem	S	ELEM	M
Két elem cseréjéhez	CS	ELEM	M

Gyorsrendezés

```
/* Gyorsrendezés rekurzívan */
```

```
GYORSREND(A,K,V)
```

```
if K<V
```

```
    /* Van legalább két elem */
```

```
    I ← K
```

```
    J ← V
```

```
    S ← A[(I+J) DIV 2]
```

```
    /* Szétválogatás a strázsa (S) elemhez képest */
```

```
    ...
```



Gyorsrendezés

```
...
/* Szétválogatás a strázsa (S) elemhez képest */
while I <= J
    while A[I] < S
        I ← I + 1
    while A[J] > S
        J ← J - 1
    if I <= J
        /* Az I. és J. elemek cseréje */
        CS ← A[I]
        A[I] ← A[J]
        A[J] ← CS
        I ← I + 1
        J ← J - 1
/* Az első rész rendezése rekurzív hívással */
GYORSREND(A, K, J)
/* A második rész rendezése rekurzív hívással */
GYORSREND(A, I, V)
```



Gyorsrendezés

- Megjegyzés:
 - A kezdőhívás paraméterei:
 - A rendezendő elemeket tartalmazó tömb
 - Az első és utolsó elem indexe
 - Az $I=J$ esetben történő felesleges elemcsere egy újabb feltétellel kivédhető, de ekkor a feltétel kiértékelése kerül „plusz” időbe.



Kínai gyűrűk

- **Kínai gyűrűk:**
 - Egy mechanikus játék, amelyben adott számú (N) gyűrű található.
 - Minden gyűrűnek kétféle állapota van: **fent** vagy **lent**.
- **Feladat:** Vegyük le a kezdetben fent lévő összes gyűrűt az alábbi szabályok betartásával:
 - Egyszerre csak egy gyűrű mozgatható.
 - Az 1. gyűrű szabadon mozgatható.
 - Az N . gyűrű ($N > 1$) mozgatásához
 - Az $N-1$. gyűrűnek fent kell lennie,
 - Az $N-2$., ..., 1. gyűrűknek lent kell lennie.



Kínai gyűrűk

■ **Megoldás:**

- Három, egymást rekurzívan hívó és egy kiíró szubrutin.
- A gyűrűk állapotát egy egydimenziós (globális) tömbben tároljuk:
 - 1:fent, 0:lent van az adott gyűrű.



Kínai gyűrűk

- **Feladat:** Vegyünk le egy adott gyűrűt!



Kínai gyűrűk

- **Feladat:** Vegyünk le egy adott gyűrűt!

Funkció	Azonosító	Típus	Jelleg
A gyűrűk aktuális állapota	GYURUK	Egydimenziós egész tömb	I, O
A gyűrű sorszáma	N	Egész	I

/* Az N. gyűrű levétele */

LE(N)

if GYURUK[N]=1

if N>1

 FEL(N-1)

if N>2

 BALRALE(N-2)

GYURUK[N] ← 0

KIIR

Kínai gyűrűk

- Feladat:** Tegyük fel egy adott gyűrűt!

Funkció	Azonosító	Típus	Jelleg
A gyűrűk aktuális állapota	GYURUK	Egydimenziós egész tömb	I, O
A gyűrű sorszáma	N	Egész	I

/* Az N. gyűrű feltétele */

FEL(N)

if GYURUK[N]=0

if N>1

 FEL(N-1)

if N>2

 BALRALE(N-2)

GYURUK[N] ← 1

KIIR

Kínai gyűrűk

- **Feladat:** Vegyünk le egy adott gyűrűt, és az azt megelőző összes gyűrűt!

Funkció	Azonosító	Típus	Jelleg
A gyűrűk aktuális állapota	GYURUK	Egydimenziós egész tömb	I, O
A gyűrű sorszáma	N	Egész	I
Az aktuális gyűrű	I	Egész	M

/* Az N. és a megelőző gyűrűk levétele */

BALRALE(N)

for I \leftarrow N, 1, -1

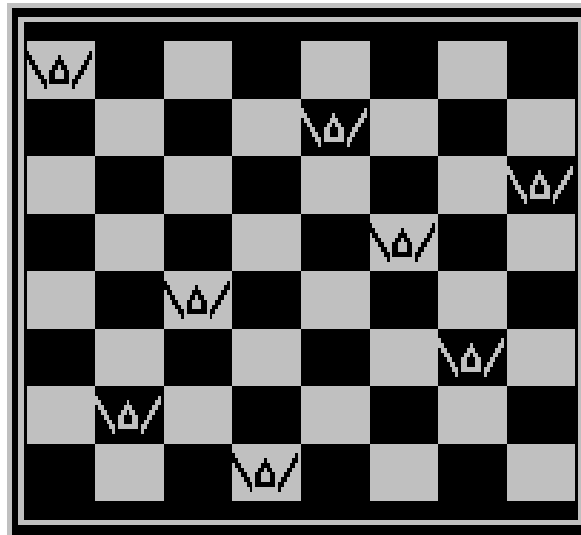
LE(I)

