Rendezési algoritmusok

belsõ rendezés

külsõ rendezés

belsõ rendezési algoritmusok

- buborékrendezés (Bubble sort)
- kiválasztó rendezés (Selection sort)
- számláló rendezés (Counting sort)
- beszúró rendezés (Insertion sort)
- Shell-rendezés (Shell sort)
- összefésülõ rendezés (Merge sort)
- kupacrendezés (Heapsort)
- gyorsrendezés (Quicksort)
- edényrendezés (Bucket sort)
- számjegyes rendezés (Radix sort)

Buborékrendezés (Bubble sort)

BUBORÉKRENDEZÉS-1(A)

1. $n := hossz[A]$
2. **repeat**
3. $\quad ind := 0$
4. $\quad$ **for** $i := 1$ **to** $n - 1$
5. $\quad\quad$ **do if** $A_i > A_{i+1}$
6. $\quad\quad\quad$ **then** $A_i \Leftrightarrow A_{i+1}$
7. $\quad\quad\quad\quad ind := 1$
8. **until** $ind = 0$
9. **return** A

repeat ... until $ind = 0$

helyett C-ben:

do... while $ind\,! = 0$

Legrosszabb esetben az összehasonlítások száma:

$$(n-1) + (n-2) + \ldots + 1 = \frac{(n-1)n}{2} = \Theta(n^2).$$

BUBORÉKRENDEZÉS-2(A)

```
1.  ind := hossz[A]
2.  repeat
3.      k := ind − 1
4.      ind := 0
5.      for i := 1 to k
6.          do if A_i > A_{i+1}
7.                then A_i ⇔ A_{i+1}
8.                     ind := i
9.  until ind = 0
10. return A
```

Legrosszabb esetben az összehasonlítások száma szintén:

$$(n-1)+(n-2)+\ldots+1=\frac{(n-1)n}{2}=\Theta(n^2).$$

Átlagérték: $\sim \frac{n^2}{4}=\Theta(n^2)$

Példa

Kiválasztó rendezés (Selection sort)

Kiválasztjuk a legkisebb elemet, és az elsõ helyre tesszük, majd úgyanígy folytatjuk a maradék sorozattal.

KIVÁLASZTÓ-RENDEZÉS(A)

```
1.  n := hossz[A]
2.  for i := 1 to n − 1
3.       do min := i
4.            for j := i + 1 to n
5.                 do if A_min > A_j
6.                       then min := j
7.            A_i ⇔ A_min
10. return A
```

Bonyolultság szintén $\Theta(n^2)$**.**

Példa

Számláló rendezés (Counting sort)

Különbözõ elemek esetében minden elemre megszámoljuk a kisebb elemeket. Ha egy elemnél $k-1$ elem kisebb, akkor õ a k-adik helyre kerül.

SZÁMLÁLÓ-RENDEZÉS-1(A)

```
1.  n := hossz[A]
2.  for i := 1 to n
3.       do k := 1
4.          for j := 1 to n
5.               do if A_i > A_j
6.                    then k := k + 1
7.          B_k := A_i
8.  return B
```

Bonyolultság szintén $\Theta(n^2)$.

Ha az elemek 0 és n közötti egész számok, akkor meg lehet valósítani $\Theta(n)$ idõben.

Ha az elemek nem mind különbözõek, akkor helyet keresünk az illetõ elemnek.

SZÁMLÁLÓ-RENDEZÉS-2(A)

```
1.  n := hossz[A]
2.  for i := 1 to n
3.       do B_i := 0
4.  for i := 1 to n
5.       do k := 1
6.            for j := 1 to n
7.                 do if A_i > A_j
8.                         then k := k + 1
9.            while B_k ≠ 0
10.                do k := k + 1
11.           B_k := A_i
12. return B
```

Bonyolultság szintén $\Theta(n^2)$**.**

Beszúró rendezés (Insertion sort)

BESZÚRÓ-RENDEZÉS(A)

```
1. n := hossz[A]
2. for i := 2 to n
3.     do k := A_i
4.        j := i − 1
5.        while (j > 0) és (A_j > k)
6.              do A_{j+1} := A_j
7.                 j := j − 1
8.        A_{j+1} := k
9. return A
```

Bonyolultság legrosszabb esetben szintén $\Theta(n^2)$.

