STRUKTUR DATA (3)

sorting array

Oleh **Antonius Rachmat C, S.Kom, M.Cs**

Sorting

- 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)
- Pengurutan (Sorting) adalah proses menyusun kembali data yang sebelumnya telah disusun dengan suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu.

- **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

Metode Pengurutan Data

- Pengurutan berdasarkan perbandingan (*comparison-based sorting*)
  - Bubble sort, exchange sort
- Pengurutan berdasarkan prioritas (*priority queue sorting method*)
  - Selection sort, heap sort (menggunakan tree)
- Pengurutan berdasarkan penyisipan dan penjagaan terurut (*insert and keep sorted method*)
  - Insertion sort, tree sort
- Pengurutan berdasarkan pembagian dan penguasaan (*devide and conquer method*)
  - Quick sort, merge sort
- Pengurutan berkurang menurun (*diminishing increment sort method*)
  - Shell sort (pengembangan insertion)

Deklarasi Array

- **Misal, deklarasikan:**

```
int data[100];
int n; //untuk jumlah data
```

- **Fungsi untuk Tukar 2 Buah Data (by reference)**:

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

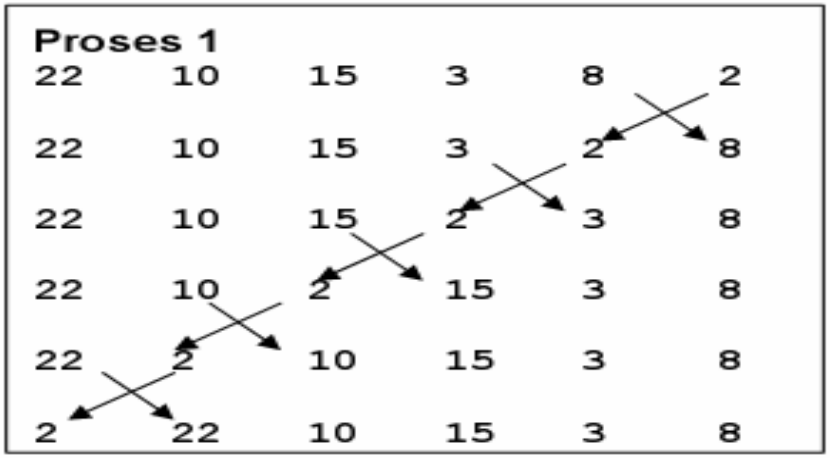
Bubble Sort

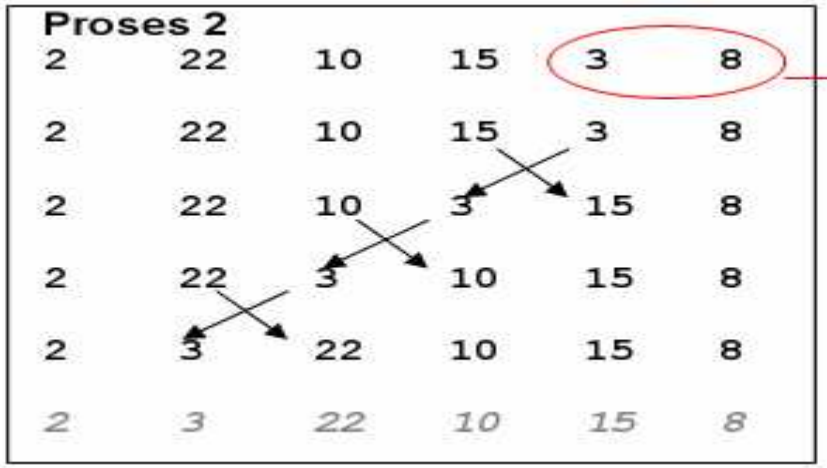
- Metode sorting termudah
- Diberi nama "Bubble" karena proses pengurutan secara berangsur-angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda.
- Bubble Sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya.

