

# Sorting Algorithms

1. Selection
2. Bubble
3. Insertion
4. Merge
5. Quick
6. *Shell*

# Definisi

- Metode ini disebut juga dengan metode pertambahan menurun (*diminishing increment sort*). Metode ini dikembangkan oleh Donald L. Shell pada tahun 1959, sehingga sering disebut dengan Metode Shell Sort.
- Metode ini mengurutkan data dengan cara membandingkan suatu data dengan data lain yang memiliki jarak tertentu – sehingga membentuk sebuah sub-list-, kemudian dilakukan penukaran bila diperlukan

# Definisi

- Jarak yang dipakai didasarkan pada *increment value* atau *sequence number k*
- Misalnya Sequence number yang dipakai adalah 5,3,1. Tidak ada pembuktian di sini bahwa bilangan-bilangan tersebut adalah sequence number terbaik
- Setiap sub-list berisi setiap elemen ke-k dari kumpulan elemen yang asli

# Definisi

- Contoh: Jika  $k = 5$  maka sub-list nya adalah sebagai berikut :
  - $s[0]$   $s[5]$   $s[10]$  ...
  - $s[1]$   $s[6]$   $s[11]$  ...
  - $s[2]$   $s[7]$   $s[12]$  ...
  - dst
- Begitu juga jika  $k = 3$  maka sub-list nya adalah:
  - $s[0]$   $s[3]$   $s[6]$  ...
  - $s[1]$   $s[4]$   $s[7]$  ...
  - dst

# Proses Shell Sort

- Buatlah sub-list yang didasarkan pada jarak (Sequence number) yang dipilih
- Urutkan masing-masing sub-list tersebut
- Gabungkan seluruh sub-list

*Let's see this algorithm in action*

# Proses Shell Sort

- Urutkan sekumpulan elemen di bawah ini ,  
misalnya diberikan sequence number : 5, 3, 1

30	62	53	42	17	97	91	38
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]

# Proses Shell Sort k=5

**30    62    53    42    17    97    91    38**

Step 1: Buat sub list  $k = 5$

S[0] S[5]

S[1] S[6]

S[2] S[7]

S[3]



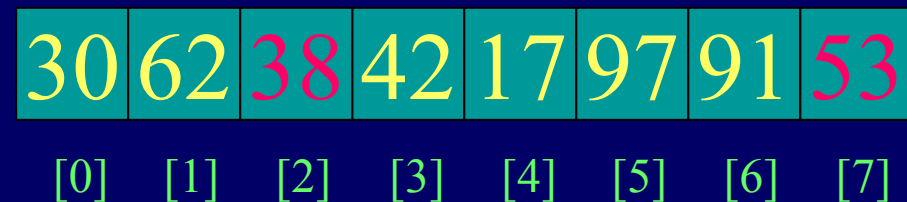
Step 2 - 3: Urutkan sub list & gabungkan

S[0] < S[5]    30, 97    OK

S[1] < S[6]    62, 91    OK

S[2] > S[7]    53, 38    not OK.

Swap them    38, 53



# Proses Shell Sort utk k=3

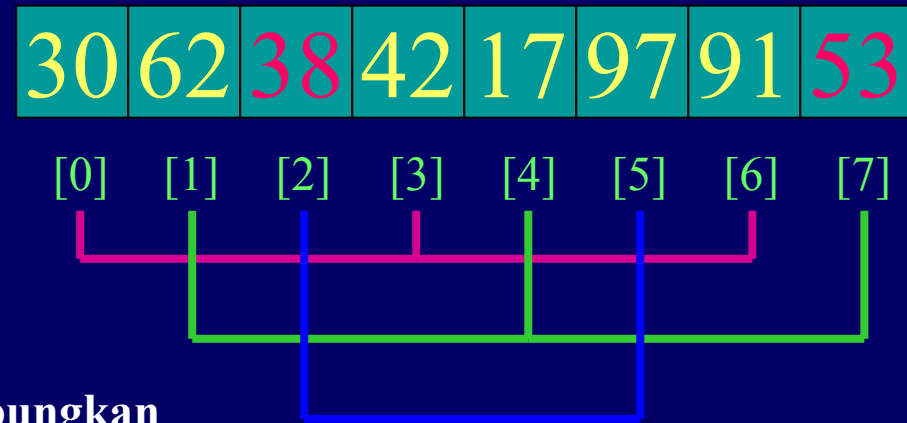
**30      62      53      42      17      97      91      38**

Step 1: Buat sub list  $k = 3$

S[0] S[3] S[6]

S[1] S[4] S[7]

S[2] S[5]