Példa

Shell-rendezés (Shell sort)

$H : h_t > h_{t-1} > \ldots > h_2 > h_1 = 1$ **növekmények**

SHELL-RENDEZÉS(A, h)

```
1.  n := hossz[A]
2.  t := hossz[H]
3.  for s := t downto 1
4.      do p := h_s
5.          for i := p + 1 to n
6.              do k := A_i
7.                 j := i − p
7.                 while (j > 0)  és (A_j > k)
8.                       do A_{j+p} := A_j
9.                          j := j − p
10.                A_{j+p} := k
11. return A
```

Bonyolultság legrosszabb esetben $\Theta(n^2)$, átlagesetben sokkal jobb.

Példa

Összefésülõ rendezés (Merge sort) Neumann János, 1945

MERGESORT(A, b, j)

1. **if** $b < j$ ▷ b **bal index,** j **jobb index**

2. **then** $k := \left\lfloor \frac{b+j}{2} \right\rfloor$

3. MERGESORT(A, b, k)

4. MERGESORT$(A, k+1, j)$

5. ÖSSZEFÉSÜL(A, b, k, j)

7. **return** A

ÖSSZEFÉSÜL(A, b, k, j)

1. $n_1 := k - b + 1$
2. $n_2 := j - k$
3. **for** $p := 1$ **to** n_1
4. **do** $L_p := A_{b+p-1}$
5. **for** $r := 1$ **to** n_2
6. **do** $R_r := A_{k+r}$
7. $L_{n_1+1} := \infty$ ▷ **strázsa**
8. $R_{n_2+1} := \infty$ ▷ **strázsa**
9. $p := 1$
10. $r := 1$
11. **for** $i := b$ **to** j
12. **do if** $L_p \leq R_r$
13. **then** $A_i := L_p$
14. $p := p + 1$
15. **else** $A_i := R_r$
16. $r := r + 1$

Az eljárás hívása: Mergesort($A, 1, n$**),**
ha $A = (A_1, A_2, \ldots, A_n)$**.**

Bonyolultság: $\Theta(n \log n)$**.**

Példa

Kupacrendezés (Heapsort)

Kupacot használunk és a KUPACOT_ÉPÍT **valamint a** KUPACOL **eljárást.**

A bemeneti $A_1, A_2, \ldots, A_n$ **sorozatra meghívjuk a** KUPACOT_ÉPÍT **eljárást, amely kupactulajdonságúvá változtatja a kupacot.**

KUPACOL(A, i)

1. $b := \text{BAL}(i)$
2. $j := \text{JOBB}(i)$
3. **if** $(b \leq kupacméret[A])$ **és** $(A_b > A_i)$
4. **then** $max := b$
5. **else** $max := i$
6. **if** $(j \leq kupacméret[A])$ **és** $(A_j > A_{max})$
7. **then** $max := j$
8. **if** $max \neq i$
9. **then** $A_i \Leftrightarrow A_{max}$
10. KUPACOL(A, max)

KUPACOT-ÉPÍT(A)

1. $kupacméret[A] := hossz[A]$
2. **for** $i := \dfrac{hossz[A]}{2}$ **downto 1**
3. **do** KUPACOL(A, i)

A kupacrenzedés a gyökérelemet (amely a legnagyobb) az utolsó helyre teszi (felcseréléssel), kiveszi azt a kupacból, a többi elemre helyreállítja a kupactulajdonságot, és ugyanúgy folytatja.

