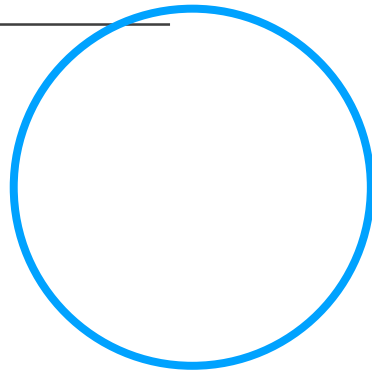


# Algoritma Pengurutan (Sorting)

---

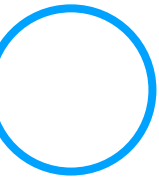
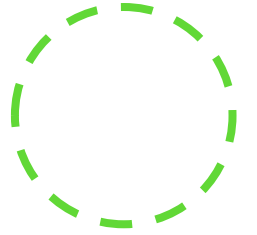
TIM DOSEN





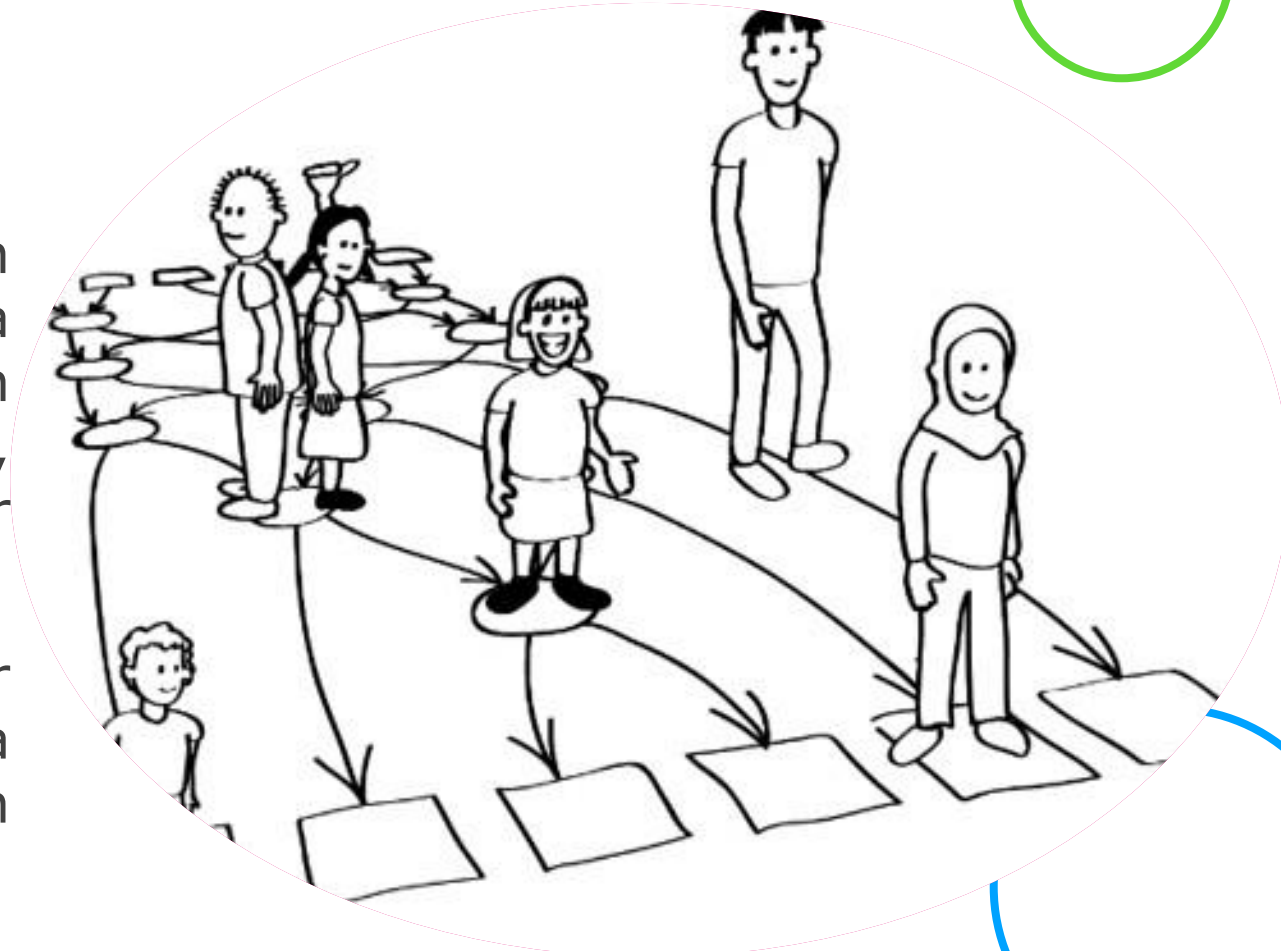
# OUTLINE

- a. Definisi Pengurutan
- b. Tipe Urutan Proses Sorting
- c. Pengenalan Metode Sorting



# DEFINISI PENGURUTAN

- Pengurutan (Sorting) adalah proses menyusun kembali data yang sebelumnya telah disusun dengan suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu.
- Pengurutan data dalam struktur data sangat penting untuk data yang beripe data numerik ataupun karakter.
- Pengurutan dapat dilakukan secara ascending (urut naik) dan descending (urut turun).





