PERTEMUAN 10

METODE DIVIDE AND CONQUER

Metode D And C

Divide

Memilah data nilai elemen–elemen dari rangkaian data menjadi dua bagian dan mengulangi pemilahan hingga satu elemen terdiri maksimal dua nilai (Sonita & Nurtaneo, 2015).

Conquer

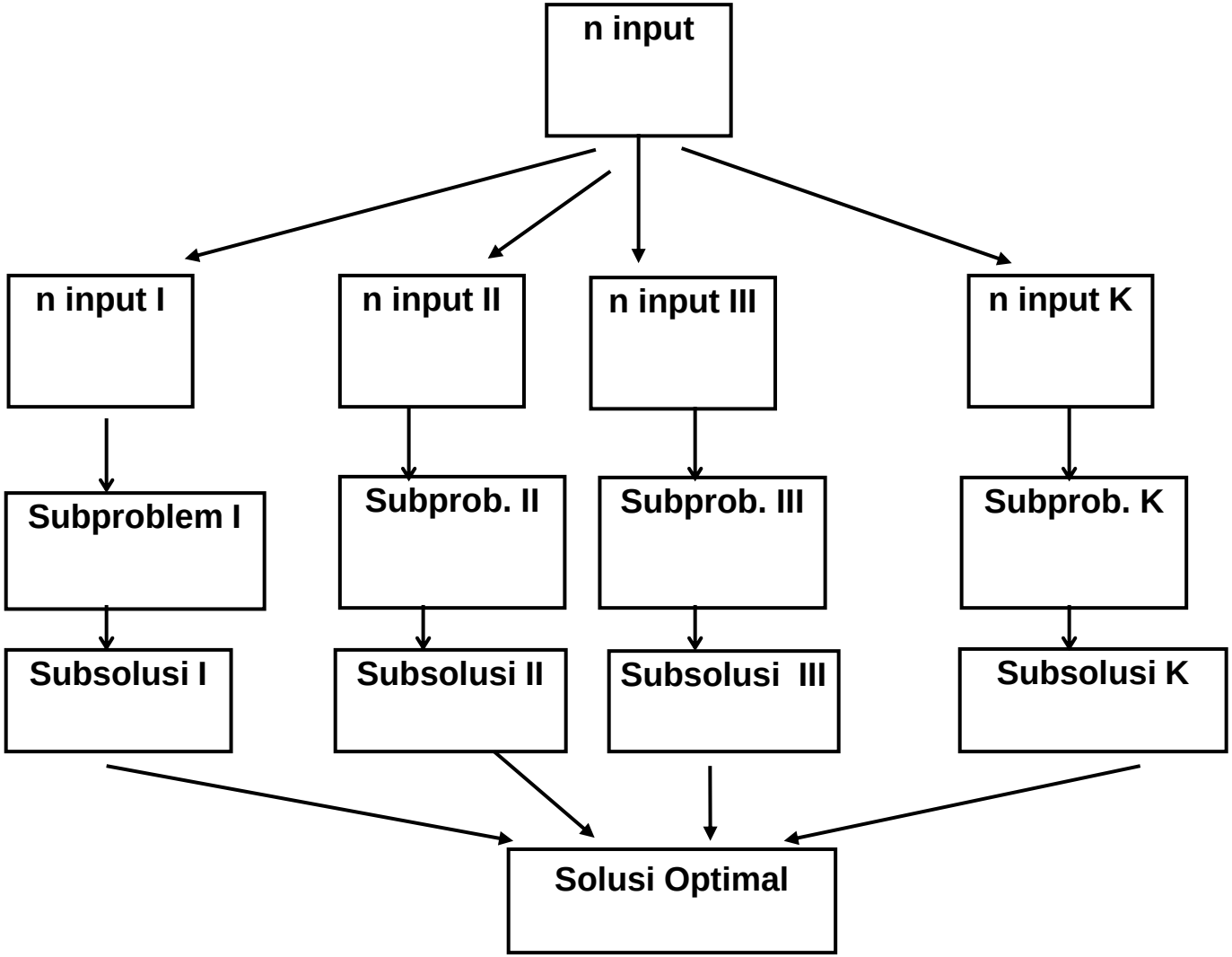
Mengurutkan masing-masing data nilai elemen (Sonita & Nurtaneo, 2015).