KUPACRENDEZÉS(A)

0. KUPACOT-ÉPÍT(A)
1. $kupacméret[A] := hossz[A]$ ▷ **a kupacméret változik**
2. $n := hossz[A]$
3. **for** $i := n$ **downto** **2**
4. **do** $A_1 \Leftrightarrow A_i$
5. $kupacméret[A] := kupacméret[A] - 1$
6. KUPACOL(A,1)
7. **return** A

Bonyolultság: $\Theta(n \log n)$**.**

Példa

Gyorsrendezés (Quicksort)

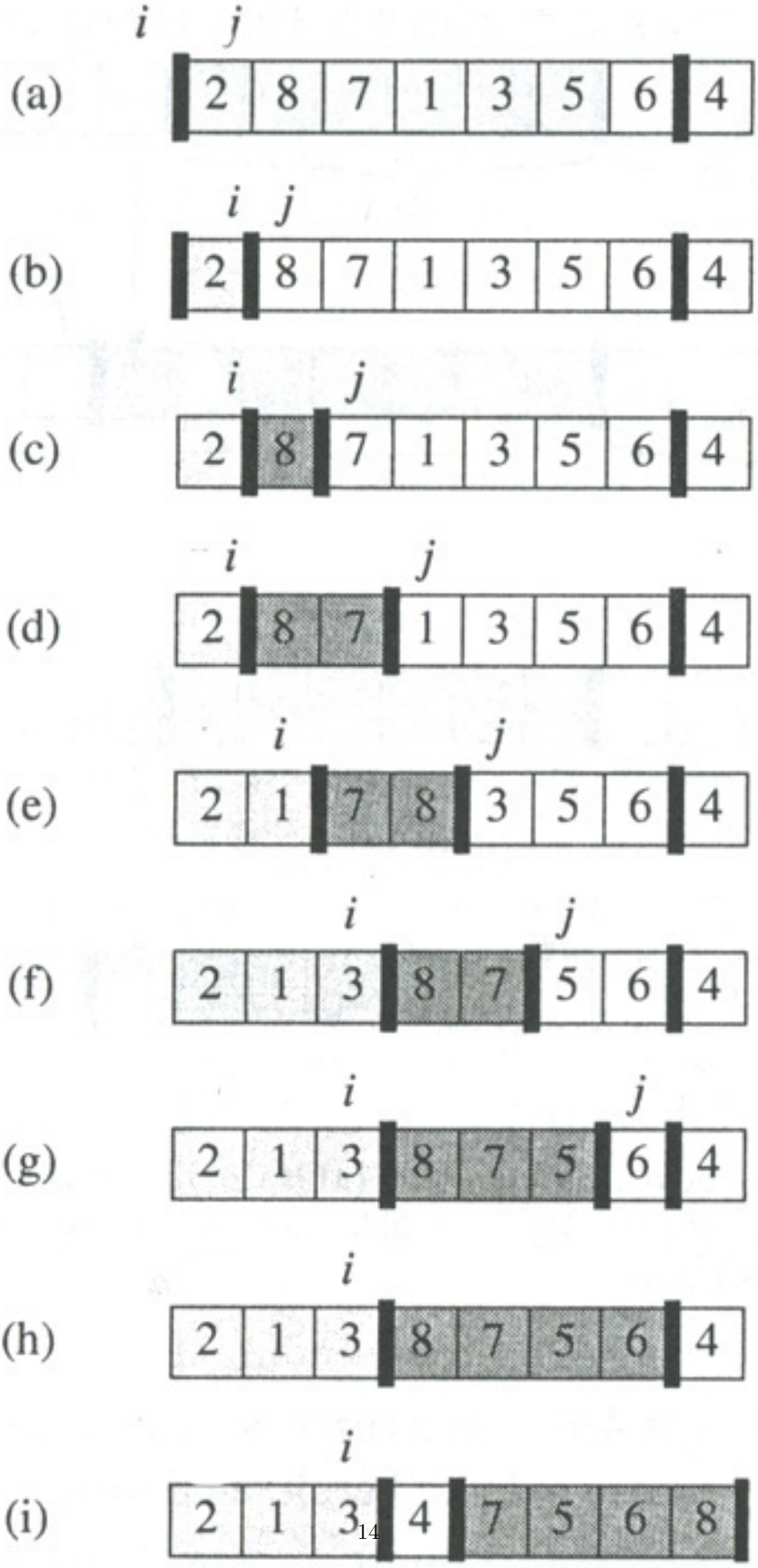
GYORSRENDEZÉS(A, b, j)