Kínai gyűrűk

- Megjegyzés:
 - A kezdőhívás előtt a **gyűrűk állapotát** jelző tömbbe (GYURUK) 1-es értékeket kell tennünk.
 - A **kezdőhívás** BALRALE(N), ahol N a gyűrűk száma.
 - A **megoldások** kiírását a KIIR szubrutin végzi a GYURUK tömb alapján.

Nyolc királynő

- **Feladat:** Helyezzünk el egy sakktáblán 8 db királynőt úgy, hogy azok ne üssék egymást, azaz ne essenek egymás ütésvonalába!



A 8 királynő probléma egy megoldása

Összes megoldás megkeresése

Próbáljunk új választást

for Az összes választáson

Válasszuk ki az adott választást

if Megfelelő

Jegyezzük fel

if Megoldás nem teljes

/* Rekurzív hívás */

Próbáljunk új választást

else

Megoldás kiírása

A feljegyzés törlése

Nyolc királynő

Funkció	Azonosító	Típus	Jelleg
A sakktábla mérete	N	Egész	I
A királynők helye	HOL	Egydimenziós egész tömb	I, M, O
Az aktuális sor sorszáma	S	Egész	I
A főátlók foglaltsága	FATL	Egydimenziós logikai tömb	I, M, O
A mellékátlók foglaltsága	MATL	Egydimenziós logikai tömb	I, M, O
Az oszlopok foglaltsága	OSZL	Egydimenziós logikai tömb	I, M, O
Az aktuális oszlop	O	Egész	M

■ Megjegyzés:

- Ha MAXMERET jelöli a maximális méretet, akkor a HOL és OSZL tömbök MAXMERET eleműek, az FATL, MATL tömbök $2 \cdot \text{MAXMERET} - 1$ eleműek.

Nyolc királynő

/* Egy királynő elhelyezése az S. sorba */

PROBAL(S)

/* Az összes választáson */

for O \leftarrow 1,N

/* S. sorba az O. oszlopba téve az S. királynőt */

/* Megfelelő? */

if NOT OSZL[O] AND NOT FATL[S-O+N] AND NOT MATL[S+O-1]

/* Jegyezzük fel */

HOL[S] \leftarrow O

OSZL[O] \leftarrow igaz

FATL[S-O+N] \leftarrow igaz

MATL[S+O-1] \leftarrow igaz

/* A megoldás nem teljes? */

...

Nyolc királynő

```
...
/* A megoldás nem teljes? */
if S<N
    /* Rekurzív hívás */
    PROBAL(S+1)
else
    MEGOKIIR
/* A feljegyzés törlése */
OSZL[O] ← hamis
FATL[S-O+N] ← hamis
MATL[S+O-1] ← hamis
```




Nyolc királynő

- Megjegyzés:
 - A szubrutin kezdőhívása előtt a **foglaltságokat** jelző logikai tömbökbe (OSZL, FATL, MATL) hamis kezdőértékeket kell tennünk.
 - A **kezdőhívás** 1-es paraméterrel végzendő, a **megoldások** kiírását a MEGOKIIR szubrutin végzi a HOL tömb alapján.



Huszar útja

- **Feladat:** Járjunk be egy $N \times N$ -es méretű „sakktáblát”, egy adott kezdőmezőről indulva úgy, hogy minden mezőre pontosan egyszer lépünk!

	8		1	
7				2
				
6				3
	5		4	

Huszar lehetséges lépései

1	20	17	12	3
16	11	2	7	18
21	24	19	4	13
10	15	6	23	8
25	22	9	14	5

Huszar útja egy 5×5 -ös táblán

Egy megoldás megkeresése

Próbáljunk új választást

A választás előkészítése

repeat

 Válasszuk ki a következő választást

if Megfelelő

 Jegyezzük fel

if Megoldás nem teljes

 /* Rekurzív hívás */

 Próbáljunk új választást

if NOT (Van Megoldás)

 A feljegyzés törlése

else

 Van megoldás \leftarrow igaz

until (Van megoldás) OR (Nincs több választás)

