



YAYASAN PERGURUAN CIKINI
INSTITUT SAINS DAN TEKNOLOGI NASIONAL
FAKULTAS SAINS DAN TEKNOLOGI INFORMASI

Jl. Moh. Kahfi II, Bhumi Srengseng Indah, Jagakarsa, Jakarta Selatan 12640
Telp. 021-7270090 (hunting), Fax. 021-7866955, hp: 081291030024
Email: fsti@istn.ac.id Website: www.istn.ac.id

SURAT PENUGASAN TENAGA PENDIDIK
Nomor : 204 / 03.1 – I / IX / 2022
SEMESTER GANJIL, TAHUN AKADEMIK 2022 / 2023

| Nama | : Aryo Nur Utomo, ST.,M.Kom. | Status Pegawai | : Edukatif Tetap | | |
|---|---|------------------|--------------------|-----------------|----------------------|
| NIK | : 01.121225 | Program Studi | : Sistem Informasi | | |
| Jabatan Akademik | : Asiste Ahli | | | | |
| Bidang | Perincian Kegiatan | Ruang/ Tempat | Jam/ Minggu | Kredit (sks) | Hari / Waktu |
| I PENDIDIKAN Dan PENGAJARAN | MENGAJAR DI KELAS (KULIAH / RESPONSI DAN LABORATORIUM) | | | | |
| | 1. Cloud Computing (SI) | A-1 | 1 Jam/Minggu | 1 | Senin / 08:00-09:40 |
| | 2. Sistem Pendukung Keputusan (SI) | D-2 | 1,5 Jam/Minggu | 1,5 | Selasa / 08:00-10:00 |
| | 3. Sistem Temu Kembali Informasi(SI) | E-4 | 1 Jam/Minggu | | Senin /15:30-17:00 |
| | 4. IT Service Management (SI) | D-3 | 1 Jam/Minggu | 1 | Jum'at / 14:30-16:00 |
| | 5. Keamanan Sistem Informasi (SI) | E-1 | 1 Jam/Minggu | 1 | Rabu / 15:30-17:00 |
| | 6. Algoritma dan Pemrograman (TIF) | A-2 | 1 Jam/Minggu | 1 | Kamis / 08:00-09:40 |
| | 7. Analisis dan Perancangan Algoritma (TIF) | A-1 | 1 Jam/Minggu | 1 | Senin / 10.30-12.10 |
| | 8. Pemrograman Jaringan (Java/Python) | D-2 | 1,5 Jam/Minggu | 1,5 | Selasa / 14.41-15.40 |
| | 9. Pembelajaran Mesin (TIF) | E-1 | 1,5 Jam/Minggu | 1,5 | Rabu / 13.00-14.40 |
| | 10. Komputer Forensik (TIF) | A-2 | 1,5 Jam/Minggu | 1,5 | Jumat / 12.30-14.00 |
| | 11. Pengelolaan Layanan TI (ITSM) (TIF) | A-1 | 1 Jam/Minggu | 1 | Senin / 08.00-09.40 |
| | 12. Data Compress & Coding (PIGS) | A-2 | 1 Jam/Minggu | | Selasa / 08.00-09.40 |
| | 13. Manajemen Proyek Perangkat Lunak | A-3 | 1,5 Jam/Minggu | | Kamis / 10.00-11.30 |
| 14. Menduduki Jabatan Struktural (Ka.Prodi TIF) | | | 20 Jam/Minggu | 3 | |
| II PENELITIAN | Penulisan Karya Ilmiah | | | 1 | |
| III PENGABDIAN DAN MASYARAKAT | Pelatihan dan Penyuluhan | | | | |
| IV UNSUR-UNSUR PENUNJANG | Berperan Serta Aktif dalam Pertemuan Ilmiah/Seminar | | | | |
| Jumlah Total | | | | 16 | |

Kepada yang bersangkutan akan diberikan gaji / honorarium sesuai dengan peraturan penggajian yang berlaku di Institut Sains Dan Teknologi Nasional
Penugasan ini berlaku dari tanggal **02 September 2022** sampai dengan tanggal **29 Februari 2023**.

Jakarta, 30 September 2022
Dekah,

(Marnaeni, S.Kom.,M.Kom.)



Tembusan :

1. Direktur Akademik – ISTN
2. Direktur Non Akademik – ISTN
3. Ka. Biro Sumber Daya Manusia – ISTN
4. Kepala Program Studi Sistem Informasi
5. Arsip.

DAFTAR HADIR PESERTA KULIAH MAHASISWA**GANJIL REGULER TAHUN 2022/2023**

FAK / JURUSAN : Teknik Informatika S1 HARI/TANGGAL : Senin
MATA KULIAH : Analisa & Perancangan Algoritma
KELAS / PESERTA : A / 9 JAM KULIAH : 10.30-12.10
KURIKULUM : 2018
DOSEN : Aryo Nur Utomo, ST.M.Kom. RUANG :

| NO | NIM | NAMA MAHASISWA | TANGGAL PERTEMUAN | | | | | | | | JUMLAH |
|----|----------|-------------------------------|-------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|--------|
| | | | 21/ 11 | 28/ 11 | 05/ 12 | 12/ 12 | 19/ 12 | 26/ 12 | 02/ 01 | 16/ 01 | |
| 1 | 20360016 | Rakha Arya Ramadhan | √ | √ | √ | √ | √ | √ | √ | U | 7 |
| 2 | 21360001 | Muhamad Azlan Nugraha Firdaus | √ | √ | √ | √ | √ | √ | √ | U | 7 |
| 3 | 21360002 | Afzal Fazli Mawla Aziz | √ | √ | √ | √ | √ | √ | √ | U | 7 |
| 4 | 21360003 | Muhammad Adzka Aulia | √ | √ | √ | √ | √ | √ | √ | U | 7 |
| 5 | 21360004 | Muhammad Reza Pahlevi | √ | √ | √ | √ | √ | √ | √ | U | 7 |
| 6 | 21360005 | Rangga Dwi Prilian | √ | √ | √ | √ | √ | √ | √ | U | 7 |
| 7 | 21360006 | Muhamad Ariiq Athomufid | √ | √ | √ | √ | √ | √ | √ | U | 7 |
| 8 | 21360007 | Sepansya Aria Muhammad Asfian | √ | √ | √ | √ | √ | √ | √ | U | 7 |
| 9 | 21360011 | Syahru Rahmana Putra | √ | √ | √ | √ | √ | √ | √ | U | 7 |