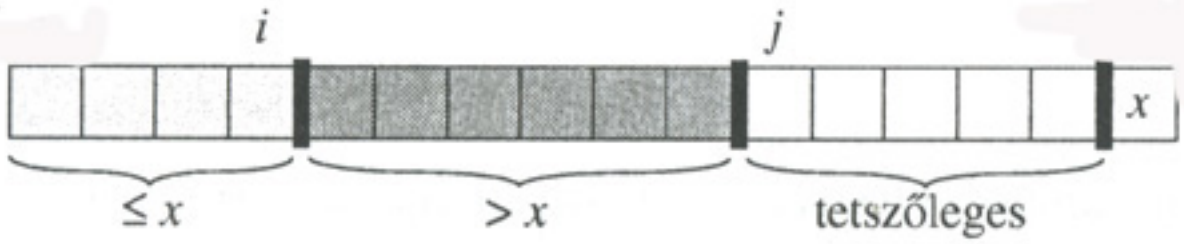
```
1. if b < j
2.     then k := FELOSZT(A, b, j)
3.          GYORSRENDEZÉS(A, b, k − 1)
4.          GYORSRENDEZÉS(A, k + 1, j)
5.     else return A
```

FELOSZT$(A, bal, jobb)$

```
1. x := A_jobb
2. i := bal − 1
3. for j := bal to jobb − 1
4.     do if A_j ≤ x
5.          then i := i + 1
6.               A_i ⇔ A_j
7. A_{i+1} ⇔ A_jobb
8. return i + 1
```

Az eljárás hívása: GYORSRENDEZÉS$(A, 1, n)$


(a)
i
j
2 8 7 1 3 5 6 4
(b)
i
j
2 8 7 1 3 5 6 4
(c)
i
j
2 8 7 1 3 5 6 4
(d)
i
j
2 8 7 1 3 5 6 4
(e)
i
j
2 1 7 8 3 5 6 4
(f)
i
j
2 1 3 8 7 5 6 4
(g)
i
j
2 1 3 8 7 5 6 4
(h)
i
2 1 3 8 7 5 6 4
(i)
i
2 1 3 4 7 5 6 8



C. A. Hoare felosztó algoritmusa:

FELOSZT2(A, b, j)

1. $x := A_b$
2. $i := b - 1$
3. $j := j + 1$
4. **while** IGAZ
5. **repeat** $j := j - 1$
6. **until** $A_j \leq x$
7. **repeat** $i := i + 1$
8. **until** $A_i \geq x$
9. **if** $i < j$
10. **then** $A_i \Leftrightarrow A_j$
11. **else** **return** j

Nem rekurzív változat

Az $A_b, \ldots, A_j$ résztömbök indexeit egy veremben õrizzük $[b, j]$ formában, kezdetben a verem üres.

push($[b, j]$) betesszük a verembe,
pop($[b, j]$) kiveszzük a veremből

NEMREKURZÍV-GYORSRENDEZÉS(A)

```
1.  n := hossz[A]
2.  push([1, n])
3.  while a verem nem üres
4.        do pop([b, j])
5.           while b < j
6.                 do k := FELOSZT(A, b, j)
7.                    push([k + 1, j])
8.                    j := k − 1
9.  return A
```

Példa

Bonyolultság legrosszabb esetben: $\Theta(n^2)$, átlagesetben $\Theta(n \log n)$.

Edényrendezés (Bucket sort)