Huszár útja

■ Konstans

LEPS= $(-2, -1, 1, 2, 2, 1, -1, -2)$

/* Relatív sor elmozdulások */

LEPO= $(1, 2, 2, 1, -1, -2, -2, -1)$

/* Relatív oszlop elmozdulások */

■ Típus

MEZO Rekord

S, O

Egész



Huszár útja

Funkció	Azonosító	Típus	Jelleg
A sakktábla mérete	N	Egész	I
A sakktábla	TABLA	Kétdimenziós egész tömb	I, M, O
Az aktuális lépés sorszáma	LEPES	Egész	I
Az aktuális mező	AKT	MEZO	I
Van-e megoldás	VANMEGO	Logikai	M, O
Az aktuális lépés iránya	IRANY	Egész	M
A következő lépés mezője	KOV	MEZO	M

■ Megjegyzés:

- Ha MAXMERET jelöli a maximális méretet, akkor a TABLA tömb $\text{MAXMERET} \times \text{MAXMERET}$ elemű.

Huszár útja

```
/* A huszár következő lépése */
```

```
PROBAL(LEPES,AKT)
```

```
/* A választás előkészítése */
```

```
IRANY  $\leftarrow$  1
```

```
repeat
```

```
    /* Válasszuk ki a következő választást */
```

```
    KOV.S  $\leftarrow$  AKT.S+LEPS[IRANY]
```

```
    KOV.O  $\leftarrow$  AKT.O+LEPO[IRANY]
```

```
    /* Megfelelő? */
```

```
    if (KOV.S $\geq$ 1) AND (KOV.S $\leq$ N) AND
```

```
        (KOV.O $\geq$ 1) AND (KOV.O $\leq$ N) AND
```

```
        (TABLA[KOV.S,KOV.O]=0)
```

```
        /* Jegyezzük fel */
```

```
        ...
```

Huszár útja

```

...
/* Jegyezzük fel */
TABLA[KOV.S,KOV.O] ← LEPES
/* A megoldás nem teljes? */
if LEPES<N*N
    /* Rekurzív hívás */
    PROBAL(LEPES+1,KOV)
    if NOT VANMEGO
        /* A feljegyzés törlése */
        TABLA[KOV.S,KOV.O] ← 0
else
    VANMEGO ← igaz
    IRANY ← IRANY+1
until VANMEGO OR (IRANY>8)
    
```



Huszár útja

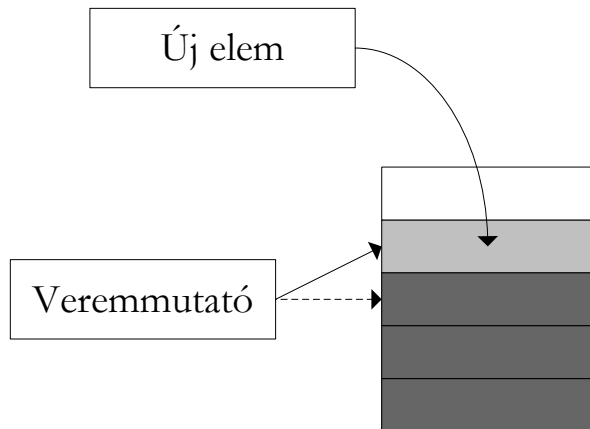
- Megjegyzés:
 - A szubrutin kezdőhívása előtt be kell állítanunk a **kezdőmezőt**, a táblába ide 1-t, a többi helyre 0-t kell tennünk. A VANMEGO értékét *hamisra* kell állítanunk.
 - A **kezdőhívás** 2 lépésszámmal végzendő, az **eredményt** a VANMEGO ill. a TABLA változók tartalmazzák.



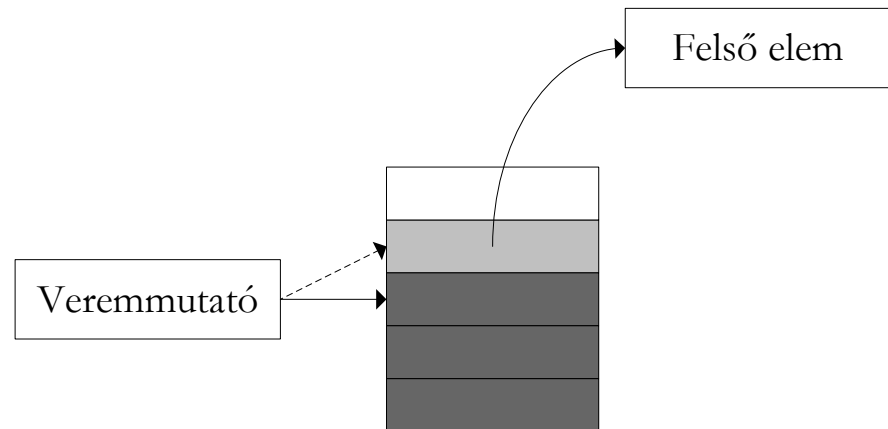
Verem

- A **verem**, mint összetett adatstruktúra műveletei:
 - Ráhelyezés (**push**): egy elemet teszünk a verembe, ez lesz a verem legfelső eleme.
 - Levétel (**pop**): a verem legfelső elemét kivesszük, eltávolítjuk a veremből.