Prinsip Dasar

- Membagi n input menjadi k subset input yang berbeda ($1<k\leq n$).
- k subset input tersebut akan terdapat k subproblem.
- Setiap subproblem mempunyai solusi menjadi k subsolusi.
- Dari k subsolusi akan mendapatkan solusi yang optimal

Jika subproblem masih besar → D and C

Bentuk Umum Proses Metode D And C dpt dilihat sbb :



METODE SORTING

1. Pengertian Sorting

Proses pengaturan sederetan data ke dalam suatu urutan atau susunan urutan tertentu. Data yang diurutkan dapat berupa data bilangan, data karakter maupun data string (Sitorus, 2015).

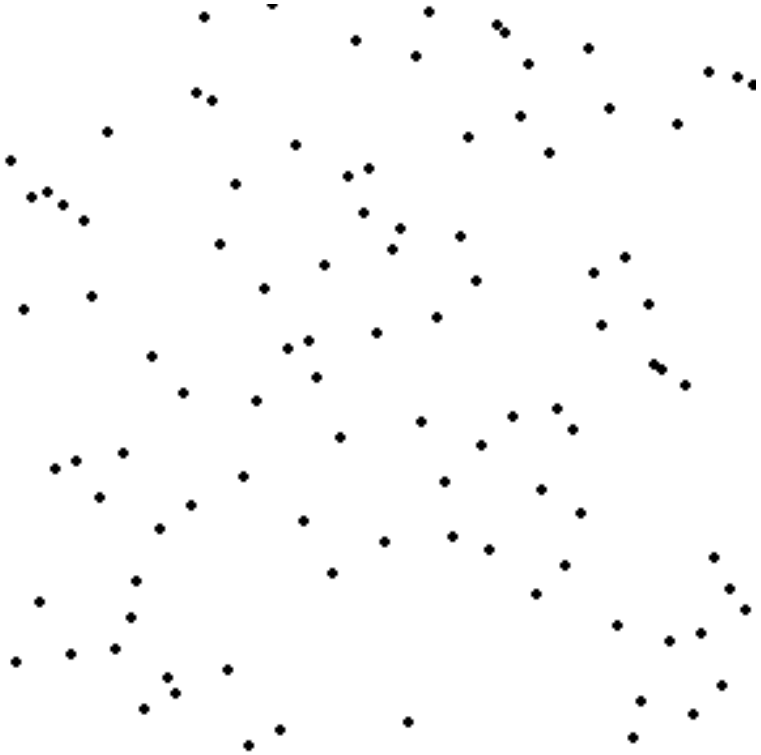
2. Macam-Macam Metode Sorting:

1. Selection Sort
2. Bubble Sort
3. Merge Sort
4. Quick Sort
5. Insertion Sort

Hal yang mempengaruhi Kecepatan Algoritma Sorting:
Jumlah Operasi Perbandingan & Jumlah Operasi pemindahan Data

SELECTION SORT

Teknik pengurutan dengan cara pemilihan elemen atau proses kerja dengan memilih elemen data terkecil untuk kemudian dibandingkan & ditukarkan dengan elemen pada data awal, dst s/d seluruh elemen sehingga menghasilkan pola data yang telah disort.



SELECTION SORT (Lanjutan)

Prinsip Kerja dari Teknik Selection Sort ini adalah :

1. Pengecekan dimulai data ke-1 sampai dengan data ke-n
2. Tentukan bilangan dengan Index terkecil dari data bilangan tersebut
3. Tukar bilangan dengan Index terkecil tersebut dengan bilangan pertama (I = 1) dari data bilangan tersebut
4. Lakukan langkah 2 dan 3 untuk bilangan berikutnya (I= I+1) sampai didapatkan urutan yang optimal.