Bubble Sort (2)

- 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 Sort (3)

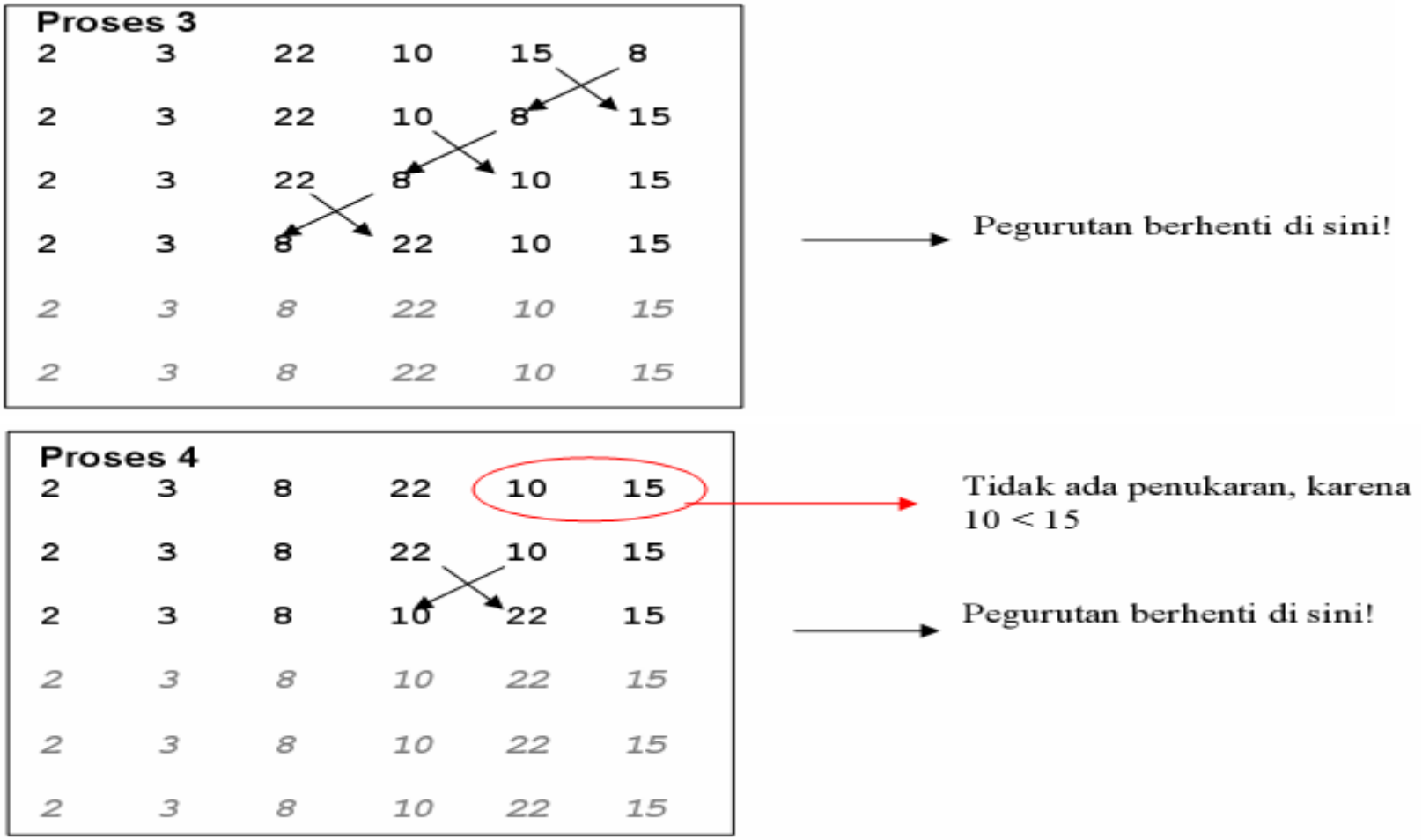




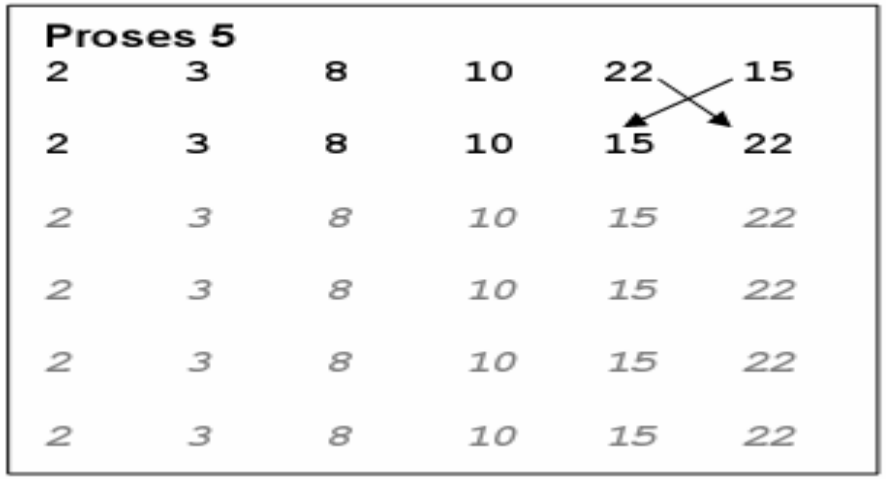
Tidak ada penukaran, karena $3 < 8$

Pegurutan berhenti di sini!