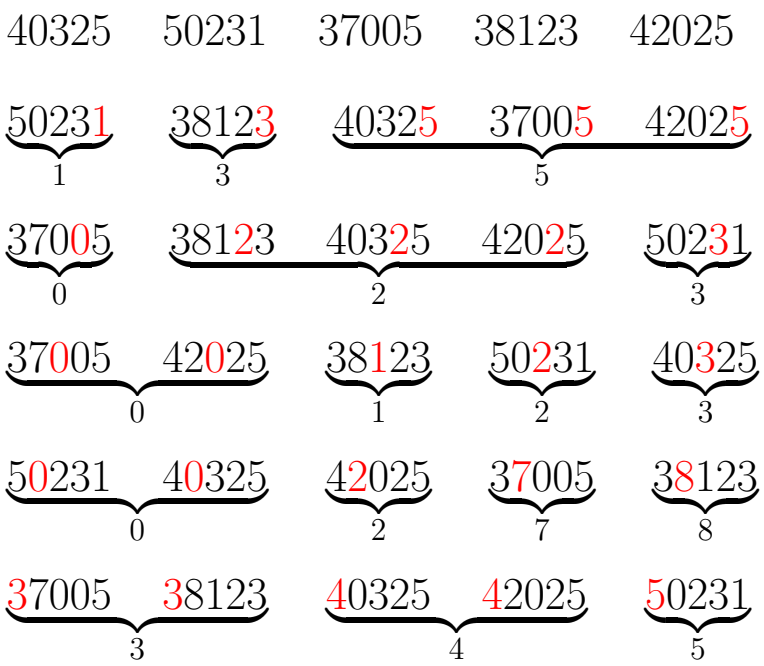
Az elemeket egyenletes elosztásúaknak tekintjük. Az elemeket osztályokba (edényekbe) osztjuk. Lineáris bonyolultságú.

Példa

Számjegyes rendezés (Radix sort)

A legkisebb helyértékû számjeggyel kezdjük a csoportosítást, majd haladunk balra. Lineáris bonyolultságú.

Példa



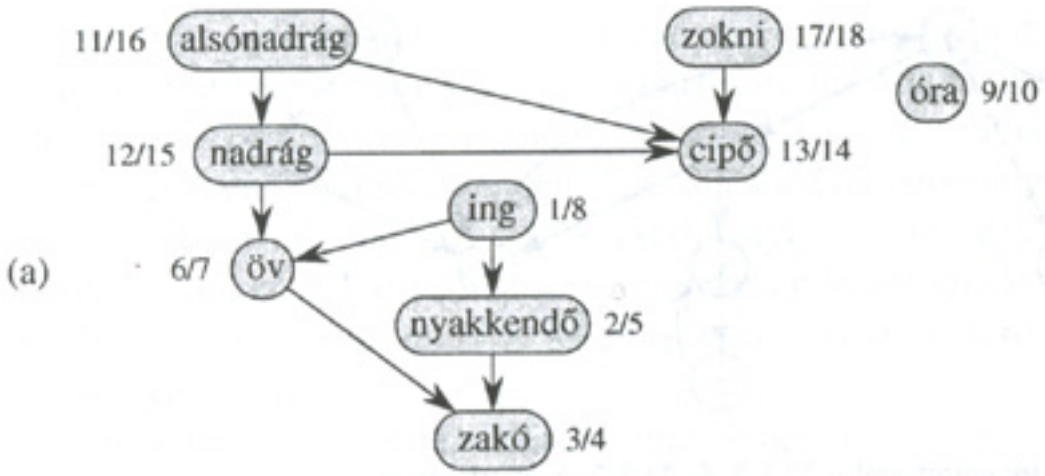
algoritmus	átlag	legrosszabb
buborékrendezés (Bubble sort)	$\Theta(n^2)$	$\Theta(n^2)$
kiválasztó rendezés (Selection sort)	$\Theta(n^2)$	$\Theta(n^2)$
számláló rendezés (Counting sort)	$\Theta(n^2)$	$\Theta(n^2)$
beszúró rendezés (Insertion sort)	$\Theta(n^2)$	$\Theta(n^2)$
Shell-rendezés (Shell sort)		$\Theta(n^2)$
gyorsrendezés (Quicksort)	$\Theta(n \log n)$	$\Theta(n^2)$
összefésülõ rendezés (Merge sort)	$\Theta(n \log n)$	$\Theta(n \log n)$
kupacrendezés (Heapsort)	$\Theta(n \log n)$	$\Theta(n \log n)$