# TIPE URUTAN PROSES SORTING


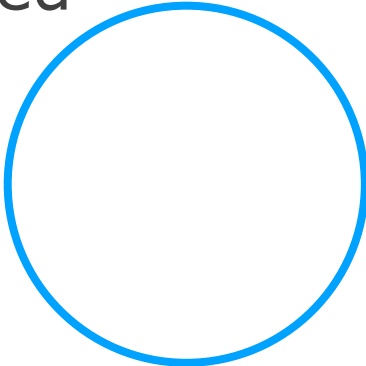

- a. Urutan Naik (*Ascending*)
- b. Urutan Turun (*Descending*)





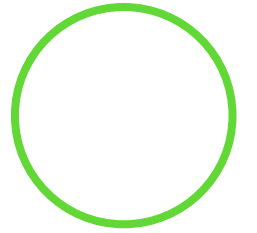
# PENGENALAN METODE SORTING



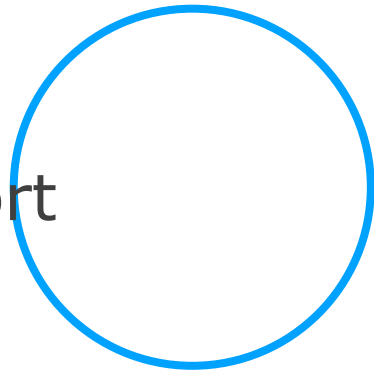
- Penerapan pengurutan pada penggunaan fungsi tukar dengan membanding nilai kemudian menukar posisi lebih kecil atau lebih besar.
  - Untuk melakukan proses pengurutan tersebut dapat digunakan berbagai macam cara/metode :
    - a. Pengurutan berdasarkan perbandingan (comparison-based sorting) :
      - Bubble sort, exchange sort
    - b. Pengurutan berdasarkan prioritas (priority queue sorting method):
      - Selection sort, heap sort (menggunakan tree)
- 
- 
- 



# PENGENALAN METODE SORTING- LANJUT

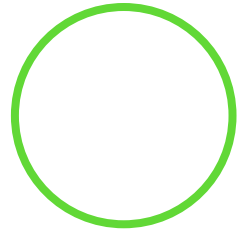


- c. Pengurutan berdasarkan penyisipan dan penjagaan terurut (insert and keep sorted method):
  - Insertion sort, tree sort
- d. Pengurutan berdasarkan pembagian dan penguasaan (divide and conquer method):
  - Quick sort, merge sort
- e. Pengurutan berkurang menurun (diminishing increment sort method):
  - Shell sort (pengembangan insertion)





# Pengenalan Metode Sorting- Lanjut



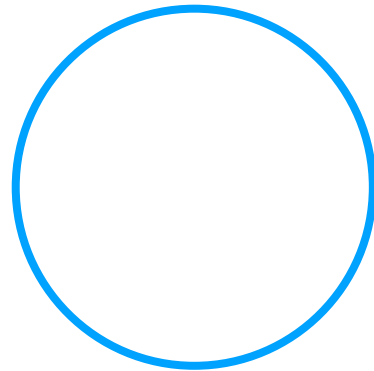
## ➤ Contoh:

- Data Acak : 5 6 8 1 3 25 10
- Ascending : 1 3 5 6 8 10 25
- Descending : 25 10 8 6 5 3 1

## ➤ Contoh penerapan pengurutan pada fungsi tukar (swapping):

Fungsi untuk Tukar 2 Buah Data (by reference):

```
void tukar(int *a,int *b){  
    int t=*a;  
    *a=*b;  
    *b=t;  
}
```





# METODE SORTING



INSERTION SORT  
(METODE PENYISIPAN)



BUBBLE SORT (METODE  
GELEMBUNG)



QUICK SORT  
(METODE QUICK)



Heap Sort  
TIDAK DIBAHAS



SELECTION SORT  
(METODE SELEKSI)



SHELL SORT  
(METODE SHELL)



MERGE SORT (METODE  
PENGGABUNGAN)






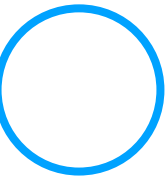
# Metode Pengurutan (*Sorting*)



# INSERTION SORT (METODE PENYISIPAN) -1

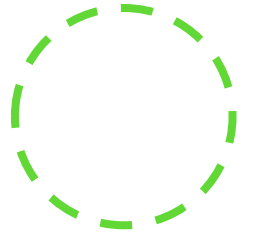


- Mirip dengan cara orang mengurutkan kartu, selembat demi selembat kartu diambil dan disisipkan (insert) ke tempat yang seharusnya.
  - Pengurutan dimulai dari data ke-2 sampai dengan data terakhir, jika ditemukan data yang lebih kecil, maka akan ditempatkan (diinsert) diposisi yang seharusnya.
  - Pada penyisipan elemen, maka elemen-elemen lain akan bergeser ke belakang.
  - Pengurutan dilakukan dengan cara membandingkan data ke-i (dimana i dimulai dari data ke-2 sampai dengan data terakhir) dengan data berikutnya.
  - Jika ditemukan data yang lebih kecil maka data tersebut disisipkan ke depan sesuai dengan posisi yang seharusnya.
- 

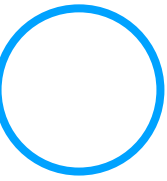




# ALGORITMA INSERTION SORT (METODE PENGURUTAN PENYISIPAN)



1.  $i \leftarrow 2$
2. Selama ( $i \leq N$ ) kerjakan baris 3 s.d. 10
3.  $x \leftarrow \text{data}[i]$
4.  $\text{Data}[0] \leftarrow x$
5.  $j \leftarrow i - 1$
6. Selama ( $x < \text{data}[j]$ ) kerjakan baris 7 s.d.8
7.  $\text{data}[j+1] > \text{data}[j]$
8.  $j \leftarrow j - 1$
9.  $\text{data}[j+1] \leftarrow x$
10.  $i \leftarrow i + 1$



# INSERTION SORT - ASCENDING

MISAL : A = [ 23, 17, 14, 6, 13, 10, 1, 5, 7 ] → data acak

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Awal	23	17	14	6	13	10	1	5	7
i=2	23	17	14	6	13	10	1	5	7
i=3	17	23	14	6	13	10	1	5	7
i=4	14	17	23	6	13	10	1	5	7
i=5	6	14	17	23	13	10	1	5	7
i=6	6	13	14	17	23	10	1	5	7
i=7	6	10	13	14	17	23	1	5	7
i=8	1	6	10	13	14	17	23	5	7
i=9	1	5	6	10	13	14	17	23	7
Akhir	1	5	6	7	10	13	14	17	23

# INSERTION SORT - ASCENDING

MISAL : A = [ 22, 10, 15, 3, 8, 2 ] → data acak

## Proses 1

0	1	2	3	4	5
22	10	15	3	8	2

Temp	Cek	Geser
10	Temp < 22?	Data ke-0 ke posisi 1

Temp menempati posisi ke -0

0	1	2	3	4	5
10	22	15	3	8	2

## Proses 2

0	1	2	3	4	5
10	22	15	3	8	2

Temp	Cek	Geser
15	Temp < 22	Data ke-1 ke posisi 2
15	Temp > 10	-

Temp menempati posisi ke-1

0	1	2	3	4	5
10	15	22	3	8	2

## Proses 3

0	1	2	3	4	5
10	15	22	3	8	2

Temp	Cek	Geser
3	Temp < 22	Data ke-2 ke posisi 3
3	Temp < 15	Data ke-1 ke posisi 2
3	Temp < 10	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