SELECTION SORT (Lanjutan)

Contoh :		**22**	**10**	**15**	**3**	**8**	**2**
Iterasi 1							
		1	2	3	4	5	6
Langkah 1	:	22	10	15	3	8	2
Langkah 2	:	22	10	15	3	8	2
Langkah 3	:	2	10	15	3	8	22
Langkah 4	:	Ulangi langkah 2 dan 3					
Iterasi 2							
Langkah 1	:	2	10	15	3	8	22
Langkah 2	:	2	10	15	3	8	22
Langkah 3	:	2	3	15	10	8	22
Langkah 4	:	Ulangi langkah 2 dan 3					

SELECTION SORT (Lanjutan)

Iterasi 3

Langkah 1	:	2	3	15	10	8	22
Langkah 2	:	2	3	15	10	**8**	22
Langkah 3	:	2	3	**8**	10	15	22
Langkah 4	:	Ulangi langkah 2 dan 3					

Iterasi 4

Langkah 1	:	2	3	8	10	15	22
Langkah 2	:	2	3	8	**10**	15	22
Langkah 3	:	2	3	8	**10**	15	22
Langkah 4	:	Ulangi langkah 2 dan 3					

SELECTION SORT (Lanjutan)

Iterasi 5

Langkah 1	:	2	3	8	10	15	22
Langkah 2	:	2	3	8	10	**15**	22
Langkah 3	:	2	3	8	10	**15**	22
Langkah 4	:	Ulangi langkah 2 dan 3					

Iterasi 6

Langkah 1	:	2	3	8	10	15	22
Langkah 2	:	2	3	8	10	15	**22**
Langkah 3	:	2	3	8	10	15	**22**
Langkah 4	:	Ulangi langkah 2 dan 3					

SELECTION SORT (Lanjutan)

ilustrasi

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

SELECTION SORT (Lanjutan)

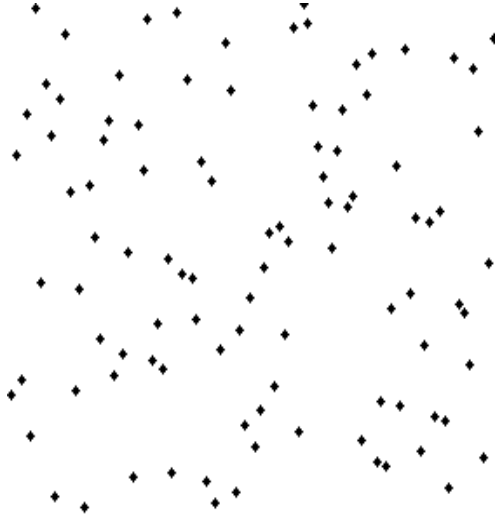
Contoh Program:

```
def SelectionSort(val):
    for i in range(len(val)-1,0,-1):
        Max=0
        for l in range(1,i+1):
            if val[l]>val[Max]:
                Max = l
        temp = val[i]
        val[i] = val[Max]
        val[Max] = temp

Angka = [22,10,15,3,8,2]
SelectionSort(Angka)
print(Angka)
```

Hasil dari program:

```
[2, 3, 8, 10, 15, 22]
```

BUBBLE SORTING



- Metode pengurutan dengan membandingkan data nilai elemen yang sekarang dengan data nilai elemen-elemen berikutnya.

- Pembandingan elemen dapat dimulai dari awal atau mulai dari paling akhir. Apabila elemen yang sekarang lebih besar (untuk urut menaik) atau lebih kecil (untuk urut menaik) dari elemen berikutnya, maka posisinya ditukar, tapi jika tidak maka posisinya tetap (Harumy et al., 2016).