Ráhelyezés



Levétel



Veremműveletek

Verem

- **Feladat:** Rendezzünk egy tömbben lévő adatsort növekvő (nem csökkenő) sorrendbe a gyorsrendezés módszerével, saját veremkezeléssel!

- **Típus**

ELEM Egész

TOMB Egydimenziós ELEM tömb

VEREMELEM Rekord

 K Egész

 V Egész

TVEREM Egydimenziós VEREMELEM tömb

/* A rendezendő elemek típusa */

/* Az elemeket tároló tömb típusa */

/* A rendezendő rész kezdete */

/* A rendezendő rész vége */

/* A verem típusa */

Verem

- **Feladat:** Tegyük egy elemet a verembe!



Verem

- **Feladat:** Tegyük egy elemet a verembe!

Funkció	Azonosító	Típus	Jelleg
A verem	VEREM	TVEREM	I, O
A veremmutató	VEREMMUT	Egész	I, O
A rendezendő rész kezdete	K	Egész	I
A rendezendő rész vége	V	Egész	I

```
/* Adat betétele a verembe */  
VEREMBE(VEREM,VEREMMUT,K,V)  
VEREMMUT  $\leftarrow$  VEREMMUT+1  
VEREM[VEREMMUT].K  $\leftarrow$  K  
VEREM[VEREMMUT].V  $\leftarrow$  V
```



Verem

- **Feladat:** Vegyünk ki egy elemet a veremből!

Funkció	Azonosító	Típus	Jelleg
A verem	VEREM	TVEREM	I, O
A veremmutató	VEREMMUT	Egész	I, O
A rendezendő rész kezdete	K	Egész	O
A rendezendő rész vége	V	Egész	O

```
/* Adat kivétele a veremből */  
VEREMBOL(VEREM,VEREMMUT,K,V)  
K ← VEREM[VEREMMUT].K  
V ← VEREM[VEREMMUT].V  
VEREMMUT ← VEREMMUT-1
```

Verem

Funkció	Azonosító	Típus	Jelleg
A rendezendő elemek	A	TOMB	I, M, O
A rendezendő elemek száma	N	Egész	I
A verem	VEREM	TVEREM	M
A veremmutató	VEREMMUT	Egész	M
A rendezendő rész kezdete	K	Egész	M
A rendezendő rész vége	V	Egész	M
Az előlről induló pásztázó index	I	Egész	M
A hátulról induló pásztázó index	J	Egész	M
A strázsa elem	S	ELEM	M
Két elem cseréjéhez	CS	ELEM	M

Verem

```
/* Gyorsrendezés nem rekurzívan */  
GYORSREND(A,N)  
/* A verem inicializálása */  
VEREMMUT  $\leftarrow$  0  
/* A teljes rendezendő részt a verembe */  
VEREMBE(VEREM,VEREMMUT,1,N)  
while VEREMMUT>0  
    /* A rendezendő rész kivétele a veremből */  
    VEREMBOL(VEREM,VEREMMUT,K,V)  
    if K<V  
        /* Van legalább két elem */  
        I  $\leftarrow$  K  
        J  $\leftarrow$  V  
        S  $\leftarrow$  A[(I+J) DIV 2]  
        /* Szétválogatás a strázsa (S) elemhez képest */  
        ...
```



Verem

```

...
/* Szétválogatás a strázsa (S) elemhez képest */
while I ≤ J
    while A[I] < S
        I ← I + 1
    while A[J] > S
        J ← J - 1
    if I ≤ J
        /* Az I. és J. elemek cseréje */
        CS ← A[I]
        A[I] ← A[J]
        A[J] ← CS
        I ← I + 1
        J ← J - 1
/* A két rész felvétele a verembe */
...

```



Verem

```
...  
/* A két rész felvétele a verembe */  
if  $J - K > V - I$   
    VEREMBE(VEREM, VEREMMUT, K, J)  
    VEREMBE(VEREM, VEREMMUT, I, V)  
else  
    VEREMBE(VEREM, VEREMMUT, I, V)  
    VEREMBE(VEREM, VEREMMUT, K, J)
```