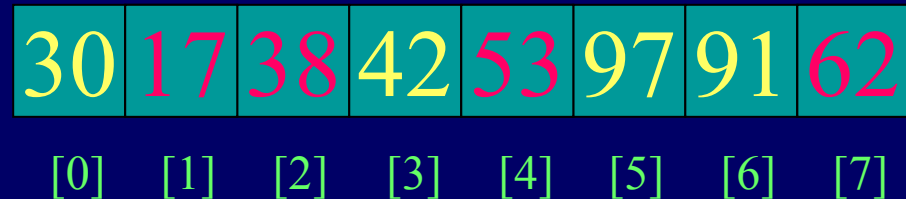
Step 2 - 3: Urutkan sub list & gabungkan

S[0] S[3] S[6] 30, 42, 91 OK

S[1] S[4] S[7] 62, 17, 53 not OK

SORT them 17, 53, 62

S[2] S[5] 38, 97 OK





# Shell Sort Process k=1

**30 62 53 42 17 97 91 38**

Step 1: Buat sub list  $k=1$

S[0] S[1] S[2] S[3] S[4] S[5] S[6] S[7]

30	17	38	42	53	97	91	62
----	----	----	----	----	----	----	----

[0] [1] [2] [3] [4] [5] [6] [7]



Step 2 - 3: Urutkan sub list & gabungkan

Sorting akan seperti insertion sort

17	30	38	42	53	62	91	97
----	----	----	----	----	----	----	----

[0] [1] [2] [3] [4] [5] [6] [7]

**DONE**

# Pemilihan Sequence Number

- Disarankan jarak mula-mula dari data yang akan dibandingkan adalah:  $N / 2$ .
- Pada proses berikutnya, digunakan jarak  $(N / 2) / 2$  atau  $N / 4$ .
- Pada proses berikutnya, digunakan jarak  $(N / 4) / 2$  atau  $N / 8$ .
- Demikian seterusnya sampai jarak yang digunakan adalah 1.

# Urutan prosesnya...

- Untuk jarak  $N/2$  :
  - Data pertama ( $i=0$ ) dibandingkan dengan data dengan jarak  $N / 2$ . Apabila data pertama lebih besar dari data ke  $N / 2$  tersebut maka kedua data tersebut ditukar.
  - Kemudian data kedua ( $i=1$ ) dibandingkan dengan jarak yang sama yaitu  $N / 2$  = elemen ke- $(i+N/2)$
  - Demikian seterusnya sampai seluruh data dibandingkan sehingga semua data ke- $i$  selalu lebih kecil daripada data ke- $(i + N / 2)$ .
- Ulangi langkah-langkah di atas untuk jarak =  $N / 4$  → lakukan perbandingan dan pengurutan sehingga semua data ke- $i$  lebih kecil daripada data ke- $(i + N / 4)$ .
- Ulangi langkah-langkah di atas untuk jarak =  $N / 8$  → lakukan perbandingan dan pengurutan sehingga semua data ke- $i$  lebih kecil daripada data ke- $(i + N / 8)$ .
- Demikian seterusnya sampai jarak yang digunakan adalah 1 atau data sudah terurut (`did_swap = false`)

# Algoritma Metode Shell Sort

1.  $\text{Jarak} \leftarrow N$
2. Selama  $\text{Jarak} > 1$  kerjakan baris 3 sampai dengan 12
3.  $\text{Jarak} \leftarrow \text{Jarak} / 2.$
4.  $\text{did\_swap} \leftarrow \text{true}$
5. Kerjakan baris 6 sampai dengan 12 selama  $\text{did\_swap} = \text{true}$
6.  $\text{did\_swap} \leftarrow \text{false}$
7.  $i \leftarrow 0$
8. Selama  $i < (N - \text{Jarak})$  kerjakan baris 9 sampai dengan 12
9. Jika  $\text{Data}[i] > \text{Data}[i + \text{Jarak}]$  kerjakan baris 10 dan 11
10.  $\text{tukar}(\text{Data}[i], \text{Data}[i + \text{Jarak}])$
11.  $\text{did\_swap} \leftarrow \text{true}$
12.  $i \leftarrow i + 1$

# Analisis Metode Shell Sort

- Running time dari metode Shell Sort bergantung pada pemilihan sequence number-nya.
- Disarankan untuk memilih sequence number dimulai dari  $N/2$ , kemudian membaginya lagi dengan 2, seterusnya hingga mencapai 1.
- Shell sort menggunakan 3 nested loop, untuk merepresentasikan sebuah pengembangan yang substansial terhadap metode insertion sort

# Pembandingan Running time (millisecond) antara insertion and Shell

N	insertion	Shellsort
1000	122	11
2000	483	26
4000	1936	61
8000	7950	153
16000	32560	358

Ref: Mark Allan Wiess  
(Florida International University)

# Shellsort Analysis

Ref: Mark Allan Wiess