

Nyers erő módszere
Rekurzió
Oszd-meg-és-uralkodj

2. ELŐADÁS
Sapientia-EMTE
2020-21



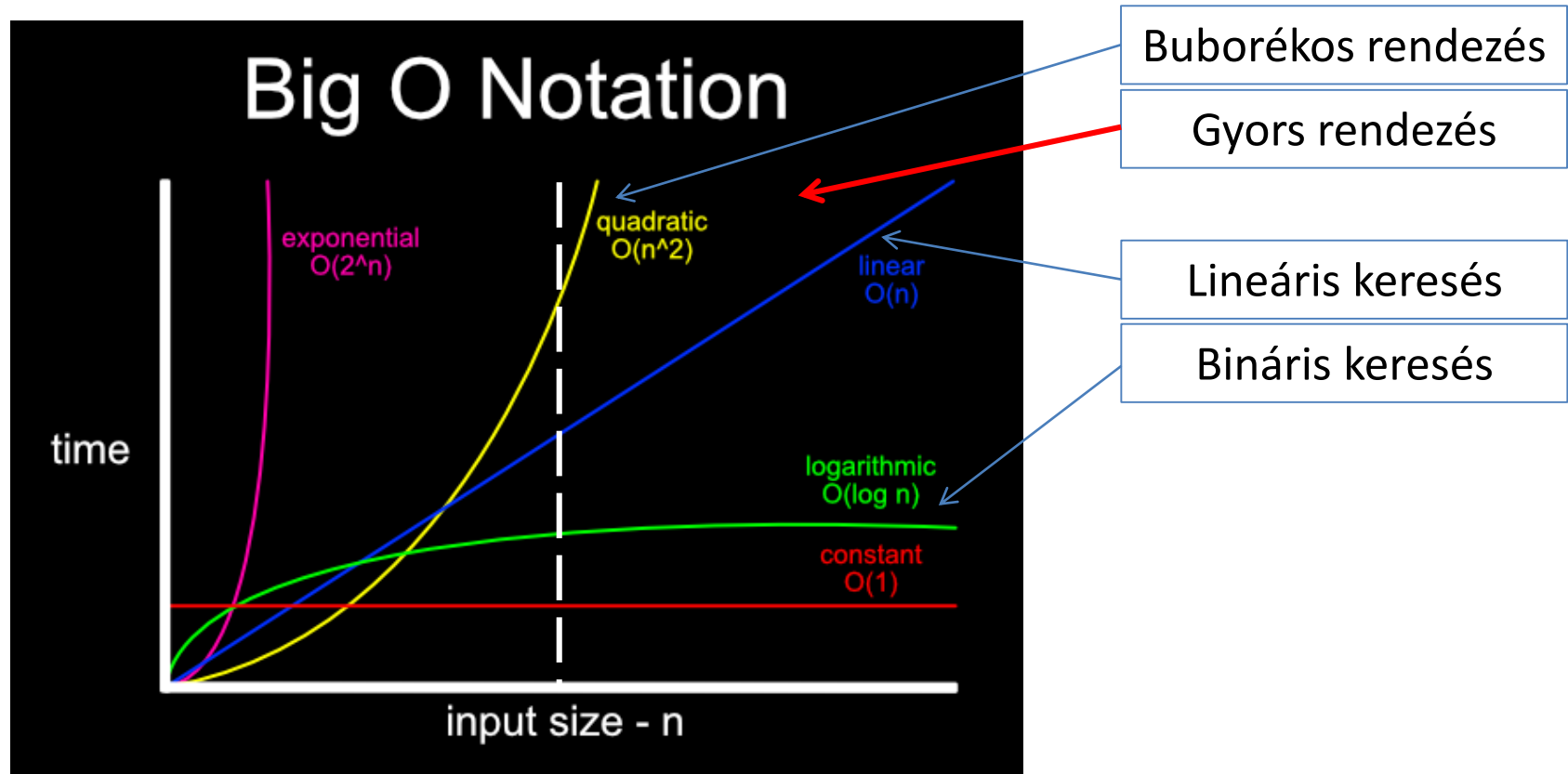
Algoritmus

- Az „algoritmus” kifejezés a bagdadi arab tudós, al-Hvárizmi (780-845) nevének eltorzított, rosszul latinra fordított változatából ered.
- Az első, számítógépre kigondolt algoritmust Ada Lovelace írta 1843-ban Charles Babbage analitikai gépére a Bernoulli-számok kiszámítására, tehát ő tekinthető az első programozónak.