BUBBLE SORTING (Lanjutan)

Prinsip Kerja dari Bubble Sort adalah :

1. Pengecekan mulai dari data ke-1 sampai data ke-n
2. Bandingkan data ke-n dengan data sebelumnya (n-1)
3. Jika lebih kecil maka pindahkan bilangan tersebut dengan bilangan yang ada didepannya (sebelumnya) satu persatu (n-1,n-2,n-3,....dst)
4. Jika lebih besar maka tidak terjadi pemindahan
5. Ulangi langkah 2 dan 3 s/d sort optimal.

BUBBLE SORTING (Lanjutan)

Contoh : 22 10 15 3 8 2

Iterasi 1

		1	2	3	4	5	6
Langkah 1	:	22	10	15	3	8	2
Langkah 2	:	22	10	15	3	8	2
Langkah 3	:	22	10	15	3	2	8
Langkah 4	:	Ulangi langkah 2 dan 3					

Hasil iterasi 1	:	2	22	10	15	3	8

BUBBLE SORTING (Lanjutan)

Iterasi 2

Langkah 1 : 2 22 10 15 3 8

Langkah 2 : 2 22 10 15 3 8

- 8>3, maka 8 tidak pindah, untuk selanjutnya bandingkan data sebelumnya yaitu 3.

Langkah 3 : 2 3 22 10 15 8

Langkah 4 : Ulangi langkah 2 dan 3

Lakukan Iterasi selanjutnya sampai iterasi ke- 6

BUBBLE SORTING (Lanjutan)

Ilustrasi

Iterasi 1

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

Iterasi 2

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

Iterasi 3

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

BUBBLE SORTING (Lanjutan)

Iterasi 4

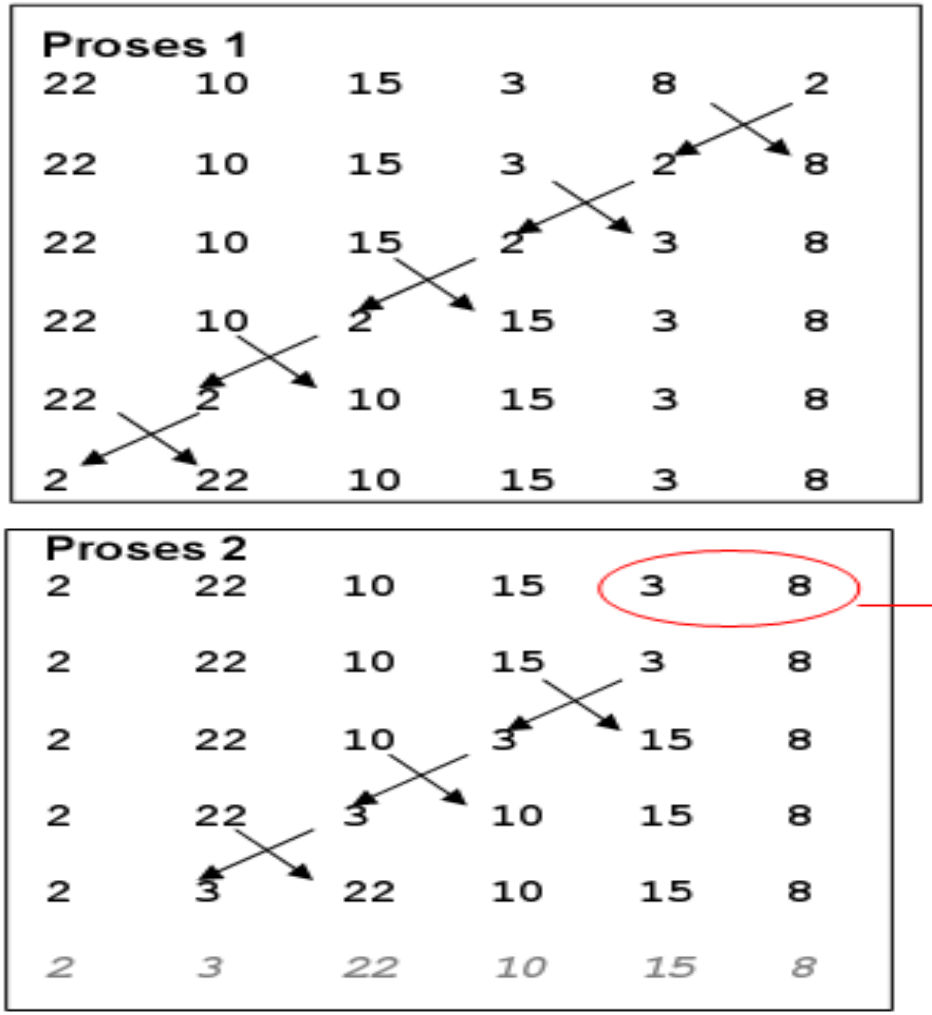
2	22	10	15	3	8
2	3	22	10	15	8
2	3	8	22	10	15
2	3	8	22	10	15