Jakarta , Februari 2023

Dosen Pengajar

(Aryo Nur Utomo, ST.M.Kom)



BERITA ACARA PERKULIAHAN
(PRESENTASI KEHADIRAN DOSEN)
SEMESTER GANJIL TAHUN AKADEMIK 2022/2023
PROGRAM STUDI TEKNIK INFORMATIKA S1 FSTI-ISTN

Mata Kuliah : Analisis dan Perancangan Algo
Dosen : Aryo Nur Utomo, ST, M.Kom
Hari : Senin
Jam : 10:30-12:10

Semester : 363002
SKS : 2
Kelas : A
Ruang : B-2

| No. | TANGGAL | MATERI KULIAH | JML MHS HADIR | TANDA TANGAN DOSEN |
|-----|------------------|--|---------------|--------------------|
| 9. | 21-November 2022 | Algoritma Divide and Conquer bagian-1. | 9 | Ah |
| 10. | 28-November 2022 | Algoritma Divide and Conquer bagian-2. | 9 | Ah |
| 11. | 5-Desember 2022 | Algoritma RunutBalik (Backtracking). | 9 | Ah |
| 12. | 12-Desember 2022 | Algoritma Branch and Bound. | 9 | Ah |
| 13. | 19-Desember 2022 | Program Dinamis. | 9 | Ah |
| 14. | 26-Desember 2022 | String Matching. String Matching dgn Java. | 9 | Ah |
| 15. | 2-Januari 2023 | Proyek Kecil – Pustaka Polinom dgn Divide Conquer. | 9 | Ah |
| 16. | 16-Januari 2023 | UJIAN AKHIR SEMESTER (UAS) | 9 | Ah |

DOSEN PENGAJAR

(Aryo Nur Utomo, ST. M.Kom)

DAFTAR NILAI

SEMESTER GANJIL REGULER TAHUN 2022/2023

Program Studi : Teknik Informatika S1

Matakuliah : Analisis dan Perancangan Algoritma

Kelas / Peserta : A

Perkuliahan : Kampus ISTN Bumi Srengseng Indah

Dosen : 1. Siti Madinah Ladjamuddin., S.Kom. M.Kom.
2. Aryo Nur Utomo, ST.M.Kom

Hal. 1/1

| No | NIM | N A M A | ABSEN | TUGAS | UTS | UAS | MODEL | PRESENTASI | NA | HURUF |
|----|----------|-------------------------------|-------|-------|-----|-----|-------|------------|------|-------|
| | | | 10% | 0% | 30% | 30% | 30% | 0% | | |
| 1 | 20360016 | Rakha Arya Ramadhan | 100 | 0 | 73 | 72 | 70 | 0 | 74.5 | B+ |
| 2 | 21360001 | Muhamad Azlan Nugraha Firdaus | 100 | 0 | 79 | 77 | 70 | 0 | 77.8 | A- |
| 3 | 21360002 | Afzal Fazli Mawla Aziz | 100 | 0 | 77 | 78 | 70 | 0 | 77.5 | A- |
| 4 | 21360003 | Muhammad Adzka Aulia | 100 | 0 | 65 | 0 | 0 | 0 | 0 | |
| 5 | 21360004 | Muhammad Reza Pahlevi | 100 | 0 | 79 | 79 | 70 | 0 | 78.4 | A- |
| 6 | 21360005 | Rangga Dwi Prilian | 100 | 0 | 78 | 79 | 70 | 0 | 78.1 | A- |
| 7 | 21360006 | Muhamad Ariiq Athomufid | 100 | 0 | 5 | 0 | 0 | 0 | 0 | |
| 8 | 21360007 | Sepansya Aria Muhammad Asfian | 100 | 0 | 78 | 79 | 70 | 0 | 78.1 | A- |
| 9 | 21360011 | Syahru Rahmana Putra | 100 | 0 | 73 | 55 | 70 | 0 | 69.4 | B |

| Rekapitulasi Nilai | | | | | | | |
|--------------------|---|----|---|----|---|----|---|
| A | 0 | B+ | 1 | C+ | 0 | D+ | 0 |
| A- | 5 | B | 1 | C | 0 | D | 0 |
| | | B- | 0 | C- | 0 | E | 0 |

Jakarta, 12 February 2023

— Dosen Pengajar



1. Siti Madinah L., S.Kom. M.Kom.



2. Aryo Nur Utomo, ST.M.Kom.

Algoritma *Divide and Conquer*

Analisis & Perancangan Algoritma

Algoritma *Divide and Conquer* **(Bagian 1)**

- *Divide and Conquer* dulunya adalah strategi militer yang dikenal dengan nama *divide ut imperes* (pecah belah).
- Sekarang strategi tersebut menjadi strategi fundamental di dalam ilmu komputer dengan nama *Divide and Conquer*.

Definisi

- *Divide*: membagi masalah menjadi beberapa upa-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil (idealnya berukuran hampir sama),
- *Conquer*: memecahkan (menyelesaikan) masing-masing upa-masalah (secara rekursif), dan
- *Combine*: menggabungkan solusi masing-masing upa-masalah sehingga membentuk solusi masalah semula.

- Obyek permasalahan yang dibagi :
masuk (*input*) atau *instances* yang berukuran n seperti:
 - tabel (larik),
 - matriks,
 - eksponen,
 - dll, bergantung pada masalahnya.
- Tiap-tiap upa-masalah mempunyai karakteristik yang sama (*the same type*) dengan karakteristik masalah asal, sehingga metode *Divide and Conquer* lebih natural diungkapkan dalam skema rekursif.