0	1	2	3	4	5
3	10	15	22	8	2

## Proses 4

0	1	2	3	4	5
3	10	15	22	8	2

Temp	Cek	Geser
8	Temp < 22	Data ke-3 ke posisi 4
8	Temp < 15	Data ke-2 ke posisi 3
8	Temp < 10	Data ke-1 ke posisi 2
8	Temp > 3	-

Temp menempati posisi ke-1

0	1	2	3	4	5
3	8	10	15	22	2

# LANJUT

## Proses 5

0	1	2	3	4	5
3	8	10	15	22	2

Temp	Cek	Geser
2	Temp<22	Data ke-4 ke posisi 5
2	Temp<15	Data ke-3 ke posisi 4
2	Temp<10	Data ke-2 ke posisi 3
2	Temp<8	Data ke-1 ke posisi 2
2	Temp<3	Data ke-0 ke posisi 1

Temp menempati posisi ke-0

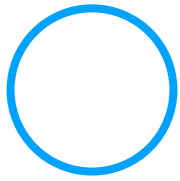

0	1	2	3	4	5
2	3	8	10	15	22



# PROGRAM INSERTION SORT




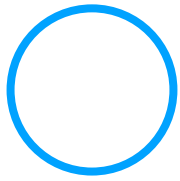
```
void insertion_sort(int data[]){  
    int temp;  
    for(int i=1;i<n;i++){  
        temp = data[i];  
        j = i - 1;  
        while(data[j]>temp && j>=0){  
            data[j+1] = data[j];  
            j--;  
        }  
        data[j+1] = temp;  
    }  
}
```





# SELECTION SORT (METODE SELEKSI) - 2

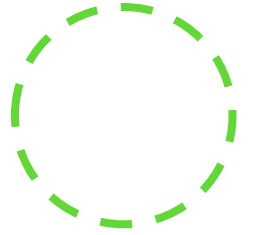



- Merupakan **kombinasi antara sorting dan searching**.
  - Membandingkan elemen yang sekarang dengan elemen yang berikutnya sampai dengan elemen yang terakhir.
  - Jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka dicatat posisinya dan kemudian ditukarkan.
  - Untuk setiap proses, akan dicari elemen-elemen yang belum diurutkan yang memiliki nilai terkecil atau terbesar akan dipertukarkan ke posisi yang tepat di dalam array.
  - Misalnya untuk putaran pertama, akan dicari data dengan nilai terkecil dan data ini akan ditempatkan di indeks terkecil (`data[0]`), pada putaran kedua akan dicari data kedua terkecil, dan akan ditempatkan di indeks kedua (`data[1]`).
  - Selama proses, perbandingan dan pengubahan hanya dilakukan pada indeks perbandingan saja, pertukaran data secara fisik terjadi pada akhir proses.
- 
- 

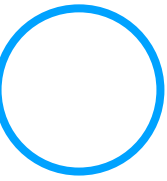




# ALGORITMA SELECTION SORT (METODE PENGURUTAN SELEKSI)



1.  $i \leftarrow 1$
  2. Selama ( $i \leq N-1$ ) kerjakan baris 3 s.d.9
  3.  $k \leftarrow i$
  4.  $j \leftarrow i + 1$
  5. Selama ( $j \leq N$ ) kerjakan baris 6 s.d. 7
  6. Jika ( $\text{Data}[k] > \text{Data}[j]$ ) maka  $k \leftarrow j$
  7.  $j \leftarrow j + 1$
  8. Tukar Data  $[i]$  dengan Data  $[k]$
  9.  $i \leftarrow i + 1$
- 



# SELECTION SORT - ASCENDING

MISAL : A = [ 32, 75, 69, 58, 21, 40 ] → data acak

## Proses 1

0	1	2	3	4	5
32	75	69	58	21	40

Pembanding	Posisi
32 < 75	0
32 < 69	0
32 < 58	0
32 > 21 (tukar idx)	4
21 < 40	4

Tukar data ke-0 (32) dengan data ke-4 (21)

0	1	2	3	4	5
21	75	69	58	32	40

## Proses 2

0	1	2	3	4	5
21	75	69	58	32	40

Pembanding	Posisi
75 > 69 (tukar idx)	2
69 > 58 (tukar idx)	3
58 > 32 (tukar idx)	4
32 < 40	4

Tukar data ke-1 (75) dengan data ke-4 (32)

0	1	2	3	4	5
21	32	69	58	75	40

## Proses 3

0	1	2	3	4	5
21	32	69	58	75	40

Pembanding	Posisi
69 > 58 (tukar idx)	3
58 < 75	3
58 > 40	5

Tukar data ke-2 (69) dengan data ke-5 (40)

0	1	2	3	4	5
21	32	40	58	75	69

# SELECTION SORT - ASCENDING

## Proses 4

0	1	2	3	4	5
21	32	40	<b>58</b>	75	69

Pembanding	Posisi
$58 < 75$	3
$58 < 69$	3

Tukar data ke-3 (**58**) dengan data ke-3 (**58**)

0	1	2	3	4	5
21	32	40	58	75	69

## Proses 5

0	1	2	3	4	5
21	32	40	58	<b>75</b>	69

Pembanding	Posisi
$75 > 69$	5

Tukar data ke-4 (**75**) dengan data ke-5 (**69**)

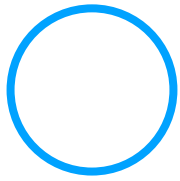

0	1	2	3	4	5
21	32	40	58	69	75



# PROGRAM SELECTION SORT

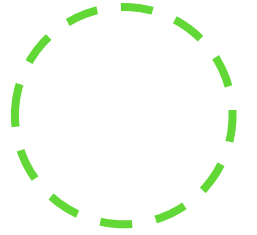
## Prosedur Selection Sort

```
void selection_sort(int data[]){  
    for(int i=0;i<n-1;i++){  
        pos = i;  
        for(int j=i+1;j<n;j++){  
            if(data[j] < data[pos]) pos = j; //ascending  
        }  
        if(pos != i) tukar(&data[pos], &data[i]);  
    }  
}
```

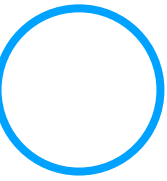




# BUBBLE SORT (METODE GELEMBUNG) - 3


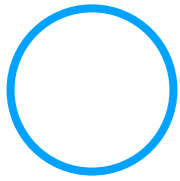


- Metode sorting termudah
- Diberi nama “Bubble” karena proses pengurutan secara berangsur-angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda.
- Bubble Sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya.