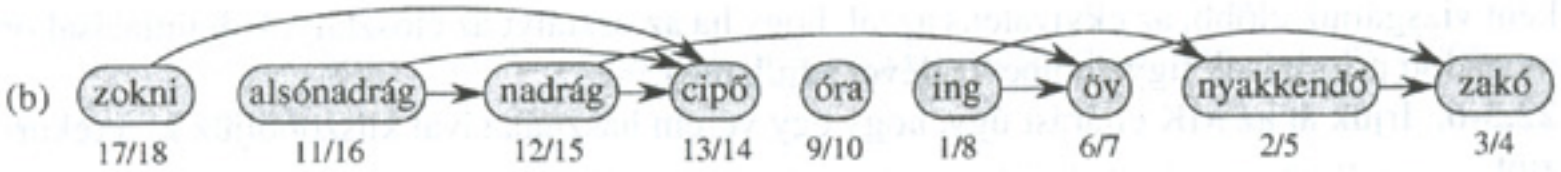
Topologikus rendezés

mélységi bejárás alkalmazása

```
MK(G)
1  for minden u ∈ V[G] csúcsra
2      do szín[u] ← FEHÉR
3          π[u] ← NIL
4  idő ← 0
5  for minden u ∈ V[G] csúcsra
6      do if szín[u] = FEHÉR
7          then MK-BEJÁR(u)

MK-BEJÁR(u)
1  szín[u] ← SZÜRKE                ▷ Most érjük el a fehér u csúcsot.
2  idő ← idő +1
3  d[u] ← idő
4  for minden v ∈ Adj[u] csúcsra   ▷ (u, v) él vizsgálata.
5      do if szín[v] = FEHÉR
6          then π[v] ← u
7               MK-BEJÁR(v)
8  szín[u] ← FEKETE                ▷ u fekete, mert elhagytuk.
9  f[u] ← idő ← idő +1
```





TOPOLOGIKUS-RENDEZÉS(G)

1 MK(G) hívása; minden v csúcsra meghatározzuk az $f[v]$ elhagyási időt
2 az egyes csúcsok elhagyásakor szúrjuk be azokat egy láncolt lista elejére
3 **return** a csúcsok láncolt listája

Külsõ rendezés (Knuth, 3. kötet)

két lépésbõl áll:

futamok elõállítása (angolul run)

futamok összefésülése

két módszer:

1) többfázisú öszefésülés

2) kaszkád összefésülés

Többfázisú öszefésülés

3 szalagot használunk: T_1, T_2, T_3

1. **Osszuk szét a kezdeti futamokat felváltva T_1-en és T_2-n.**
2. **A T_1 és T_2 szalagokról fésüljük össze a futamokat T_3-ra. Ha T_3 egyetlen futamot tartalmaz, álljunk meg.**
3. **Másoljuk a T_3-on levõ futamokat felváltva T_1-re és T_2-re, majd folytassuk a 2. lépéssel**

1. fázis	1^8	1^5	–
2. fázis	1^3	–	2^5
3. fázis	–	3^3	2^2
4. fázis	5^2	3^1	–
5. fázis	5^1	–	8^1
6. fázis	–	13^1	–

k^n **jelentése:** n **darab** k **hosszúságú futam (**k **hosszúság: az eredeti futam** k**-szorosa)**

Kaszkád összefésülés

	T1	T2	T3	T4	T5	T6	Feldolgozott kezdeti futamok
1. menet	1^{55}	1^{50}	1^{41}	1^{29}	1^{15}	–	190
2. menet	–	$1^{5}*$	2^{9}	3^{12}	4^{14}	5^{15}	190
3. menet	15^{5}	14^{4}	12^{3}	9^{2}	$5^{1}*$	–	190
4. menet	–	$15^{1}*$	29^{1}	41^{1}	50^{1}	55^{1}	190
5. menet	190^{1}	–	–	–	–	–	190

http://www.youtube.com/watch?v=Ns4TPTC8whw

http://www.youtube.com/watch?v=ROalU379l3U

http://www.youtube.com/watch?v=CmPA7zE8mx0

http://www.youtube.com/watch?v=lyZQPjUT5B4

http://www.youtube.com/watch?v=XaqR3G_NVoo