Skema Umum Algoritma *Divide and Conquer*

```
procedure DIVIDE_and_CONQUER(input n : integer)
```

```
{ Menyelesaikan masalah dengan algoritma D-and-C.
```

```
  Masukan: masukan yang berukuran n
```

```
  Keluaran: solusi dari masalah semula
```

```
}
```

Deklarasi

```
  r, k : integer
```

Algoritma

```
if n ≤ n0 then {ukuran masalah sudah cukup kecil }
```

```
  SOLVE upa-masalah yang berukuran n ini
```

```
else
```

```
  Bagi menjadi r upa-masalah, masing-masing berukuran n/k
```

```
  for masing-masing dari r upa-masalah do
```

```
    DIVIDE_and_CONQUER(n/k)
```

```
  endfor
```

```
  COMBINE solusi dari r upa-masalah menjadi solusi masalah semula }
```

```
endif
```

Jika pembagian selalu menghasilkan dua upa-masalah yang berukuran sama:

```
procedure DIVIDE_and_CONQUER(input n : integer)  
{ Menyelesaikan masalah dengan algoritma D-and-C.  
  Masukan: masukan yang berukuran n  
  Keluaran: solusi dari masalah semula  
}
```

Deklarasi

```
r, k : integer
```

Algoritma

```
if n ≤ n0 then {ukuran masalah sudah cukup kecil }  
  SOLVE upa-masalah yang berukuran n ini  
else  
  Bagi menjadi 2 upa-masalah, masing-masing berukuran n/2  
  DIVIDE_and_CONQUER(upa-masalah pertama yang berukuran n/2)  
  DIVIDE_and_CONQUER(upa-masalah kedua yang berukuran n/2)  
  COMBINE solusi dari 2 upa-masalah  
endif
```

$$T(n) = \begin{cases} g(n) & , n \leq n_0 \\ 2T(n/2) + f(n) & , n > n_0 \end{cases}$$

Contoh-contoh masalah

1. Mencari Nilai Minimum dan Maksimum (MinMaks)

Persoalan: Misalkan diberikan tabel A yang berukuran n elemen dan sudah berisi nilai *integer*.

Carilah nilai minimum dan nilai maksimum sekaligus di dalam tabel tersebut.

Penyelesaian dengan *Algoritma Brute Force*

```
procedure MinMaks1(input A : TabelInt, n : integer,  
                  output min, maks : integer)  
{ Mencari nilai minimum dan maksimum di dalam tabel A yang berukuran n  
  elemen, secara brute force.  
Masukan: tabel A yang sudah terdefinisi elemen-elemennya  
Keluaran: nilai maksimum dan nilai minimum tabel  
}  
Deklarasi  
  i : integer  
  
Algoritma:  
  min ← A1 { inisialisasi nilai minimum }  
  maks ← A1 { inisialisasi nilai maksimum }  
  for i ← 2 to n do  
    if Ai < min then  
      min ← Ai  
    endif  
    if Ai > maks then  
      maks ← Ai  
    endif  
  
  endfor
```

$$T(n) = (n - 1) + (n - 1) = 2n - 2 = O(n)$$

Penyelesaian dengan *Divide and Conquer*

Contoh 4.1. Misalkan tabel A berisi elemen-elemen sebagai berikut:

4 12 23 9 21 1 35 2 24

Ide dasar algoritma secara *Divide and Conquer*:

4 12 23 9 21 1 35 2 24

DIVIDE

4 12 23 9 21 1 35 2 24

SOLVE: tentukan min & maks pada tiap bagian

4 12 23 9 21 1 35 2 24

min = 4
maks = 23

min = 1
maks = 35

COMBINE

4 12 23 9 21 1 35 2 24

min = 1
maks = 35

Combine
membandingkan :
Min → 4 & **1**
Maks → 23 & **35**

- Ukuran tabel hasil pembagian dapat dibuat cukup kecil sehingga mencari minimum dan maksimum dapat diselesaikan (SOLVE) secara lebih mudah.
- Dalam hal ini, ukuran kecil yang dipilih adalah 1 elemen atau 2 elemen.

MinMaks(A, n, min, maks)

Algoritma:

1. Untuk kasus $n = 1$ atau $n = 2$,
SOLVE: Jika $n = 1$, maka $min = maks = A[n]$
Jika $n = 2$, maka bandingkan kedua elemen untuk menentukan min dan $maks$.

2. Untuk kasus $n > 2$,
 - (a) DIVIDE: Bagi dua tabel A menjadi dua bagian yang sama, A1 dan A2

 - (b) CONQUER:
MinMaks(A1, n/2, min1, maks1)
MinMaks(A2, n/2, min2, maks2)

 - (c) COMBINE:
if min1 < min2 then min <- min1 else min <- min2
if maks1 < maks2 then maks <- maks2 else maks <- maks1

Contoh 4.2. Tinjau kembali Contoh 4.1 di atas.

DIVIDE dan CONQUER:

4 12 23 9 21 1 35 2 24

4 12 23 9 21 1 35 2 24

4 12 23 9 21 1 35 2 24

SOLVE dan COMBINE:

| | | | | |
|----------------|----------------|----------------|-----------|----------------|
| <u>4 12</u> | <u>23 9</u> | <u>21 1</u> | <u>35</u> | <u>2 24</u> |
| min = 4 | min = 9 | min = 1 | min = 35 | min = 2 |
| maks = 12 | maks = 23 | maks = 21 | maks = 35 | maks = 24 |

| | | |
|---------------------------|----------------|----------------------|
| <u>4 12 23 9</u> | <u>21 1</u> | <u>35 2 24</u> |
| min = 4 | min = 1 | min = 2 |
| maks = 23 | maks = 21 | maks = 35 |

| | |
|---------------------------|---------------------------------|
| <u>4 12 23 9</u> | <u>21 1 35 2 24</u> |
| min = 4 | min = 1 |
| maks = 23 | maks = 35 |

| |
|--|
| <u>4 12 23 9 21 1 5 2 24</u> |
| min = 1 |
| maks = 35 |

```

procedure MinMaks2(input A : TabelInt, i, j : integer,
                    output min, maks : integer)
{ Mencari nilai maksimum dan minimum di dalam tabel A yang berukuran n
  elemen secara Divide and Conquer.
  Masukan: tabel A yang sudah terdefinisi elemen-elemennya
  Keluaran: nilai maksimum dan nilai minimum tabel
}