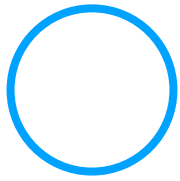


# BUBBLE/EXCHANGE SORT

- Pengurutan **Ascending** :Jika elemen sekarang **lebih besar** dari elemen berikutnya maka kedua elemen tersebut ditukar.
  - Pengurutan **Descending**: Jika elemen sekarang **lebih kecil** dari elemen berikutnya, maka kedua elemen tersebut ditukar.
  - Algoritma ini seolah-olah menggeser satu per satu elemen dari kanan ke kiri atau kiri ke kanan, tergantung jenis pengurutannya, asc atau desc.
  - Ketika satu proses telah selesai, maka bubble sort akan mengulangi proses, demikian seterusnya sampai dengan iterasi sebanyak  $n-1$ .
  - Kapan berhentinya? Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan, serta tercapai perurutan yang telah diinginkan.
- 
- 


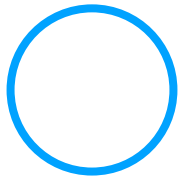


# BUBBLE/EXCHANGE SORT - LANJUT

- Membandingkan elemen yang sekarang dengan elemen yang berikutnya, jika elemen sekarang  $>$  elemen berikutnya, maka ditukar.
  - Metode pengurutan gelembung (Bubble Sort), menginginkan larik terurut menaik, maka elemen larik yang berharga paling kecil, dipindahkan ke ujung kiri larik melalui proses pertukaran.
  - Proses metode ini dilakukan sebanyak  $N-1$  langkah(pass) dengan  $N$  adalah ukuran larik.
  - Pada akhir setiap langkah ke- $I$ , larik  $L[1..N]$  akan terdiri atas dua bagian yaitu bagian yang sudah terurut, yaitu  $L[1..I]$ , dan bagian yang belum terurut  $L[I+1..N]$ .
  - Sedangkan langkah terakhir, diperoleh larik  $L[1..N]$  yang terurut menaik.
- 
- 



# ALGORITMA BUBBLE/EXCHANGE SORT

1.  $i \leftarrow 2$
  2. Selama ( $i \leq N-1$ ) kerjakan baris 3 s.d. 7
  3.  $j \leftarrow N$
  4. Selama ( $j \geq i$ ) kerjakan baris 5 s.d. 7
  5. Jika ( $\text{data}[j-1] > \text{data}[j]$ ) maka tukar  $\text{data}[j-1]$  dengan  $\text{data}[j]$
  6.  $j \leftarrow j - 1$
  7.  $i \leftarrow i + 1$
- 
- 



# BUBBLE SORT - ASCENDING

A: [ 23,17,14,6,13,10,1,5,7]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Awal	23	17	14	6	13	10	1	5	7
i=2	23	17	14	6	13	10	1	5	7
j=9	23	17	14	6	13	10	1	5	7
j=8	23	17	14	6	13	10	1	5	7
j=7	23	17	14	6	13	1	10	5	7
j=6	23	17	14	6	1	13	10	5	7
j=5	23	17	14	1	6	13	10	5	7
j=4	23	17	1	14	6	13	10	5	7
j=3	23	1	17	14	6	13	10	5	7
J=2	1	23	17	14	6	13	10	5	7

# BUBBLE SORT - ASCENDING

A: [ 23,17,14,6,13,10,1,5,7]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
i=3	1	23	17	14	6	13	10	5	7
j=9	1	23	17	14	6	13	10	5	7
j=8	1	23	17	14	6	13	5	10	7
j=7	1	23	17	14	6	5	13	10	7
j=6	1	23	17	14	5	6	13	10	7
j=5	1	23	17	5	14	6	13	10	7
j=4	1	23	5	17	14	6	13	10	7
j=3	1	5	23	17	14	6	13	10	7

# BUBBLE SORT - ASCENDING

A: [ 23,17,14,6,13,10,1,5,7]

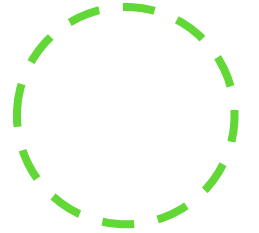
Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
i=4	1	5	23	17	14	6	13	10	7
j=9	1	5	23	17	14	6	13	7	10
j=8	1	5	23	17	14	6	7	13	10
j=7	1	5	23	17	14	6	7	13	10
j=6	1	5	23	17	6	14	7	13	10
j=5	1	5	23	6	17	14	7	13	10
j=4	1	5	6	23	17	14	7	13	10

# BUBBLE SORT - ASCENDING

A: [ 23,17,14,6,13,10,1,5,7]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
i=5	1	5	6	23	17	14	7	13	10
j=9	1	5	6	23	17	14	7	10	13
j=8	1	5	6	23	17	14	7	10	13
j=7	1	5	6	23	17	7	14	10	13
j=6	1	5	6	23	7	17	14	10	13
j=5	1	5	6	7	23	17	14	10	13

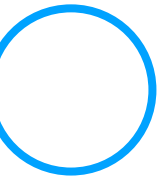
# BUBBLE SORT - ASCENDING



A: [ 23,17,14,6,13,10,1,5,7]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
i=6	1	5	6	7	23	17	14	10	13
j=9	1	5	6	7	23	17	14	10	13
j=8	1	5	6	7	23	17	10	14	13
j=7	1	5	6	7	23	10	17	14	13
j=6	1	5	6	7	10	23	17	14	13

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
i=7	1	5	6	7	10	23	17	14	13
j=9	1	5	6	7	10	23	17	13	14
j=8	1	5	6	7	10	23	13	17	14
j=7	1	5	6	7	10	13	23	17	14