Iterasi 5

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

BUBBLE SORTING (Lanjutan)

Langkah Penyelesaian *BUBBLE SORT* :



BUBBLE SORTING (Lanjutan)

Langkah Penyelesaian *BUBBLE SORT:*

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

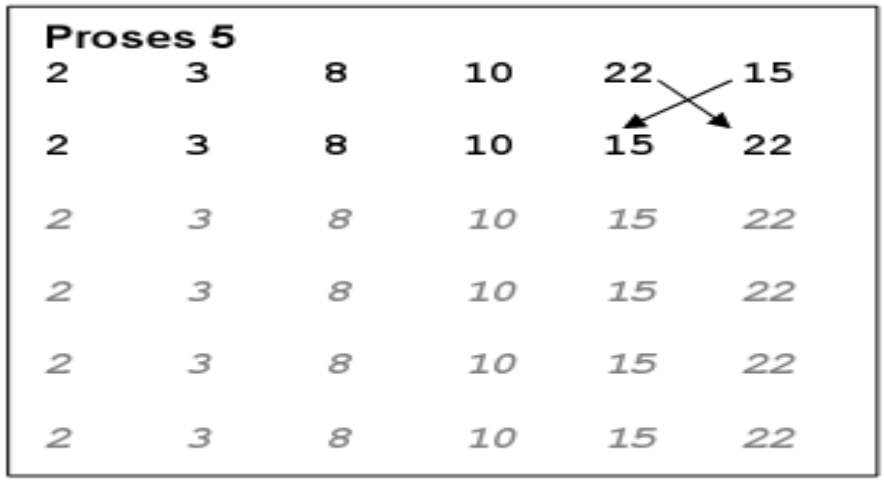
2	3	8	22	10	15
2	3	8	22	10	15
2	3	8	10	22	15
2	*3*	*8*	*10*	*22*	*15*
2	*3*	*8*	*10*	*22*	*15*
2	*3*	*8*	*10*	*22*	*15*

→ Tidak ada penukaran, karena $10 < 15$

→ Pegurutan berhenti di sini!

BUBBLE SORTING (Lanjutan)

Langkah Penyelesaian *BUBBLE SORT*



Pegurutan berhenti di sini!

BUBBLE SORTING (Lanjutan)

Contoh program:

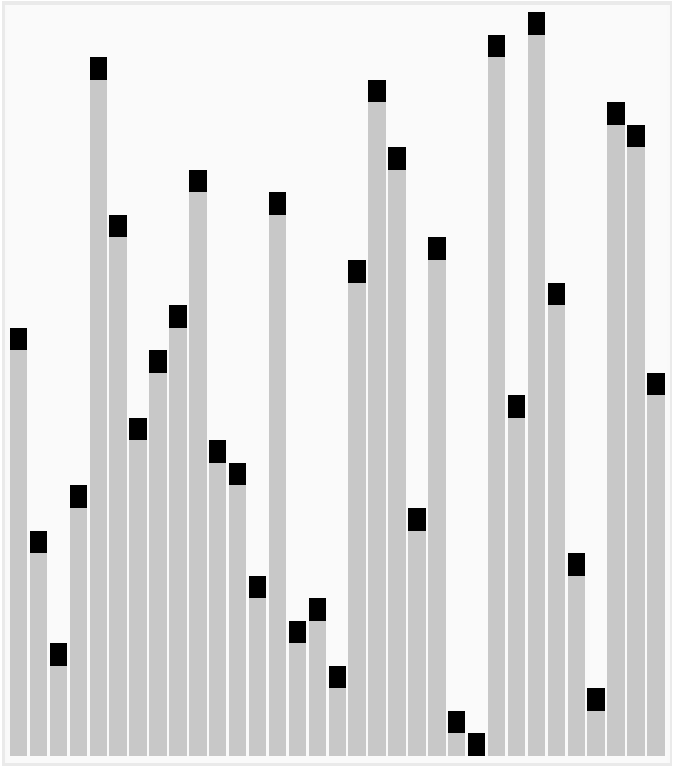
```
def BubbleSort(X):
    for i in range(len(X)-1,0,-1):
        Max=0
        for l in range(1,i+1):
            if X[l]>X[Max]:
                Max = l
        temp = X[i]
        X[i] = X[Max]
        X[Max] = temp

Hasil = [22,10,15,3,8,2]
BubbleSort(Hasil)
print(Hasil)
```

Hasil program:

```
[2, 3, 8, 10, 15, 22]
```

QUICK SORTING

- Merupakan metode sort tercepat
- Quicksort diperkenalkan oleh C.A.R. Hoare. Quicksort partition exchange sort, karena konsepnya membuat bagian-bagian, dan sort dilakukan perbagian.
- Pada algoritma quick sort, pemilihan pivot merupaka hal yang menentukan apakah algoritma quicksort tersebut akan memberikan performa terbaik atau terburuk (Nugraheny, 2018).



QUICK SORTING (Lanjutan)

Misal ada N elemen dalam keadaan urut turun, adalah mungkin untuk mengurutkan N elemen tersebut dengan N/2 kali, yakni pertama kali menukarkan elemen paling kiri dengan paling kanan, kemudian secara bertahap menuju ke elemen yang ada di tengah. Tetapi hal ini hanya bisa dilakukan jika tahu pasti bahwa urutannya adalah urut turun.

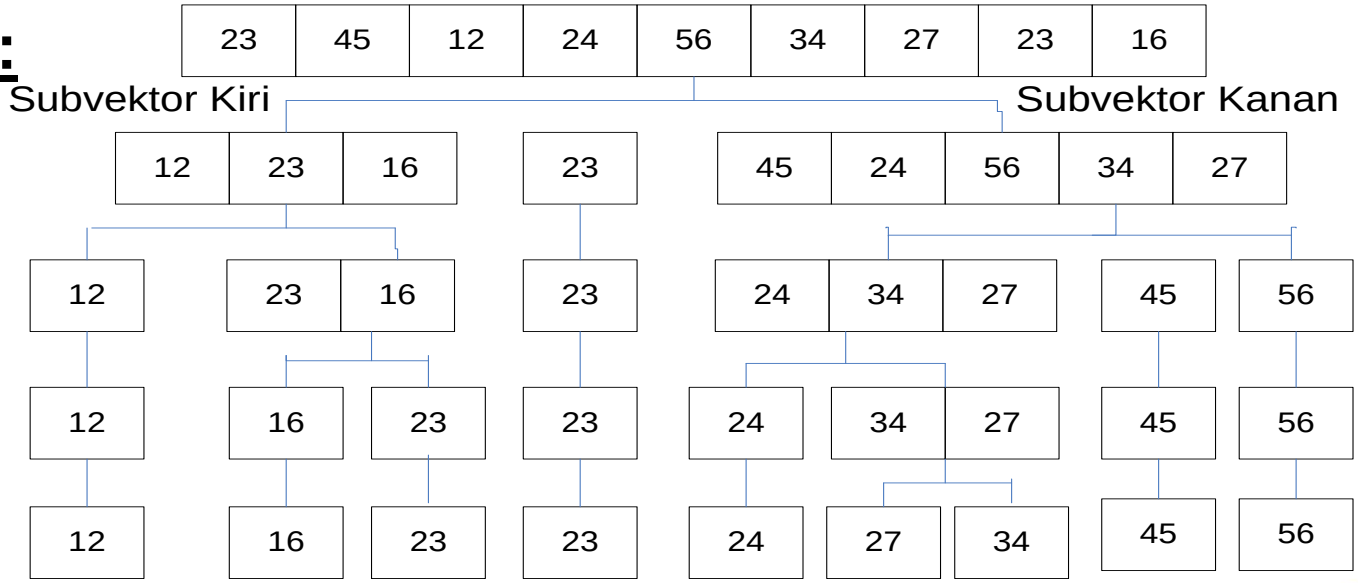
Secara garis besar metode ini dijelaskan sebagai berikut,
Misal: akan mengurutkan vektor A yang mempunyai N elemen. Pilih sembarang dari vektor tsb, biasanya elemen pertama misalnya X. Kemudian semua elemen tersebut disusun dengan menempatkan X pada posisi J sedemikian rupa sehingga elemen ke 1 sampai ke j-1 mempunyai nilai lebih kecil dari X dan elemen ke J+1 sampai ke N mempunyai nilai lebih besar dari X.