Algoritmusok hatékonysága (1)

- Lépésszám, legrosszabb esetben, a bemenet mérete (n) függvényében



Algoritmusok hatékonysága (2)

- Lépésszám, legrosszabb esetben, a bemenet mérete (n) függvényében

• $\log n \implies$ **remek**

Bináris keresés

• $n \implies$ **nagyon jó**

Lineáris keresés

• $n \log n \implies$ **egész jó**

Gyors rendezés

• $n^2 \implies$ **nem túl nagy n -re jó**

Buborékos rendezés

• $n^3 \implies$ **nem túl nagy n -re lehet jó**

• $n^4 \implies$ **nem túl nagy n -re néha jó**

• $n^{10} \implies$ **50 év múlva, nem túl nagy n -re lehet jó**

• $n^{100} \implies$ **1 millió év múlva lehet jó**

• $2^n \implies$ **nagyon kis n -re lehet jó, de nagy n -re sohasem**

Algoritmustervezési stratégiák

- (0) „Nyers erő” (Brute force)
- (1) „Oszd meg és uralkodj” (Divide et impera)
- (2) Visszalépéses keresés (Backtracking)
- *Divide et impera vs. Backtracking*
- (3) Mohó (Greedy)
- *Backtracking + Greedy*
- (4) Dinamikus programozás (Dynamic programming)
- *Dinamikus programozás vs. Divide et impera*
- *Dinamikus programozás vs. Greedy*
- (5) Ág és korlát (Branch and bound)

„Nyers erő” módszere

- A legkézenfekvőbb algoritmust jelenti, amelyet a feladat megfogalmazása vagy a hivatkozott fogalmak definíciója elsőre sugall



PÉLDÁK

```
hatvány(x,n) //  $x^n = x * x * \dots * x$   
    p = 1  
    minden i = 1,n végezd  
        p = p * x  
    vége minden  
    return p  
vége hatvány
```

Rendezd az a[1..n] tömböt

Visszatéríti az a[i..n] tömbszakasz legkisebb elemének indexét

```
kiválasztásos_rendezés(a[],n)  
    minden i = 1,n-1 végezd  
        j = min_index(a,i,n)  
        csere(a[i],a[j])  
    vége minden  
vége kiválasztásos_rendezés
```

$O(n^2)$

```
buborékos_rendezés(a[],n)  
    minden i = n,2,-1 végezd  
        végérebuborékoztat(a,i)  
    vége minden  
vége buborékos_rendezés
```

Az a[1..i] tömbszakasz legnagyobb elemét kibuborékoztatja az a[i] pozícióba

Visszatéríti az utolsó csere helyét

```
finomított_buborékos_rendezés(a[],n)  
    vég = n  
    amíg vég > 0 végezd  
        vég = végérebuborékoztat(a,vég)  
    vége minden  
vége finomított_buborékos_rendezés
```

Rendezd az a[1..n] tömböt

$O(n^2)$

REKURZIÓ – emlékeztető

- A kurrens feladat megoldása visszavezethető hasonló, „egyszerűbb” részfeladat(ok) megoldására
 - Kicsinyítsd és urald! (Decrease-and-conquer)
 - Oszd meg és urald! (Divide-and-conquer)



divide
&
CONQUER!