Bubble Sort (4)



Bubble Sort (5)



Pegurutan berhenti di sini!

Bubble Sort (6)

- **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 (6)

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

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

Menjadi:

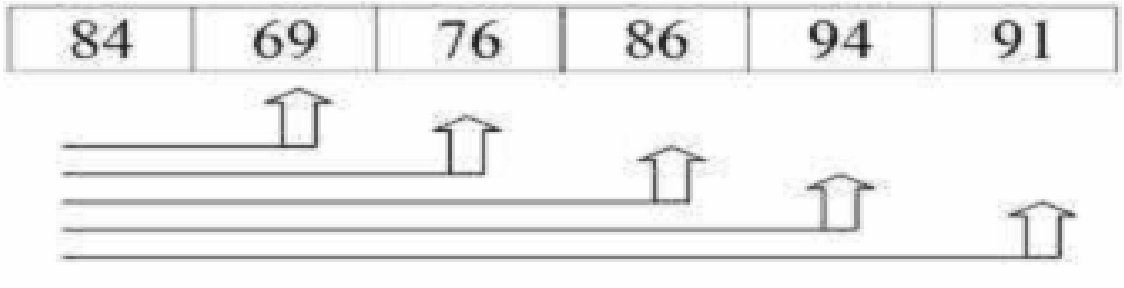
```
if (data[j]>data[j-1]) tukar(&data[j],&data[j-1]);
```

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

Exchange Sort

- Sangat mirip dengan Bubble Sort
- Banyak yang mengatakan Bubble Sort sama dengan Exchange Sort
- Pebedaan : dalam hal bagaimana membandingkan antar elemen-elemennya.
  - Exchange sort membandingkan **suatu elemen** dengan **elemen-elemen lainnya** dalam array tersebut, dan melakukan pertukaran elemen jika perlu. Jadi ada elemen yang selalu menjadi elemen **pusat (pivot).**
  - Sedangkan Bubble sort akan membandingkan **elemen pertama/terakhir** dengan **elemen sebelumnya/sesudahnya**, kemudian elemen tersebut itu akan menjadi **pusat (pivot)** untuk dibandingkan dengan elemen sebelumnya/sesudahnya lagi, begitu seterusnya.

Exchange Sort (2)



Proses 1

Pivot (Pusat)

84	69	76	86	94	91
84	69	76	86	94	91
84	69	76	**86**	94	91
86	69	76	**84**	**94**	91
94	69	76	84	**86**	91
94	69	76	84	86	91

Exchange Sort (3)

Proses 2

Pivot (Pusat)

94	**69**	**76**	84	86	91
94	**76**	**69**	**84**	86	91
94	**84**	69	**76**	**86**	91
94	**86**	69	76	**84**	**91**
94	**91**	69	76	84	86