# BUBBLE SORT - ASCENDING

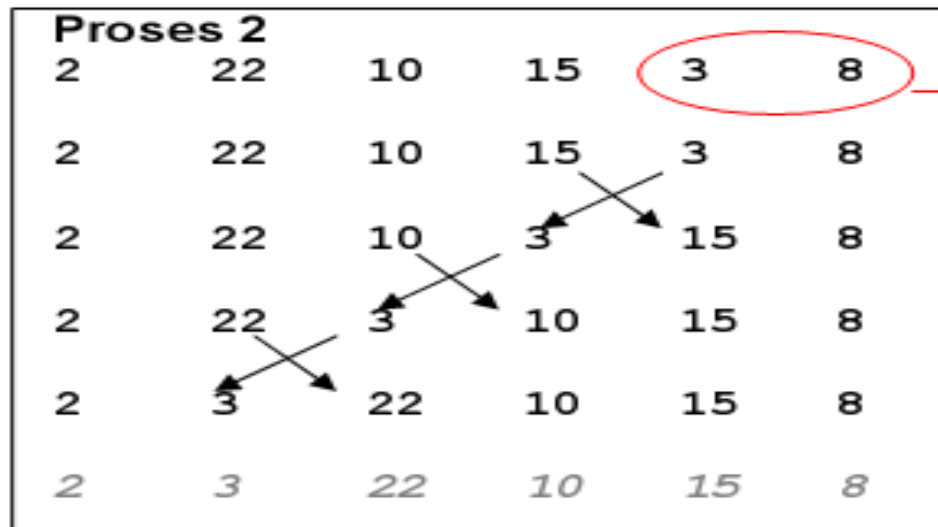
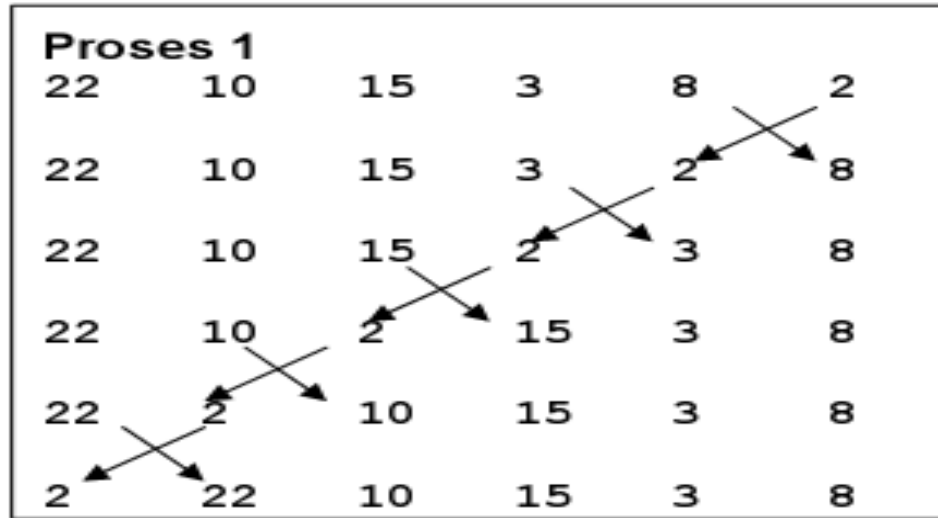
A: [ 23,17,14,6,13,10,1,5,7]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
i=8	1	5	6	7	10	13	23	17	14
j=9	1	5	6	7	10	13	23	14	17
j=8	1	5	6	7	10	13	14	23	17

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
i=9	1	5	6	7	10	13	14	23	17
j=9	1	5	6	7	10	13	14	17	23
j=8	1	5	6	7	10	13	14	17	23
Akhir	1	5	6	7	10	13	14	17	23

# BUBBLE SORT - ASCENDING

A: [ 22, 10, 15, 3, 8, 2 ]



Tidak ada penukaran,  
karena  $3 < 8$

Pegurutan berhenti di sini!

# BUBBLE SORT - ASCENDING

**Proses 3**

2	3	22	10	15	8
2	3	22	10	8	15
2	3	22	8	10	15
2	3	8	22	10	15
2	3	8	22	10	15
2	3	8	22	10	15

→ Pegurutan berhenti di sini!

**Proses 4**

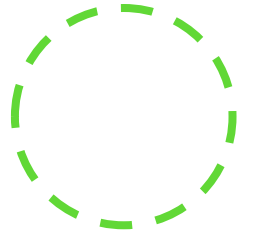
2	3	8	22	10	15
2	3	8	22	10	15
2	3	8	10	22	15
2	3	8	10	22	15
2	3	8	10	22	15
2	3	8	10	22	15

→ Tidak ada penukaran, karena  $10 < 15$

→ Pegurutan berhenti di sini!



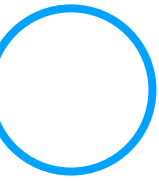
# BUBBLE SORT - ASCENDING



## Proses 5

2	3	8	10	22	15
2	3	8	10	15	22
<i>2</i>	<i>3</i>	<i>8</i>	<i>10</i>	<i>15</i>	<i>22</i>
<i>2</i>	<i>3</i>	<i>8</i>	<i>10</i>	<i>15</i>	<i>22</i>
<i>2</i>	<i>3</i>	<i>8</i>	<i>10</i>	<i>15</i>	<i>22</i>
<i>2</i>	<i>3</i>	<i>8</i>	<i>10</i>	<i>15</i>	<i>22</i>

→ Pegurutan berhenti di sini!