Kicsinyíts és urald! (1)

- Méretkicsinyítés
 - állandó értékkel
 - pl. $a^n = a^{n-1} * a$
 - adott faktorra
 - pl. $a^n = a^{n/2} * a^{n/2}$
 - változó értékkel
 - pl. Inko

```
hatvány1(x, n) //  $x^n = x^{n-1} * x$   
    ha n == 0 akkor  
        return 1  
    különben  
        return hatvány1(x, n-1) * x  
    vége ha  
vége hatvány1
```

```
hatvány2(x, n) //  $x^n = x^{n/2} * x^{n/2}$   
    ha n == 0 akkor  
        return 1  
    különben  
        ha n%2 == 0 akkor  
            h = hatvány2(x, n/2)  
            return h*h  
        különben  
            h = hatvány2(x, (n-1)/2)  
            return h*h*x  
    vége ha  
vége ha
```

```
vége ha lnko(m, n) // lnko(m, n) = lnko(n, m%n)  
    ha m == n akkor  
        return n  
    különben  
        return lnko(n, m%n)  
    vége ha  
vége lnko
```

Kicsinyíts és urald! (2)

Rendezd az $a[1..n]$ tömböt

Rendezd az $a[1..n-1]$ tömbszakaszt

```
beszúrásos_rendezés(a[], n)
  if n > 1 akkor
    beszúrásos_rendezés(a, n-1)
    beszúrutolsóelem(a, n)
  vége ha
vége beszúrásos_rendezés
```

$O(n^2)$

Beszúrja az $a[n]$ elemet a már rendezett $a[1..n-1]$ tömbszakaszba



Oszd meg és urald!

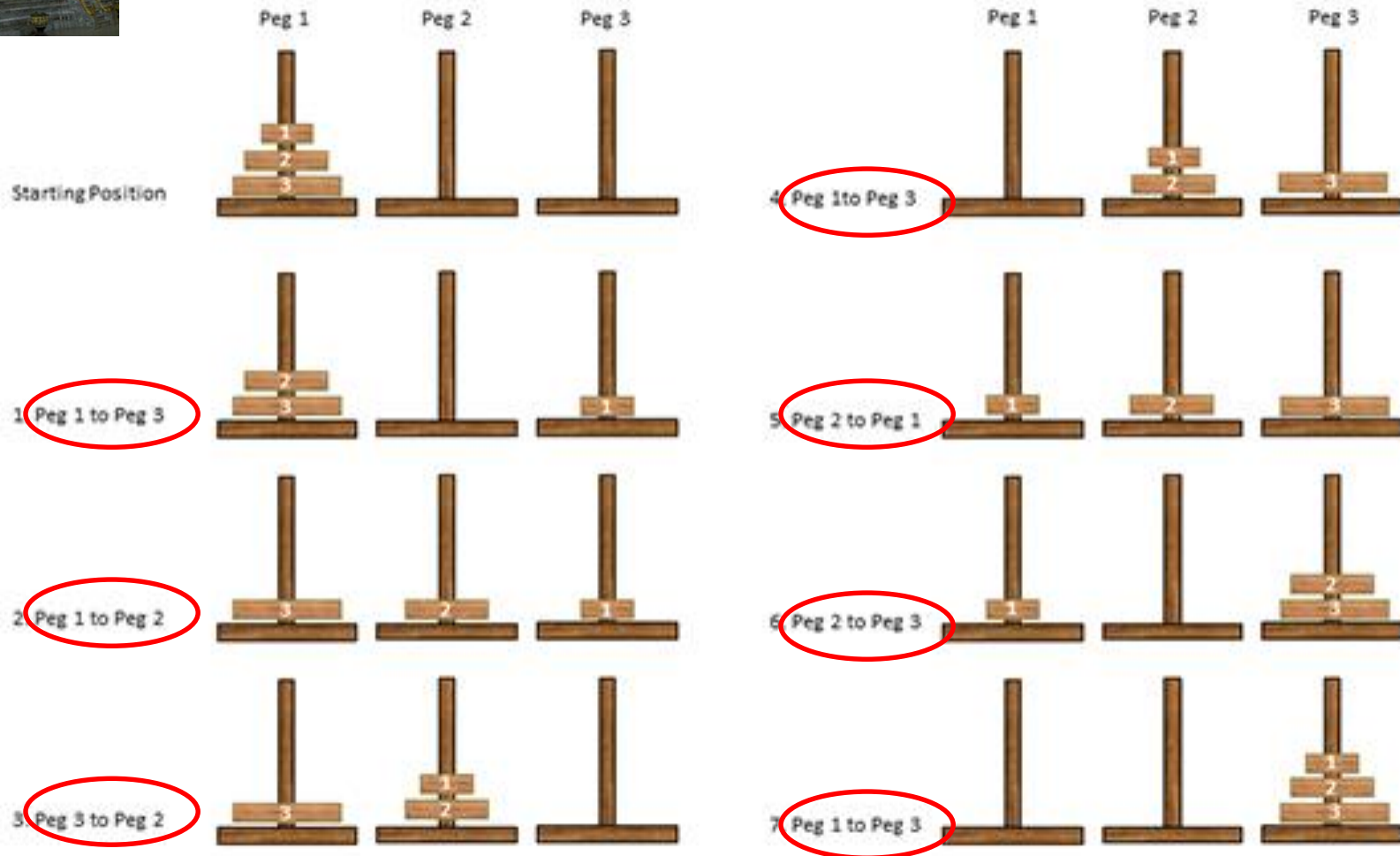
- Milyen feladatok?
 - amelyek visszavezethetők/lebonthatók, **két vagy több** hasonló, egyszerűbb részfeladatra

```
eljárás Név(az_általános_feladat_paramétereit)
    ha triviális akkor
        < oldd meg >
    különben
        < határozd meg a részfeladatait >
        < rekurzív hívások által oldd meg ezeket >
        < építsd fel a megoldást >
    vége ha
vége eljárás
```