QUICK SORTING (Lanjutan)

Dengan demikian mempunyai dua buah subvektor, subvektor pertama nilai elemennya lebih kecil dari X, subvektor kedua nilai elemennya lebih besar dari X.

Pada langkah berikutnya, proses diatas diulang pada kedua subvektor, sehingga akan mempunyai empat subvektor. Proses diatas diulang pada setiap subvektor sehingga seluruh vektor semua elemennya menjadi terurutkan.

Contoh 1:



QUICK SORTING (Lanjutan)

Contoh 2

Pilih vektor X → elemen pertama

Iterasi 1

22 10 15 3 8 2

22

10 15 3 8 2 22

QUICK SORTING (Lanjutan)

Iterasi 2 Pilih lagi vektor X berikutnya

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

Iterasi 3 Pilih lagi vektor X berikutnya

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

INSERTION SORT

- Pengurutan data yang membandingkan data dengan dua elemen data pertama, kemudian membandingkan elemen-elemen data yang sudah diurutkan, kemudian perbandingan atara data tersebut akan terus diulang hingga tidak ada elemen data yang tersisa (Rahayuningsih, 2016).

- Mirip dengan cara **mengurutkan** kartu, perlembar yang diambil & **disisipkan** (insert) ke tempat yang seharusnya**.**

INSERTION SORT (Lanjutan)

Prinsip Kerja Insertion Sort adalah:

1. Pengecekan mulai dari data ke-1 sampai data ke-n
2. Bandingkan data ke-I (I = data ke-2 s/d data ke-n)
3. Bandingkan data ke-I tersebut dengan data sebelumnya (I-1), Jika lebih kecil maka data tersebut dapat disisipkan ke data awal sesuai dengan posisisi yang seharusnya
4. Lakukan langkah 2 dan 3 untuk bilangan berikutnya (I=I+1) sampai didapatkan urutan yang optimal.

INSERTION SORT (Lanjutan)

Contoh :	22	10	15	3	8	2

Iterasi 1

	1	2	3	4	5	6
Langkah 1:	22	10	15	3	8	2
Langkah 2:	22	10	15	3	8	2
Langkah 3:	10	22	15	3	8	2
Langkah 4:	Ulangi langkah 2 dan 3					

INSERTION SORT (Lanjutan)

Iterasi 2

Langkah 1:	10	22	15	3	8	2
Langkah 2:	10	22	15	3	8	2
Langkah 3:	10	15	22	3	8	2
Langkah 4:	Ulangi langkah 2 dan 3					

Lakukan Iterasi selanjutnya sampai iterasi ke- 6

Catatan : Setiap ada pemindahan, maka elemen yang sudah ada akan di-insert sehingga akan bergeser ke belakang.