```

Deklarasi

```

    min1, min2, maks1, maks2 : integer

```

Algoritma:

```

if i=j then                                { 1 elemen }
    min←Ai
    maks←Ai
else
    if (i = j-1) then                          { 2 elemen }
        if Ai < Aj then
            maks←Aj
            min←Ai
        else
            maks←Ai
            min←Aj
        endif
    else                                        { lebih dari 2 elemen }
        k←(i+j) div 2                            { bagidua tabel pada posisi k }
        MinMaks2(A, i, k, min1, maks1)           { rekursi }
        MinMaks2(A, k+1, j, min2, maks2)
        if min1 < min2 then
            min←min1
        else
            min←min2
        endif

        if maks1 < maks2 then
            maks←maks2
        else
            maks←maks2
        endif
    endif

```

Kompleksitas waktu asimptotik:

$$T(n) = \begin{cases} 0 & , n = 1 \\ 1 & , n = 2 \\ 2T(n/2) + 2 & , n > 2 \end{cases}$$

Penyelesaian:

Asumsi: $n = 2^k$, dengan k bilangan bulat positif, maka

$$\begin{aligned} T(n) &= 2T(n/2) + 2 \\ &= 2(2T(n/4) + 2) + 2 = 4T(n/4) + 4 + 2 \\ &= 4T(2T(n/8) + 2) + 4 + 2 = 8T(n/8) + 8 + 4 + 2 \\ &= \dots \\ &= 2^{k-1} T(2) + \sum_{i=1}^{k-1} 2^i \\ &= 2^{k-1} \cdot 1 + 2^k - 2 \\ &= n/2 + n - 2 \\ &= 3n/2 - 2 \\ &= O(n) \end{aligned}$$

- MinMaks1 secara *brute force* :

$$T(n) = 2n - 2$$

- MinMaks2 secara *divide and conquer*:

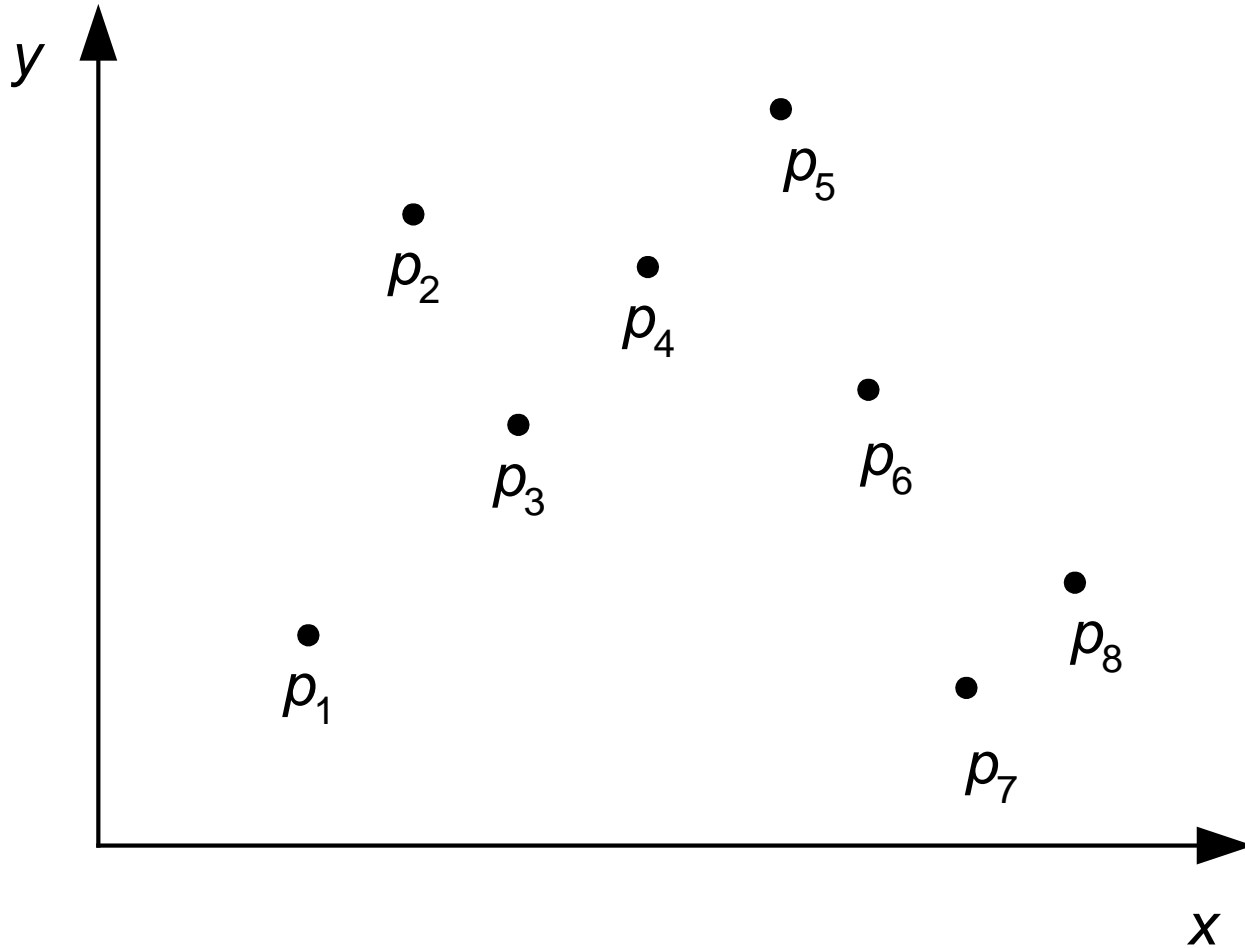
$$T(n) = 3n/2 - 2 = 1.5n - 2$$

- Perhatikan: $3n/2 - 2 < 2n - 2$, $n \geq 2$.

- Kesimpulan: algoritma MinMaks lebih mangkus dengan metdoe *Divide and Conquer*.

2. Mencari Pasangan Titik yang Jaraknya Terdekat (*Closest Pair*)

Persoalan: Diberikan himpunan titik, P , yang terdiri dari n buah titik, (x_i, y_i) , pada bidang 2-D. Tentukan jarak terdekat antara dua buah titik di dalam himpunan P .