Hanoi tornyai (1)

$2^n - 1$ steps





Hanoi tornyai (2)

k korong áthelyezése s-ről d-re h segítségével

az általános feladat paraméterei

a trivialitás feltétele

```
hanoi(k, s, d, h)
  ha k == 1 akkor
    ki: '(', s, ' ', ' ', d, ')'
  különben
    hanoi(k-1, s, h, d)
    ki: '(', s, ' ', ' ', d, ')'
    hanoi(k-1, h, d, s)
  vége ha
vége hanoi
```

a triviális feladat megoldása

a fiúrészfeladatok paraméterei

k-1 korong áthelyezése h-ról d-re s segítségével

k-1 korong áthelyezése s-ről h-ra d segítségével

Maximum-keresés

```
maxindex(a[], i, j)
  ha i == j akkor
    return i
```

Visszatéríti az $a[i..j]$ tömbszakasz legnagyobb elemének indexét

```
különben
```

```
  m1=maxindex(a, i, (i+j)/2)
```

```
  m2=maxindex(a, (i+j)/2+1, j)
```

```
  ha  $a[m1] > a[m2]$  akkor
```

```
    return m1
```

```
  különben
```

```
    return m2
```

```
  vége ha
```

```
vége ha
```

```
vége maxindex
```

Visszatéríti az $a[i..(i+j)/2]$ tömbszakasz legnagyobb elemének indexét

Visszatéríti az $a[(i+j)/2+1..j]$ tömbszakasz legnagyobb elemének indexét

Összefésüléssel rendezés

$O(n \cdot \log(n))$

`mergesort(x[], i, j)`

`ha  $i < j$  akkor`

`$k = (i+j)/2$`

`mergesort(x, i, k)`

`mergesort(x, k+1, j)`

`összefésül(x, i, k, j)`

`vége ha`

`vége mergesort`

Rendezd az $a[i..j]$ tömbszakaszt

Rendezd az $a[i..k]$ tömbszakaszt

Rendezd az $a[k+1..j]$ tömbszakaszt

Fésüld össze az $a[i..k]$ és $a[k+1..j]$ tömbszakaszokat

Gyors-rendezés

$O(n \cdot \log(n))$

quicksort(a[], i, j)