INSERTION SORT (Lanjutan)

Ilustrasi

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

INSERTION SORT (Lanjutan)

Contoh program:

```
def InsertionSort(val):
    for index in range(1,len(val)):
        a = val[index]
        b = index
        while b>0 and val[b-1]>a:
            val[b]=val[b-1]
            b = b-1
        val[b]=a

Angka = [22,10,15,3,8,2]
InsertionSort(Angka)
print(Angka)
```

Hasil program:

```
[2, 3, 8, 10, 15, 22]
```

MERGE SORT

- Menggabungkan dua array yang sudah terurut (Utami, 2017)

Prinsip Kerja Merge Sort adalah :

- Kelompokkan deret bilangan kedalam 2 bagian, 4 bagian, 8 bagian, ...dst → (2n)
- Uruntkan secara langsung bilangan dalam kelompok tersebut.
- Lakukan langkah diatas untuk kondisi bilangan yang lain sampai didapatkan urutan yang optimal.

MERGE SORT (Lanjutan)

Contoh :	22	10	15	3	8	2

Iterasi 1

	1	2	3	4	5	6
Langkah 1 :	22	10	15	3	8	2
Langkah 2 :	10	22	3	15	2	8

Iterasi 2

Langkah 1 :	10	22	3	15	2	8
Langkah 2 :	3	10	15	22	2	8

MERGE SORT (Lanjutan)

Iterasi 3

Langkah 1 : 3 10 15 22 2 8

Langkah 2 : 2 3 8 10 15 22

MERGE SORT (Lanjutan)

Ilustrasi

	22	10	15	3	8	2
1	22	10	15	3	8	2
	10	22	3	15	2	8
2	10	22	3	15	2	8
	3	10	15	22	2	8
3	3	10	15	22	2	8
	2	3	8	10	15	22

MERGE SORT (Lanjutan)

```
def mergeSort(X):
    print("Bilangan diurutkan ",X)
    if len(X)>1:
        mid = len(X)//2
        lefthalf = X[:mid]
        righthalf = X[mid:]
        mergeSort(lefthalf)
        mergeSort(righthalf)
        i=j=k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                X[k]=lefthalf[i]
                i=i+1
            else:
                X[k]=righthalf[j]
                j=j+1
            k=k+1
while i < len(lefthalf):
            X[k]=lefthalf[i]
            i=i+1
            k=k+1
        while j < len(righthalf):
            X[k]=righthalf[j]
            j=j+1
            k=k+1
        print("Merging ",X)
X = [22,10,15,3,8,2]
mergeSort(X)
print(X)
```

Hasil Program:

```
Bilangan diurutkan  [22, 10, 15, 3, 8, 2]
Bilangan diurutkan  [22, 10, 15]
Bilangan diurutkan  [22]
Bilangan diurutkan  [10, 15]
Bilangan diurutkan  [10]
Bilangan diurutkan  [15]
Merging  [10, 15]
Bilangan diurutkan  [3, 8, 2]
Bilangan diurutkan  [3]
Bilangan diurutkan  [8, 2]
Bilangan diurutkan  [8]
Bilangan diurutkan  [2]
Merging  [2, 3, 8]
[2, 3, 8, 10, 15, 22]
```

KESIMPULAN METODE SORTING

- Bubble sort membutuhkan waktu komputasi paling lama.
- Quick sort dan Merge sort yang paling cepat, tetapi Quick sort lebih cepat daripada Merge sort.
- Insertion sort dan Selection sort memilki kompleksitas yang sama dengan Bubble sort, tetapi waktunya lebih cepat.

Tugas Kelompok

1. Buatlah pengurutan dari data 29, 27, 10, 8, 76, 21 dengan metode sorting Selection Sort, Bubble Sort, Merge Sort, Quick Sort dan Insertion Sort.

2. Soal dikerjakan secara berkelompok dengan memberikan iterasi secara detail