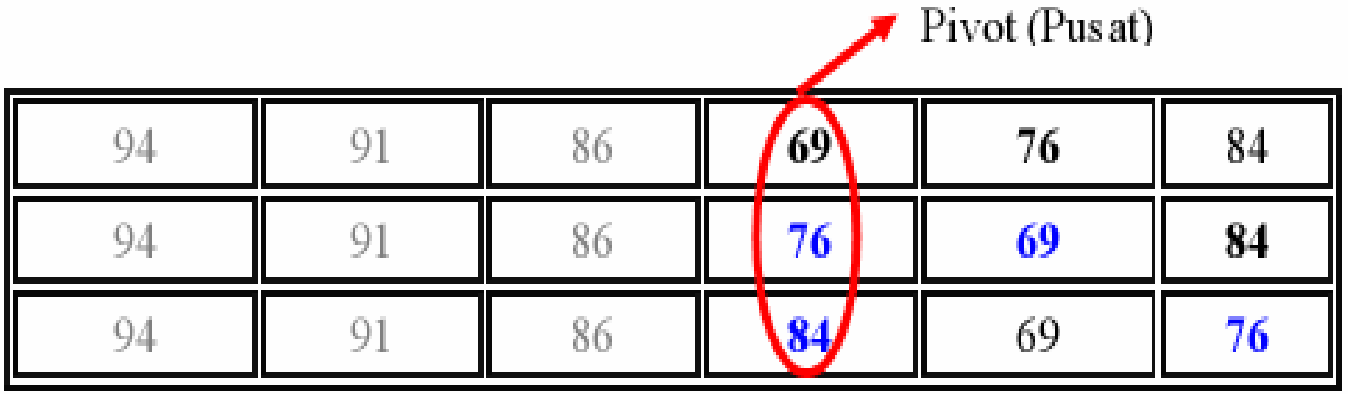
Proses 3

Pivot (Pusat)

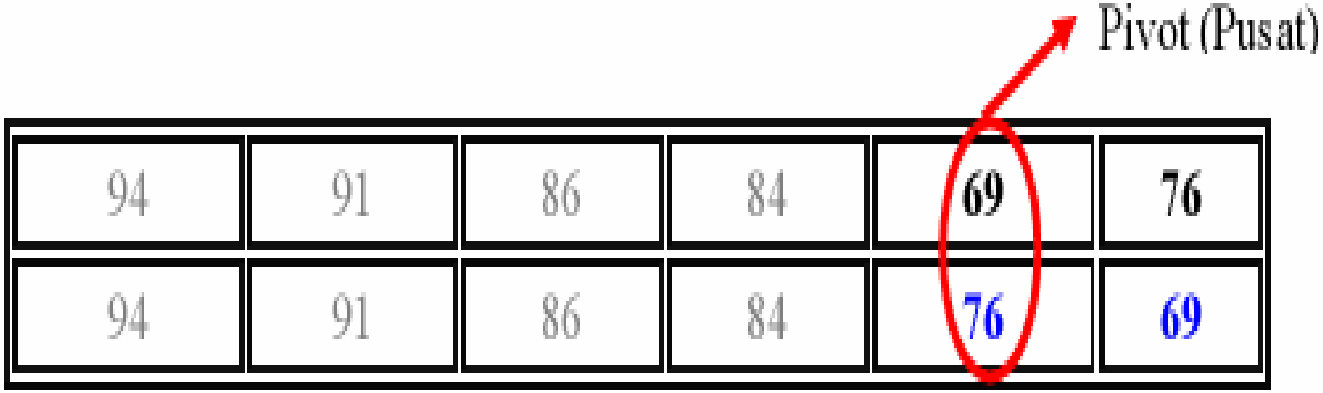
94	91	**69**	**76**	84	86
94	91	**76**	**69**	**84**	86
94	91	**84**	69	**76**	**86**
94	91	**86**	69	76	84

Exchange Sort (4)

Proses 4



Proses 5



Exchange Sort (5)

- **Prosedur Exchange Sort**

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

Selection Sort

- Merupakan kombinasi antara sorting dan searching
- 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, pembandingan dan pengubahan **hanya dilakukan** pada **indeks** pembanding saja, pertukaran data secara fisik terjadi pada **akhir** proses.

Selection Sort (2)

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

Proses 4

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

Selection Sort (3)