# BUBBLE SORT- PROGRAM

## Versi 1

```
void bubble_sort(int data[]){
    for(int i=1;i<n;i++){
        for(int j=n-1;j>=i;j--){
            if(data[j]<data[j-1]) tukar(&data[j],&data[j-1]); //ascending
        }
    }
}
```

## Versi 2

```
void bubblesort2(int data[]){
    for(i=1;i<6;i++){
        for(int j=0;j<6-i;j++){
            if(data[j]>data[j+1])
                tukar(&data[j],&data[j+1]); //descending
        }
    }
}
```



# BUBBLE SORT- PROGRAM




Dengan prosedur tersebut, data terurut naik (ascending), untuk urut turun (descending), ubah bagian:

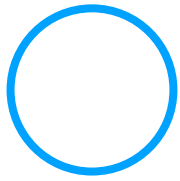
```
if (data[j]<data[j-1]) tukar(&data[j],&data[j-1]);
```

Menjadi:

```
if (data[j]>data[j-1]) tukar(&data[j],&data[j-1]);
```


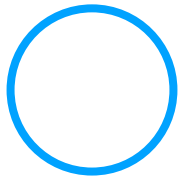


“The bubble sort is an easy algorithm to program, but it is slower than many other sorts”




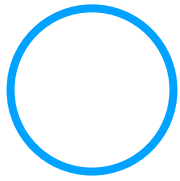


# SHELL SORT (METODE SHELL) - 4

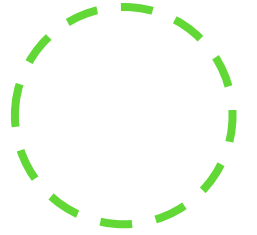
- Metode Shell sering disebut juga **metode pertambahan menurun**.
  - Metode ini mengurutkan data dengan cara **membandingkan suatu data** dengan data lain yang **memiliki jarak tertentu**, kemudian dilakukan penukaran jika diperlukan.
  - Pada metode ini pertama-tama adalah **menentukan jarak mula-mula** dari data yang akan dibandingkan, **yaitu  $N/2$** .
  - Setelah itu, dilakukan **pengulangan dari 1 sampai dengan  $N/2$** .
  - Pada masing-masing pengulangan dilakukan **pembandingan** antara **data ke- $j$**  dengan **data ke- $(j+N/2)$** .
  - Apabila **data ke- $j$  lebih besar dari data ke- $(j+N/2)$**  maka kedua data tersebut ditukar.
  - Pengulangan dari 1 sampai dengan  $N/2$  ini dilakukan sampai semua data ke- $j$  selalu lebih kecil dari data ke- $(j+N/2)$ .
- 
- 



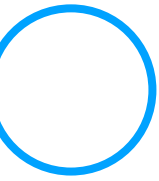
# SHELL SORT (4)

- Pada proses berikutnya, digunakan jarak  $(N/2)/2$  atau  $N/4$
  - Kemudian dilakukan pengulangan dari 1 s.d.  $(N-N/4)$  atau  $3N/4$
  - Pada masing-masing pengulangan dilakukan perbandingan antara data ke- $j$  dengan data ke- $(j+N/4)$
  - Apabila diperlukan, kedua data tersebut ditukar
  - Pengulangan dari 1 s.d.  $3N/4$  ini dilakukan sampai semua data ke- $j <$  dari pada data ke- $(j+N/4)$
  - Pada proses berikutnya, digunakan jarak  $(N/4)/2$  atau  $N/8$
  - Demikian seterusnya sampai jarak yang digunakan adalah 1
- 
- 

# ALGORITMA SHELL SORT (METODE PENGURUTAN PENAMBAHAN MENURUN)



1.  $i \leftarrow 2$
2. Selama ( $i > 1$ ) kerjakan baris 3 s.d. 10
3.  $i \leftarrow i/2$
4. Selesai  $\leftarrow$  false
5. Kerjakan baris 5 s.d. 9 sampai (Selesai=true)
6. Selesai  $\leftarrow$  true
7.  $j \leftarrow 1$
8. Selama ( $j \leq N - i$ ) kerjakan baris 9 dan 10
9. Jika ( $\text{Data}[j] > \text{Data}[j+i]$ ) maka tukar  $\text{Data}[j]$ ,  $\text{Data}[j+i]$ , Selesai  $\leftarrow$  false
10.  $j \leftarrow j + 1$



# SHELL SORT - ASCENDING

A: [ 23,17,14,6,13,10,1,5,7]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Awal	23	17	14	6	13	10	1	5	7
jarak=4	23	17	14	6	13	10	1	5	7
j=1	23	17	14	6	13	10	1	5	7
j=2	13	17	14	6	23	10	1	5	7
j=3	13	10	14	6	23	17	1	5	7
j=4	13	10	1	6	23	17	14	5	7
j=5	13	10	1	5	23	17	14	6	7
J=6	13	10	1	5	7	17	14	6	23

$N/2 = 9/2 = 4$   
 $A' : [23,17,14,6,13]$   
 $A'' : [10,1,5,7]$

# SHELL SORT - ASCENDING

A: [ 23,17,14,6,13,10,1,5,7]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
jarak=2	13	10	1	5	7	17	14	6	23
j=1	13	10	1	5	7	17	14	6	23
j=2	1	10	13	5	7	17	14	6	23
j=3	1	5	13	10	7	17	14	6	23
j=4	1	5	7	10	13	17	14	6	23
j=5	1	5	7	10	13	17	14	6	23
j=6	1	5	7	10	13	17	14	6	23
j=7	1	5	7	10	13	6	14	17	23



# SHELL SORT - ASCENDING

A: [ 23,17,14,6,13,10,1,5,7]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
jarak=2	1	5	7	10	13	6	14	17	23
j=1	1	5	7	10	13	6	14	17	23
j=2	1	5	7	10	13	6	14	17	23
j=3	1	5	7	10	13	6	14	17	23
j=4	1	5	7	10	13	6	14	17	23
j=5	1	5	7	6	13	10	14	17	23
j=6	1	5	7	6	13	10	14	17	23
j=7	1	5	7	6	13	10	14	17	23

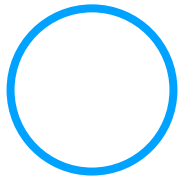
# SHELL SORT - ASCENDING

A: [ 23,17,14,6,13,10,1,5,7]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
jarak=1	1	5	7	6	13	10	14	17	23
j=1	1	5	7	6	13	10	14	17	23
j=2	1	5	7	6	13	10	14	17	23
j=3	1	5	7	6	13	10	14	17	23
j=4	1	5	6	7	13	10	14	17	23
j=5	1	5	6	7	13	10	14	17	23
j=6	1	5	6	7	10	13	14	17	23
j=7	1	5	6	7	10	13	14	17	23
j=8	1	5	6	7	10	13	14	17	23
Akhir	1	5	6	7	10	13	14	17	23