Jarak dua buah titik $p_1 = (x_1, y_1)$ dan $p_2 = (x_2, y_2)$:

Rumus Euclidean

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Penyelesaian dengan Algoritma Brute Force

- Hitung jarak setiap pasang titik. Ada sebanyak

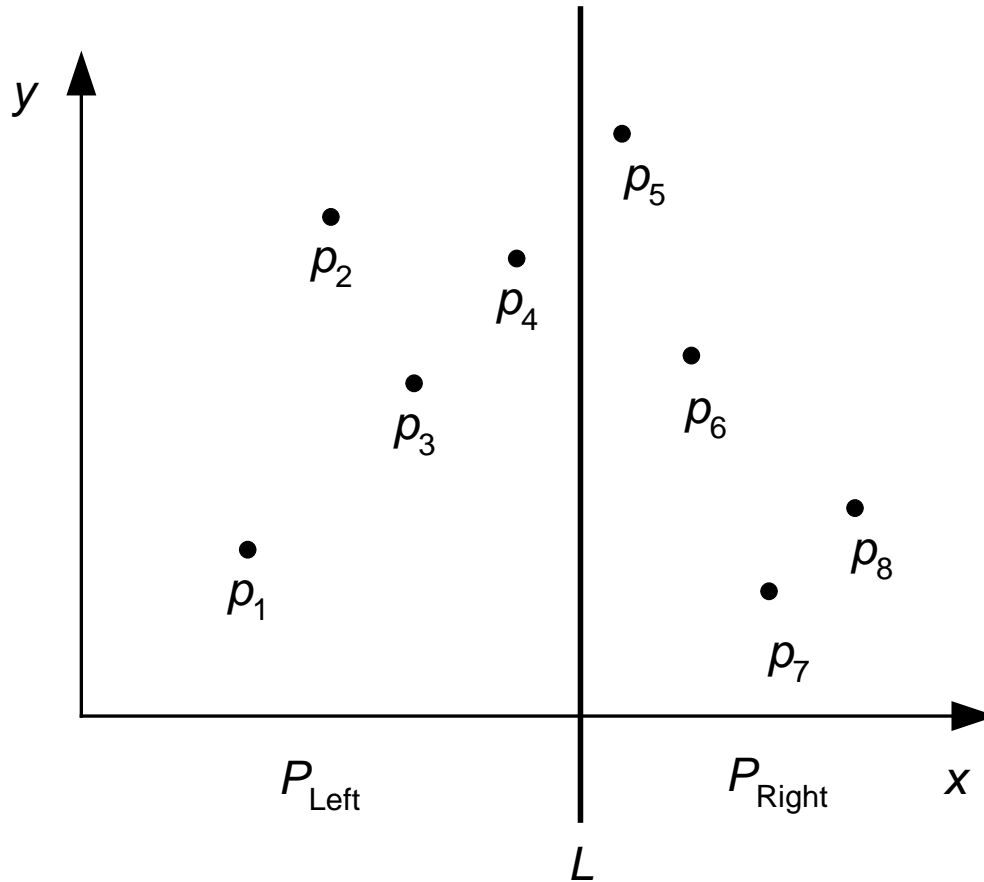
$$C(n, 2) = n(n - 1)/2 \text{ pasangan titik}$$

- Pilih pasangan titik yang mempunyai jarak terkecil.
- Kompleksitas algoritma adalah $O(n^2)$.

Penyelesaian dengan Divide and Conquer

- Asumsi: $n = 2^k$ dan titik-titik diurut berdasarkan absis (x).
- Algoritma *Closest Pair* :
 1. SOLVE: jika $n = 2$, maka jarak kedua titik dihitung langsung dengan rumus Euclidean.

2. DIVIDE: Bagi himpunan titik ke dalam dua bagian, P_{left} dan P_{right} , setiap bagian mempunyai jumlah titik yang sama.



3. CONQUER: Secara rekursif, terapkan algoritma *D-and-C* pada masing-masing bagian.
4. Pasangan titik yang jaraknya terdekat ada tiga kemungkinan letaknya:
 - (a) Pasangan titik terdekat terdapat di bagian P_{Left} .
 - (b) Pasangan titik terdekat terdapat di bagian P_{Right} .
 - (c) Pasangan titik terdekat dipisahkan oleh garis batas L , yaitu satu titik di P_{Left} dan satu titik di P_{Right} .

Jika kasusnya adalah (c), maka lakukan tahap COMBINE untuk mendapatkan jarak dua titik terdekat sebagai solusi persoalan semula.

```

procedure FindClosestPair2(input P: SetOfPoint, n : integer,
                             output delta : real)
{ Mencari jarak terdekat sepasang titik di dalam himpunan P. }