ha $i < j$ akkor

$k = \text{szétválogat}(a, i, j)$

quicksort(a, i, k-1)

quicksort(a, k+1, j)

vége ha

vége quicksort

Rendezd az $a[i..j]$ tömbszakaszt

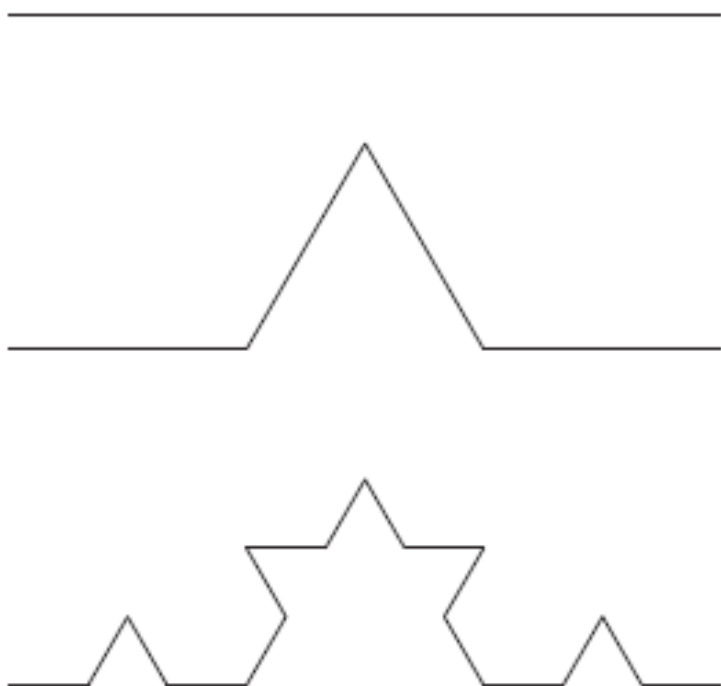
Válogasd szét az $a[i..j]$ tömbszakaszt, és térítsd vissza hol találta meg helyét az $a[i]$ elem

Rendezd az $a[i..k-1]$ tömbszakaszt

Rendezd az $a[k+1..j]$ tömbszakaszt

Koch-fraktál

Rajzolj d hosszú vonalat α szög alatt



$\text{koch}(d, \alpha, k)$

ha $k == n$ akkor

rajzol(d, α)

különben

$\text{koch}(d/3, \alpha, k+1)$

$\text{koch}(d/3, \alpha + \pi/3, k+1)$

$\text{koch}(d/3, \alpha - \pi/3, k+1)$

$\text{koch}(d/3, \alpha, k+1)$

vége ha

vége koch

Rajzolj $d/3$ hosszú vonalat α szög alatt

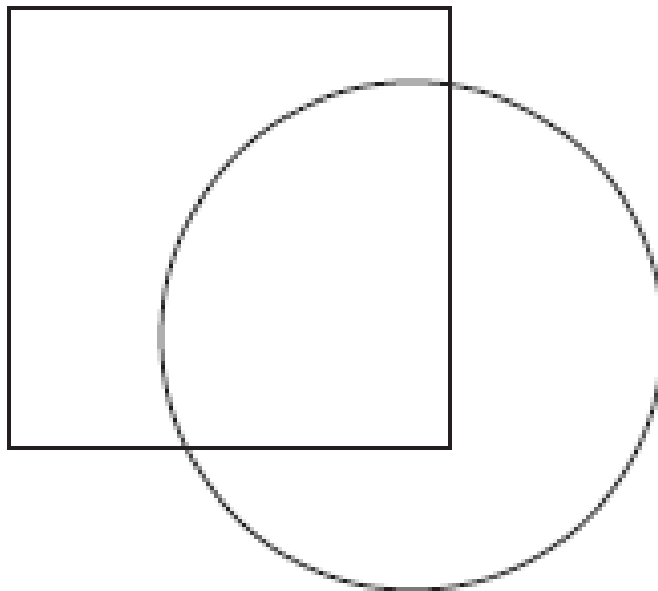
Rajzolj $d/3$ hosszú vonalat $\alpha + \pi/3$ szög alatt