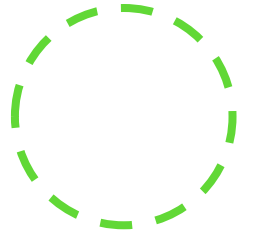


# QUICK SORT (METODE QUICK) - 5

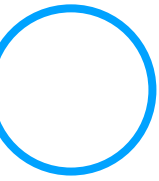
- Suatu metode pengurutan yang membandingkan **suatu elemen (disebut pivot)** dengan elemen yang lain dan menyusunnya sedemikian rupa sehingga **elemen-elemen lain yang lebih kecil dari pada pivot** tersebut **terletak di sebelah kirinya** dan elemen-elemen lain yang lebih besar daripada pivot tersebut terletak di sebelah kanannya.
  - Sehingga dengan demikian telah terbentuk **dua sublist**, yang terletak di sebelah kiri dan kanan dari pivot.
  - Lalu pada sublist kiri dan sublist kanan dianggap sebuah list baru dan kerjakan proses yang sama seperti sebelumnya.
  - List yang sebelah kiri pivot juga diterapkan aturan seperti pivot, yaitu membandingkan elemen yang lainnya lagi, kalau lebih kecil diletakkan sebelah kiri, kalau lebih besar diletakkan disebelah kanan.
  - Hal juga berlaku untuk list yang letaknya di sebelah kanan disebut juga dengan metode partisi
  - Pada metode quick jarak dari kedua elemen yang ditukarkan dibuat cukup besar dengan tujuan untuk mempertinggi efektivitasnya.
  - Demikian seterusnya sampai tidak terdapat sublist lagi sehingga di dalamnya telah **terjadi proses Rekursif**
- 



# QUICK SORT (METODE PENGURUTAN CEPAT)

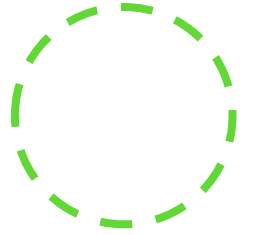


- Proses pengurutan dengan metode pengurutan cepat :
- Pilih data tertentu yang dinamakan pivot, misalnya  $x$ .
- Pivot ini harus diletakkan pada posisi ke- $j$  sedemikian hingga **data antara 1 s.d.  $(j-1)$  lebih kecil daripada  $x$** , sedangkan **data pada posisi ke  $(j+1)$  s.d.  $N$  lebih besar dari pada  $x$**
- Cara pengaturannya adalah menukarkan data diantara posisi 1 s.d.  $(j-1)$  yang lebih besar daripada  $x$  dengan data diantara posisi  $(j+1)$  s.d.  $N$  yang lebih kecil daripada  $x$

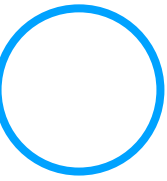





# ALGORITMA QUICK SORT (METODE PENGURUTAN CEPAT)




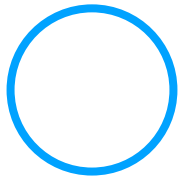
1.  $x \leftarrow \text{Data}[(L+R)/2]$
2.  $i \leftarrow L$
3.  $j \leftarrow R$
4. Selama ( $i \leq j$ ) kerjakan baris 5 s.d. 12
5. Selama ( $\text{Data}[i] < x$ ) kerjakan  $i \leftarrow i+1$
6. Selama ( $\text{Data}[i] > x$ ) kerjakan  $j \leftarrow j-1$
7. Jika ( $i \leq j$ ) maka kerjakan baris 8 s.d. 10; jika tidak dikerjakan baris 11
8. Tukar  $\text{Data}[i]$  dengan  $\text{Data}[j]$
9.  $i \leftarrow i + 1$
10.  $j \leftarrow j - 1$
11. Jika ( $L < j$ ) kerjakan lagi baris 1 dengan  $R=j$
12. Jika ( $i < R$ ) kerjakan lagi baris 1 dengan  $L=i$





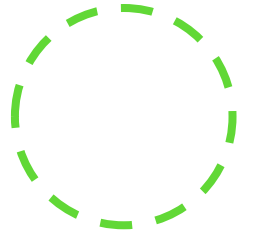
# ALGORITMA PROGRAM QUICK SORT (METODE PENGURUTAN CEPAT)

Algoritmanya sebagai berikut :

1. Tentukan data-data yang akan diurutkan dan disimpan dalam array
  2. Lakukan pengulangan dari data-data tersebut
  3. Data-data yang ada dibagi menjadi 2 bagian dimana bagian yang satu dengan yang lain dilakukan proses pengecekan
  4. Pada bagian yang pertama, lakukan perbandingan antara data yang satu dengan data yang lain, dimana kalau data yang satu lebih kecil dari data yang lain, maka posisinya ditukar. Kalau datanya lebih kecil diletakkan disebelah kiri, kalau datanya lebih besar diletakkan disebelah kanan. Hal ini berlaku pada bagian yang kedua
  5. Tampilkan data hasil perbandingan
  6. Ulangi langkah 4, sampai semua data dibandingkan
  7. Selesai
- 
- 

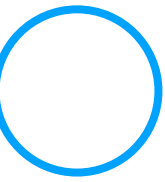


# QUICK SORT - ASCENDING



A: [ 65,2,44,26,19,22,5,3,12]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Awal	65	2	44	26	19	22	5	3	12
L=1, R=9	65	2	44	26	19	22	5	3	12
L=1, R=5	12	2	44	26	19	22	5	3	65
L=1, R=4	12	2	44	26	19	22	5	3	65
L=1, R=3	12	2	44	26	19	22	5	3	65
L=1, R=2	12	2	44	26	19	22	5	3	65
L=1, R=1	2	12	44	26	19	22	5	3	65



# QUICK SORT - ASCENDING

A: [ 65,2,44,26,19,22,5,3,12]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Awal	2	12	44	26	19	22	5	3	65
L=2, R=9	2	12	44	26	19	22	5	3	65
L=2, R=8	2	12	44	26	19	22	5	3	65
L=2, R=5	2	3	44	26	19	22	5	12	65
L=2, R=4	2	3	44	26	19	22	5	12	65
L=2, R=3	2	3	44	26	19	22	5	12	65
L=2, R=2	2	3	44	26	19	22	5	12	65