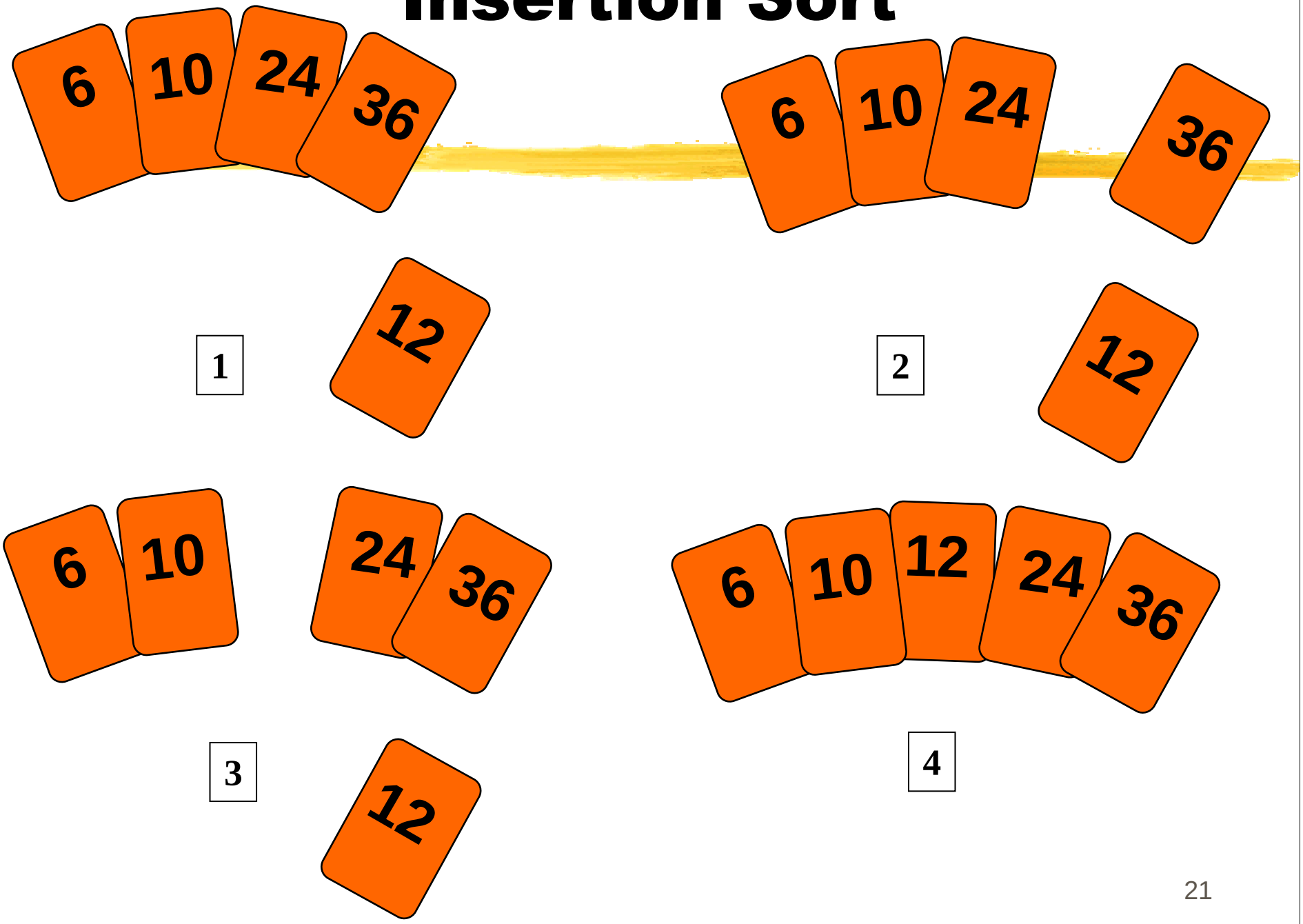
- **Prosedur Selection Sort**

```
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]);
  }
}
```

Insertion Sort

- Mirip dengan cara orang **mengurutkan** kartu, selembar demi selembar 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

Insertion Sort



Insertion Sort (2)

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

Insertion Sort (3)

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

```
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;
      }
}
```

Shell Sort

- Metode Pertambahan Menurun
- Dikembangkan oleh Donald L. Shell (1959)
- Mengurutkan data dengan cara membandingkan suatu data dengan data lain yang memiliki jarak tertentu sehingga dibentuk sub-list, kemudian dilakukan pertukaran jika diperlukan

Definisi

- Jarak yang digunakan disebut increment value, atau sequence number **k**
  - Misal sekuens: 5,3,1
  - Pengambilan sekuens bebas, asal menurun
- Jika k=5, maka sublistnya:
  - Data[0], Data[5], Data[10], …
  - Data[1], Data[6], Data[11], …
  - Data[2], Data[7], Data[12], …
- Jika k=3, maka sublistnya:
  - Data[0], Data[3], Data[6], …
  - Data[1], Data[4], Data[7], …
  - Data[2], Data[5], Data[8], …

Proses dan Contoh

- Pemilihan sekuens dimulai dari N/2
- Kemudian dilakukan perulangan dari j=1 – N/2, pada masing-masing pengulangan dilakukan pembandingan antara data yg ke-j dengan data ke-(j+N/2).
  - Bila data ke-j lebih besar dari dataa ke-(j+N/2), maka tukar
  - Perulangan dilakukan terus smp semua data ke-j selalu lebih kecil daripada data ke-(j+N/2)
- Proses berikutnya sama, gunakan jarak (N/2)/2 atau N/4 dan kemudian N/8 dst, hingga N=1
- Contoh Data:

  Data: 12 35 9 11 3 17 23 15 31

  Index: 0 1 2 3 4 5 6 7 8

  Size: 9

Proses 1 - Shell Sort

Size = 9

Data:	12	35	9	11	3	17	23	15	31
Index:	0	1	2	3	4	5	6	7	8

Jarak = 9 / 2 = 4

Buat sublist:

- Data[0], Data[4] = 12 dan 3, tukar, hasilnya:

Data:	**3**	35	9	11	**12**	17	23	15	31
Index:	0	1	2	3	4	5	6	7	8

- Data[1], Data[5] = 35 dan 17, tukar, hasilnya:

Data:	3	**17**	9	11	12	**35**	23	15	31
Index:	0	1	2	3	4	5	6	7	8

- Data[2], Data[6], tetap
- Data[3], Data[7], tetap
- Data[4], Data[8], tetap

Data:	3	17	9	11	12	35	23	15	31
Index:	0	1	2	3	4	5	6	7	8

Diulangi sekali lagi, **semua sudah benar**!

- Data[0], Data[4], tetap
- Data[1], Data[5], tetap
- Data[2], Data[6], tetap
- Data[3], Data[7], tetap
- Data[4], Data[8], tetap

Proses 2 – Shell Sort

Jarak = 4/2 = 2
Buat sublist:

- Data[0], Data[2], tetap
- Data[1], Data[3] = 17 dan 11, tukar, hasilnya:

Data:	3	**11**	9	**17**	12	35	23	15	31
Index:	0	1	2	3	4	5	6	7	8

- Data[2], Data[4], tetap
- Data[3], Data[5], tetap
- Data[4], Data[6], tetap
- Data[5], Data[7] = 35 dan 15, tukar, hasilnya:

Data:	3	11	9	17	12	**15**	23	**35**	31
Index:	0	1	2	3	4	5	6	7	8

- Data[6], Data[8], tetap

Data:	3	11	9	17	12	15	23	35	31
Index:	0	1	2	3	4	5	6	7	8

Diulangi sekali lagi, **ada yang masih salah**!

- Data[0], Data[2], tetap
- Data[1], Data[3], tetap
- Data[2], Data[4], tetap
- Data[3], Data[5], 17 dan 15, tukar, hasilnya:

Data:	3	11	9	**15**	12	**17**	23	35	31
Index:	0	1	2	3	4	5	6	7	8

- Data[4], Data[6], tetap
- Data[5], Data[7], tetap
- Data[6], Data[8], tetap

Proses 3 – Shell Sort

Jarak = 2/2 = 1

Buat sublist:

- Data[0], Data[1], tetap
- Data[1], Data[2] = 11 dan 9, tukar, hasilnya:

Data:	3	**9**	**11**	15	12	17	23	35	31
Index:	0	1	2	3	4	5	6	7	8

- Data[2], Data[3], tetap
- Data[3], Data[4] = 15 dan 12, tukar, hasilnya:

Data:	3	9	11	**12**	**15**	17	23	35	31
Index:	0	1	2	3	4	5	6	7	8

- Data[4], Data[5], tetap
- Data[5], Data[6] , tetap
- Data[6], Data[7], tetap
- Data[7], Data[8], 35 dan 31, tukar

Data:	3	9	11	12	15	17	23	31	35
Index:	0	1	2	3	4	5	6	7	8

Diulangi sekali lagi, **semua sudah benar**!

- Data[0], Data[1], tetap
- Data[1], Data[2], tetap
- Data[2], Data[3], tetap
- Data[3], Data[4], tetap
- Data[4], Data[5], tetap
- Data[5], Data[6], tetap
- Data[6], Data[7], tetap
- Data[7], Data[8], tetap

SELESAI

```c
#include <stdio.h>

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

void shell_sort(int A[], int size) {
    int jarak = size;
    int did_swap;
    int j;
    while(jarak > 0){
        jarak = jarak / 2;
        do{
            did_swap = 1;
            for(j=0;j<size-jarak;j++){
                if(A[j] > A[j+jarak]){
                    tukar(&A[j],&A[j+jarak]);
                    did_swap = 0;
                }
            }
        }while(did_swap==0);
    }
}
void main(){
    int i;
    int n=9;
    int data[9] = {12,35,9,11,3,17,23,15,31};
    shell_sort(data,n);
    for(i=0;i<n;i++){
        printf("%d ",data[i]);
    }
}
```