```

Deklarasi:

```
DeltaLeft, DeltaRight : real
```

Algoritma:

```

if n = 2 then
    delta ← jarak kedua titik dengan rumus Euclidean
else
    P-Left ← {p1, p2 , . . . . , pn/2 }
    P-Right ← {pn/2+1, pn/2+2 , . . . . , pn }
    FindClosestPair2(P-Left, n/2, DeltaLeft)
    FindClosestPair2(P-Right, n/2, DeltaRight)
    delta ← minimum(DeltaLeft, DeltaRight)
    {--*****--}
    Tentukan apakah terdapat titik pl di P-Left dan pr di P-Right
    Dengan jarak(pl, pr) < delta. Jika ada, set delta dengan jarak
    terkecil tersebut.
    {--*****--}
endif

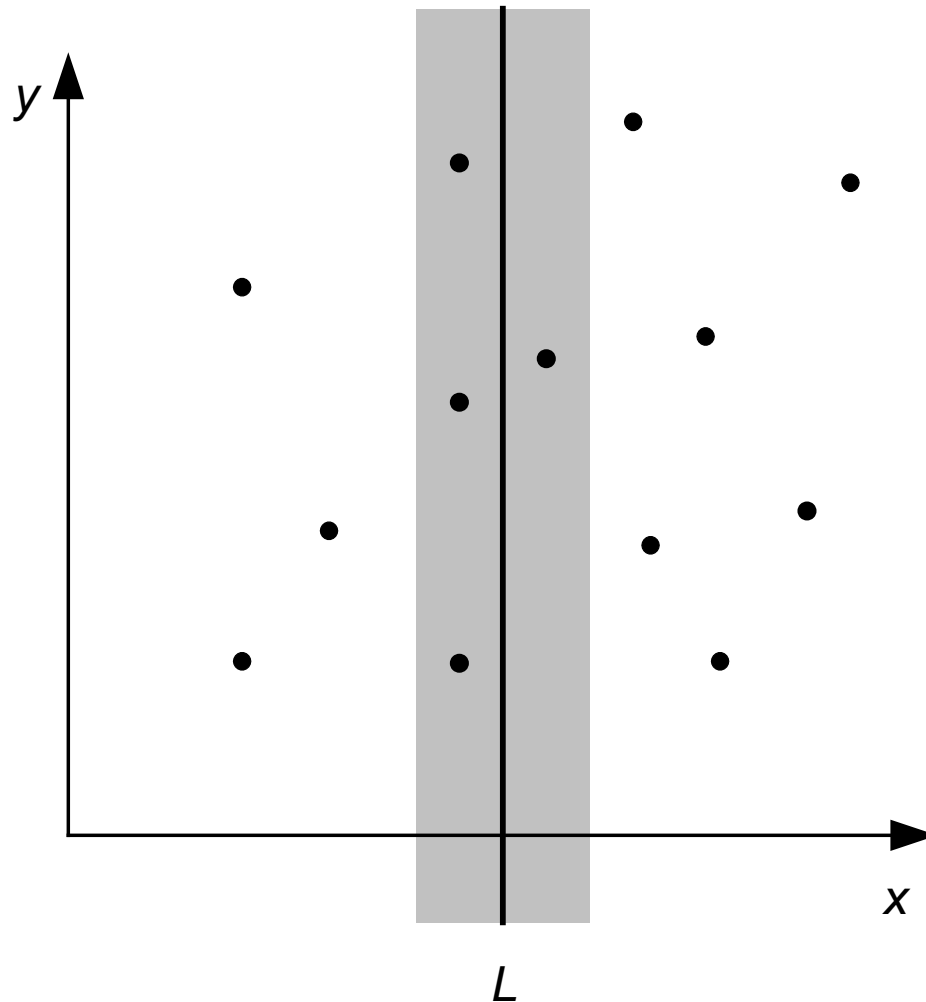
```

■ Jika terdapat pasangan titik p_l and p_r yang jaraknya lebih kecil dari *delta*, maka kasusnya adalah:

(i) Absis x dari p_l dan p_r berbeda paling banyak sebesar *delta*.

(ii) Ordinat y dari p_l dan p_r berbeda paling banyak sebesar *delta*.

- Ini berarti p_l and p_r adalah sepasang titik yang berada di daerah sekitar garis vertikal L :

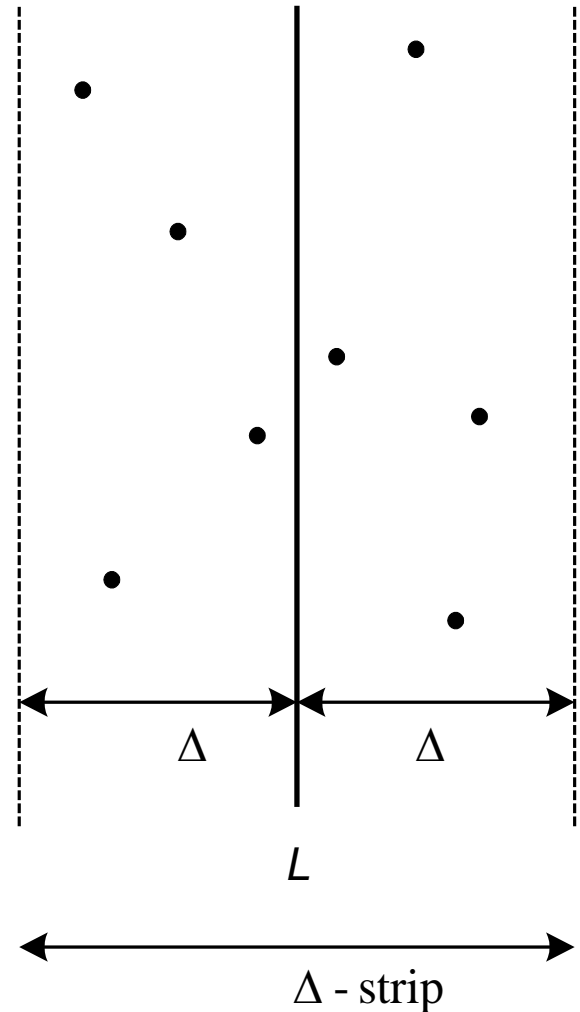


Oleh karena itu, implementasi tahap COMBINE sbb:

- (i) Temukan semua titik di P_{Left} yang memiliki absis x minimal $x_{n/2} - \delta$.
- (ii) Temukan semua titik di P_{Right} yang memiliki absis x maksimal $x_{n/2} + \delta$.

Sebut semua titik-titik yang ditemukan pada langkah (i) dan (ii) tersebut sebagai himpunan P_{strip} yang berisi s buah titik.

Urut titik-titik tersebut dalam urutan absis y yang menaik. Misalkan q_1, q_2, \dots, q_s menyatakan hasil pengurutan.



Langkah COMBINE:

```
for i←1 to s do  
  for j←i+1 to s do  
    exit when ( |qi.x - qj.x | > Delta or |qi.y - qj.y | > Delta  
    if jarak (qi, qj) < Delta then  
      Delta ← jarak(qi, qj) { dihitung dengan rumus Euclidean }  
    endif  
  endfor  
endfor
```

Kompleksitas algoritma:

$$T(n) = \begin{cases} 2T(n/2) + cn & , n > 2 \\ a & , n = 2 \end{cases}$$

Solusi dari persamaan di atas adalah $T(n) = O(n \log n)$.