Rajzolj $d/3$ hosszú vonalat α szög alatt

Rajzolj $d/3$ hosszú vonalat $\alpha - \pi/3$ szög alatt

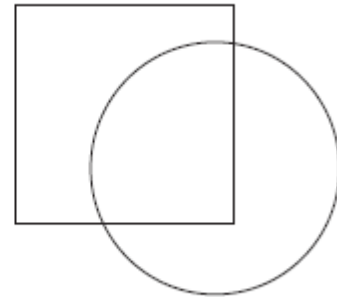
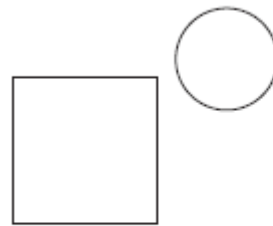
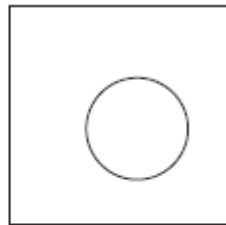
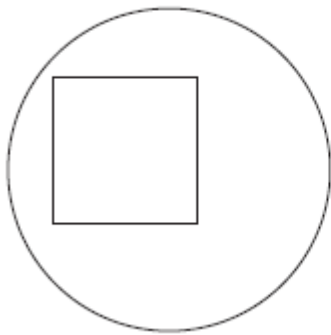
Négyzet és kör (1)

(átfedési területe méretét 3 tizedes pontossággal)



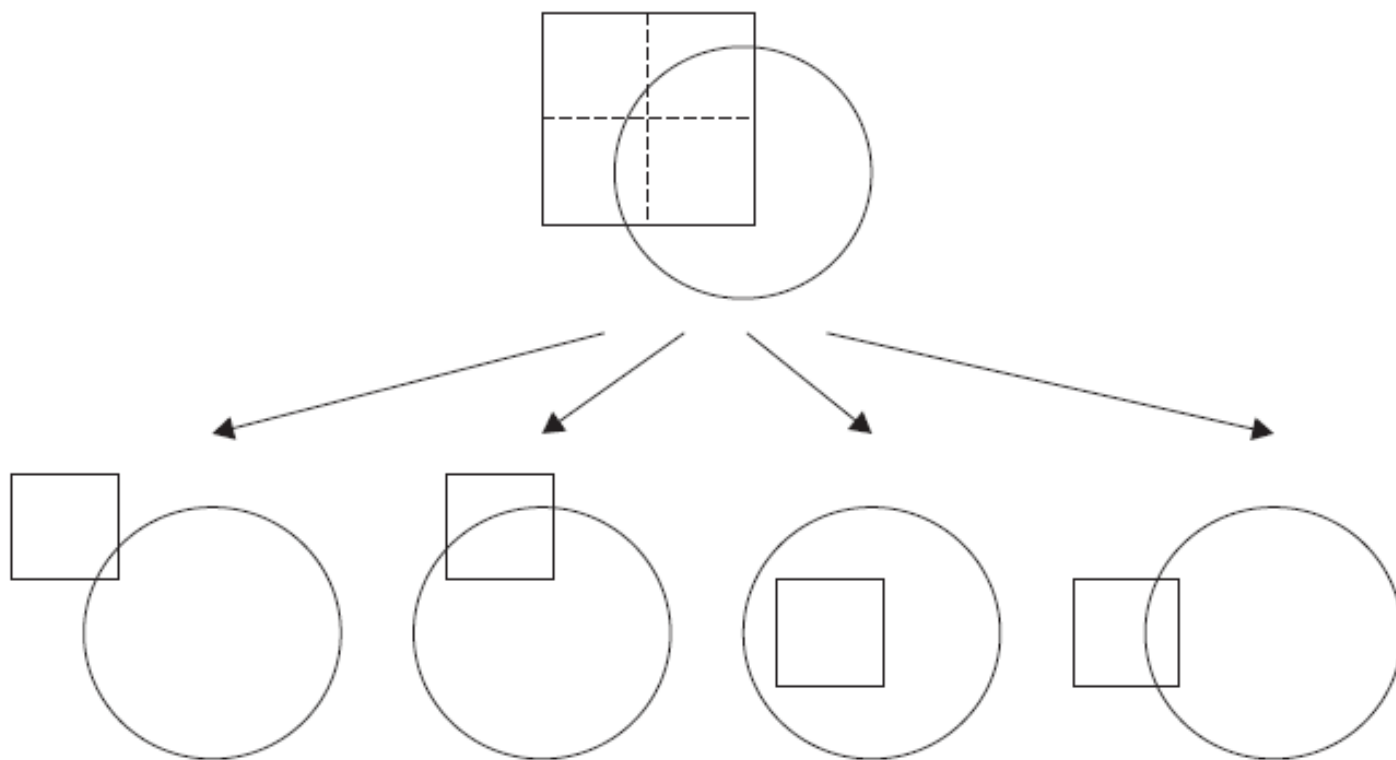
Négyzet és kör (2)