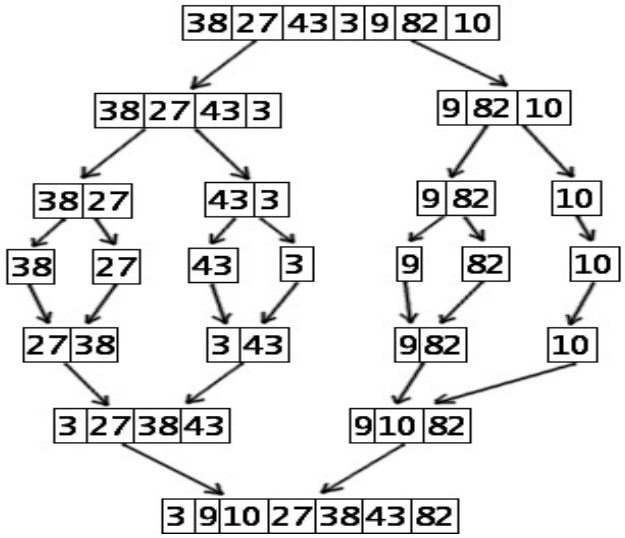
Perbandingan

- Tabel Perbandingan Kecepatan Metode Pengurutan Data
- Untuk data sejumlah 10.000 data pada komputer Pentium II 450 MHz

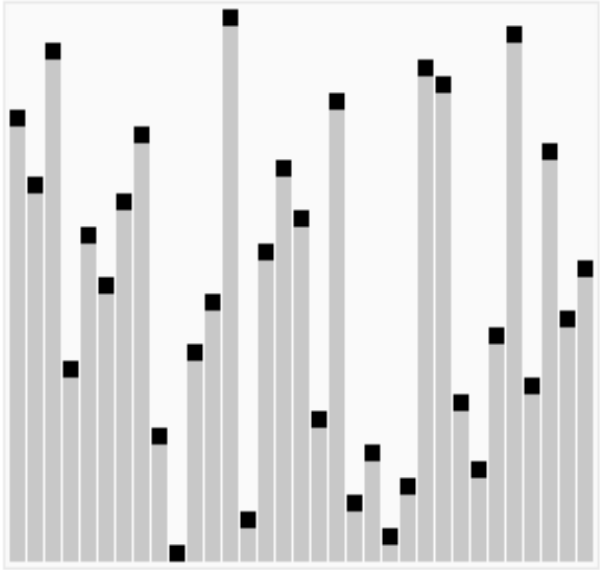
Metode	Waktu (detik)		
	Data Acak	Data Ascending	Data Descending
Bubble Sort	11,2	1,32	19,77
Insertion Sort	1,09	0,00	2,25
Selection Sort	1,32	1,32	19,77
Shell Sort	0,11	0,00	0,06

Masih banyak lagi

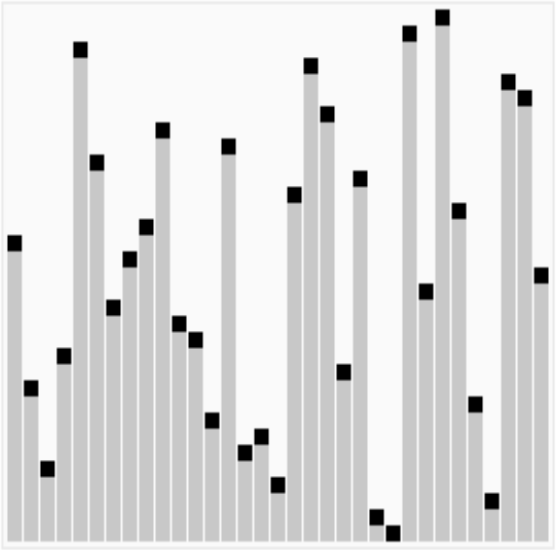
- Merge Sort



Heap Sort



Quick Sort



Soal

Carilah 3 metode sorting lainnya dan tuliskan dalam paper beserta source code, cara dan analisis dan tiap-tiap metode sorting yang ada!

Buatlah semua procedure-procedure yang ada di atas dalam program utuh!

NEXT:
Array Stack dan Queue