3. Algoritma Pengurutan dengan Metode *Divide and Conquer*

```
procedure Sort(input/output A : TabelInt, input n : integer)  
  
{ Mengurutkan tabel A dengan metode Divide and Conquer  
  Masukan: Tabel A dengan n elemen  
  Keluaran: Tabel A yang terurut  
}  
Algoritma:  
  if Ukuran(A) > 1 then  
    Bagi A menjadi dua bagian, A1 dan A2, masing-masing berukuran n1  
    dan n2 ( $n = n1 + n2$ )  
  
    Sort(A1, n1)   { urut bagian kiri yang berukuran n1 elemen }  
    Sort(A2, n2)   { urut bagian kanan yang berukuran n2 elemen }  
  
    Combine(A1, A2, A) { gabung hasil pengurutan bagian kiri dan  
                        bagian kanan }  
  
  end
```

Contoh:

| | | | | | | | | |
|---|---|----|---|---|---|----|---|---|
| A | 4 | 12 | 3 | 9 | 1 | 21 | 5 | 2 |
|---|---|----|---|---|---|----|---|---|

Dua pendekatan (*approach*) pengurutan:

1. Mudah membagi, sulit menggabung (*easy split/hard join*)

Tabel A dibagidua berdasarkan posisi elemen:

| | | | | | | | | | | | | |
|----------------|----|--|---|----|---|---|----|--|---|----|---|---|
| <i>Divide:</i> | A1 | <table border="1"><tr><td>4</td><td>12</td><td>3</td><td>9</td></tr></table> | 4 | 12 | 3 | 9 | A2 | <table border="1"><tr><td>1</td><td>21</td><td>5</td><td>2</td></tr></table> | 1 | 21 | 5 | 2 |
| 4 | 12 | 3 | 9 | | | | | | | | | |
| 1 | 21 | 5 | 2 | | | | | | | | | |

| | | | | | | | | | | | | |
|--------------|----|--|----|---|---|----|----|--|---|---|---|----|
| <i>Sort:</i> | A1 | <table border="1"><tr><td>3</td><td>4</td><td>9</td><td>12</td></tr></table> | 3 | 4 | 9 | 12 | A2 | <table border="1"><tr><td>1</td><td>2</td><td>5</td><td>21</td></tr></table> | 1 | 2 | 5 | 21 |
| 3 | 4 | 9 | 12 | | | | | | | | | |
| 1 | 2 | 5 | 21 | | | | | | | | | |

| | | | | | | | | | | |
|-----------------|----|---|---|---|---|----|----|---|----|----|
| <i>Combine:</i> | A1 | <table border="1"><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>9</td><td>12</td><td>21</td></tr></table> | 1 | 2 | 3 | 4 | 5 | 9 | 12 | 21 |
| 1 | 2 | 3 | 4 | 5 | 9 | 12 | 21 | | | |

Algoritma pengurutan yang termasuk jenis ini:

- a.urut-gabung (*Merge Sort*)
- b.urut-sisip (*Insertion Sort*)

2. Sulit membagi, mudah menggabung (*hard split/easy join*)
Tabel A dibagidua berdasarkan nilai elemennya. Misalkan elemen-elemen $A1 \leq$ elemen-elemen $A2$.

Divide: A1

| | | | |
|---|---|---|---|
| 4 | 2 | 3 | 1 |
|---|---|---|---|

 A2

| | | | |
|---|----|---|----|
| 9 | 21 | 5 | 12 |
|---|----|---|----|

Sort: A1

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
|---|---|---|---|

 A2

| | | | |
|---|---|----|----|
| 5 | 9 | 12 | 21 |
|---|---|----|----|

Combine: A

| | | | | | | | |
|---|---|---|---|---|---|----|----|
| 1 | 2 | 3 | 4 | 5 | 9 | 12 | 21 |
|---|---|---|---|---|---|----|----|

- Algoritma pengurutan yang termasuk jenis ini:
- a. urut-cepat (*Quick Sort*)
 - b. urut-seleksi (*Selection Sort*)