# QUICK SORT - ASCENDING

A: [ 65,2,44,26,19,22,5,3,12]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Awal	2	3	44	26	19	22	5	12	65
L=3, R=9	2	3	44	26	19	22	5	12	65
L=3, R=8	2	3	44	26	19	22	5	12	65
L=3, R=7	2	3	12	26	19	22	5	44	65
L=3, R=5	2	3	5	26	19	22	12	44	65
L=3, R=4	2	3	5	26	19	22	12	44	65
L=3, R=3	2	3	5	26	19	22	12	44	65

# QUICK SORT - ASCENDING

A: [ 65,2,44,26,19,22,5,3,12]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Awal	2	3	5	26	19	22	12	44	65
L=4, R=9	2	3	5	26	19	22	12	44	65
L=4, R=8	2	3	5	26	19	22	12	44	65
L=4, R=7	2	3	5	26	19	22	12	44	65
L=4, R=6	2	3	5	12	19	22	26	44	65
L=4, R=5	2	3	5	12	19	22	26	44	65
Akhir	2	3	5	12	19	22	5	44	65



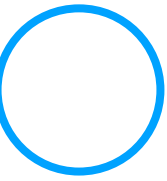
# MERGE SORT (METODE PENGGABUNGAN) - 6



- Metode ini sering digunakan pada pengurutan berkas.
- Mula-mula diberikan dua kumpulan data yang sudah dalam keadaan terurut.
- Kedua kumpulan data tersebut harus dijadikan satu tabel sehingga dalam keadaan terurut.
- Syaratnya ada dua kumpulan data yang sudah dalam keadaan urut.
- Kedua kumpulan data tersebut harus dijadikan satu tabel sehingga dalam keadaan urut.
- Misalnya :  $T1 \leftarrow 2, 19, 26, 44, 65$



$T2 \leftarrow 3, 5, 12, 22$


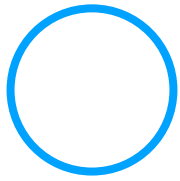




# MERGE SORT (METODE PENGGABUNGAN)

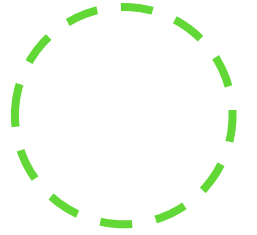



Proses pengurutan dengan metode penggabungan :

1. Mula-mula diambil data pertama dari T1, 2, dan data pertama dari T2, yaitu 3
  2. Data ini dibandingkan, kemudian yang lebih kecil diletakkan sebagai data pertama hasil hasil pengurutan, misalnya T3
  3. Jadi T3 akan memiliki satu data, yaitu 2
  4. Data yang lebih besar, yaitu 3, kemudian dibandingkan dengan data kedua dari T1, yaitu 19
  5. Karena lebih kecil, maka 3 diletakkan sebagai data kedua dari T3
- 
- 

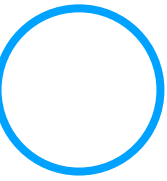


# ALGORITMA MERGE SORT (METODE PENGGABUNGAN)



- 
1.  $i \leftarrow 1$
  2.  $j \leftarrow 1$
  3.  $J3 \leftarrow 1$
  4. Kerjakan baris 5 s.d. 7 sampai ( $i > J1$ ) atau ( $j > J2$ )
  5.  $J3 \leftarrow J3 + 1$
  6. Jika ( $T1[j] < T2[j]$ ) maka  $T3[J3] \leftarrow T1[i]$ ,  $i \leftarrow i + 1$
  7. Jika ( $T1[j] \geq T2[j]$ ) maka  $T3[J3] \leftarrow T2[j]$ ,  $j \leftarrow j + 1$
  8. Jika ( $i > J$ ) maka kerjakan baris 9, jika tidak dikerjakan baris 15
  9.  $i \leftarrow j$

1. Selama ( $i \leq J2$ ) kerjakan baris 11 s.d. 13
2.  $J3 \leftarrow J3 + 1$
3.  $T3[J3] \leftarrow T2[i]$
4.  $i \leftarrow i + 1$
5. Selesai
6.  $j \leftarrow i$
7. Selama ( $j \leq J1$ ) kerjakan baris 17 s.d. 19
8.  $J3 \leftarrow J3 + 1$
9.  $T3[J3] \leftarrow T1[j]$
10.  $j \leftarrow j + 1$



# MERGE SORT - ASCENDING

A: [2, 19, 26, 44, 65, 3, 5, 12, 22]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Tahap 1									
T1	2	19	26	44	65				
T2	3	5	12	22					
T3	2								

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Tahap 2									
T1		19	26	44	65				
T2	3	5	12	22					
T3	2	3							

# MERGE SORT - ASCENDING

A: [2, 19, 26, 44, 65, 3, 5, 12, 22]

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Tahap 3									
T1		19	26	44	65				
T2		5	12	22					
T3	2	3	5						

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Tahap 4									
T1		19	26	44	65				
T2			12	22					
T3	2	3	5	12					

# MERGE SORT - ASCENDING

A: [2, 19, 26, 44, 65, 3, 5, 12, 22]

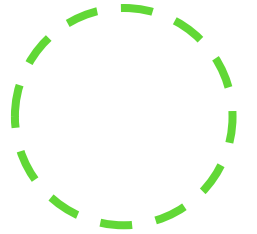
Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Tahap 5									
T1		19	26	44	65				
T2				22					
T3	2	3	5	12	19				

Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Tahap 6									
T1			26	44	65				
T2				22					
T3	2	3	5	12	19	22			



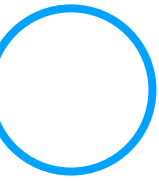
# MERGE SORT - ASCENDING

A: [2, 19, 26, 44, 65, 3, 5, 12, 22]



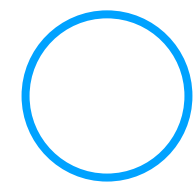
Iterasi	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Tahap 7									
T1			26	44	65				
T2									
T3	2	3	5	12	19	22	26		

Iterasi i	Data								
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
Tahap 8									
T1				44	65				
T2									
T3	2	3	5	12	19	22	44		
Akhir									
T3	2	3	5	12	19	22	<b>26</b>	44	65





# QUESTIONS





# TUGAS LMS