(átfedési területe méretét 3 tizedes pontossággal)



Négyzet és kör (3)

(átfedési területe méretét 3 tizedes pontossággal)



Négyzet és kör (4)

(átfedési területe méretét 3 tizedes pontossággal)

```
négyzet_és_kör(x,y,o,xk,yk,rk)
  ha o*o < 0.001 akkor
    return 0
  vége ha
  i=kör_négyzet_helyzete(x,y,o,xk,yk,rk)
  ha i == 1 akkor
    return o*o
  vége ha
  ha i == 2 akkor
    return  $\pi$ *rk*rk
  vége ha
  ha i == 3 akkor
    return 0
  vége ha
  ha i == 4 akkor
    t1=négyzet_és_kör(x1+o/2,y1,o/2,xk,yk,rk)
    t1=négyzet_és_kör(x1+o/2,y1+o/2,o/2,xk,yk,rk)
    t1=négyzet_és_kör(x1,y1+o/2,o/2,xk,yk,rk)
    t1=négyzet_és_kör(x1,y1,o/2,xk,yk,rk)
    return t1+t2+t3+t4
  vége ha
vége négyzet_és_kör
```

Alakítsd át, hogy uralhasd

- Egyszerűsítsd, hogy uralhasd
 - Például hatékonyabb algoritmust eredményezhet, ha előre rendezed a bementet
 - Melyik érték fordul elő a legtöbbször egy adott sorozatban?
 - $O(n \cdot \log(n))$: rendezd, majd keresd meg a leghosszabb konstans részsorozatot
- Ábrázold másként, hogy uralhasd
 - Gausz módszer egyenletrendszer-megoldáshoz
- Vezesd vissza, hogy uralhasd
 - Vezesd vissza egy olyan feladatra, amelyre tudsz hatékony algoritmust
 - $lkkt(m,n) = (m \cdot n) / lko(m,n)$

Ismétlesként

- Decrease-and-conquer
 - Az $x^n = x^{n/2} * x^{n/2}$ felbontás hatékonyabb algoritmust eredményez, mint az $x^n = x^{n-1} * x$
- Divide-and-conquer
 - A „divide et impera” rendezések (merge-sort, quick-sort) hatékonyabbak, mint a nyers erő megközelítések (selection-sort, bubble-sort)
- Transform-and-conquer
 - $lkkt(m,n) = (m * n) / lnko(m,n)$