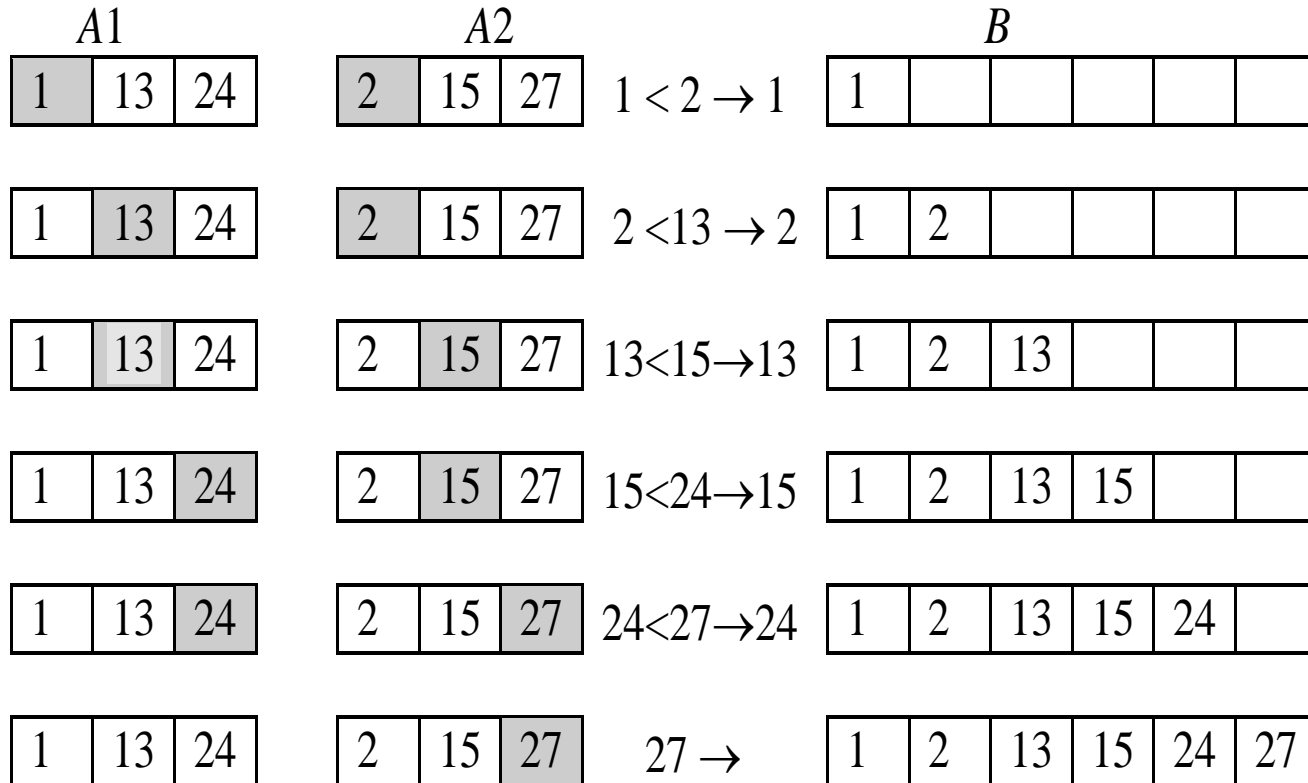
(a) Merge Sort

Algoritma:

1. Untuk kasus $n = 1$, maka tabel A sudah terurut dengan sendirinya (langkah SOLVE).

2. Untuk kasus $n > 1$, maka
 - (a) DIVIDE: bagi tabel A menjadi dua bagian, bagian kiri dan bagian kanan, masing-masing bagian berukuran $n/2$ elemen.
 - (b) CONQUER: Secara rekursif, terapkan algoritma *D-and-C* pada masing-masing bagian.
 - (c) MERGE: gabung hasil pengurutan kedua bagian sehingga diperoleh tabel A yang terurut.

Contoh Merge:



Contoh 4.3. Misalkan tabel *A* berisi elemen-elemen berikut:

4 12 23 9 21 1 5 2

DIVIDE, CONQUER, dan SOLVE:

4 12 23 9 21 1 5 2

4 12 23 9 21 1 5 2

4 12 23 9 21 1 5 2

4 12 23 9 21 1 5 2

MERGE: 4 12 9 23 1 21 2 5

4 9 12 23 1 2 5 21

1 2 4 5 9 12 21 23

```
procedure MergeSort(input/output A : TabelInt, input i, j : integer)
```

```
{ Mengurutkan tabel A[i..j] dengan algoritma Merge Sort  
  Masukan: Tabel A dengan n elemen  
  Keluaran: Tabel A yang terurut  
}
```

Deklarasi:

```
k : integer
```

Algoritma:

```
if i < j then           { Ukuran(A) > 1 }  
  k ← (i+j) div 2  
  MergeSort(A, i, k)  
  MergeSort(A, k+1, j)  
  Merge(A, i, k, j)  
endif
```

Prosedur *Merge*:

```
procedure Merge(input/output A : TabelInt, input kiri,tengah,kanan :
integer)
{ Menggabung tabel A[kiri..tengah] dan tabel A[tengah+1..kanan]
menjadi tabel A[kiri..kanan] yang terurut menaik.
Masukan: A[kiri..tengah] dan tabel A[tengah+1..kanan] yang sudah
terurut menaik.
Keluaran: A[kiri..kanan] yang terurut menaik.
}
Deklarasi
B : TabelInt
i, kidall, kidal2 : integer

Algoritma:
kidall←kiri           { A[kiri .. tengah] }
kidal2←tengah + 1    { A[tengah+1 .. kanan] }
i←kiri
while (kidall ≤ tengah) and (kidal2 ≤ kanan) do
  if  $A_{kidall} \leq A_{kidal2}$  then
     $B_i \leftarrow A_{kidall}$ 
    kidall←kidall + 1
  else
     $B_i \leftarrow A_{kidal2}$ 
    kidal2←kidal2 + 1
  endif
  i←i + 1
endwhile
{ kidall > tengah or kidal2 > kanan }

{ salin sisa A bagian kiri ke B, jika ada }
while (kidall ≤ tengah) do
   $B_i \leftarrow A_{kidall}$ 
  kidall←kidall + 1
  i←i + 1
endwhile
{ kidall > tengah }

{ salin sisa A bagian kanan ke B, jika ada }
while (kidal2 ≤ kanan) do
   $B_i \leftarrow A_{kidal2}$ 
  kidal2←kidal2 + 1
  i←i + 1
endwhile
{ kidal2 > kanan }

{ salin kembali elemen-elemen tabel B ke A }
for i←kiri to kanan do
   $A_i \leftarrow B_i$ 
endfor
{ diperoleh tabel A yang terurut membesar }
```


- Kompleksitas waktu:

Asumsi: $n = 2^k$

$T(n)$ = jumlah perbandingan pada pengurutan dua buah upatabel + jumlah perbandingan pada prosedur *Merge*

$$T(n) = \begin{cases} a & , n = 1 \\ 2T(n/2) + cn & , n > 1 \end{cases}$$

Penyelesaian:

$$\begin{aligned}T(n) &= 2T(n/2) + cn \\&= 2(2T(n/4) + cn/2) + cn = 4T(n/4) + 2cn \\&= 4(2T(n/8) + cn/4) + 2cn = 8T(n/8) + 3cn \\&= \dots \\&= 2^k T(n/2^k) + kcn\end{aligned}$$

Berhenti jika ukuran tabel terkecil, $n = 1$:

$$n/2^k = 1 \rightarrow k = {}^2\log n$$

sehingga

$$\begin{aligned}T(n) &= nT(1) + cn {}^2\log n \\&= na + cn {}^2\log n \\&= O(n {}^2\log n)\end{aligned}$$

(b) *Insertion Sort*

```
procedure InsertionSort(input/output A : TabelInt,  
                        input i, j : integer)  
{ Mengurutkan tabel A[i..j] dengan algoritma Insertion Sort.  
  Masukan: Tabel A dengan n elemen  
  Keluaran: Tabel A yang terurut  
}  
Deklarasi:  
  k : integer  
  
Algoritma:  
  if i < j then           { Ukuran(A) > 1 }  
    k ← i  
    InsertionSort(A, i, k)  
    InsertionSort(A, k+1, j)  
    Merge(A, i, k, j)  
  endif
```

Perbaikan:

```
procedure InsertionSort(input/output A : TabelInt,  
                        input i, j : integer)  
{ Mengurutkan tabel A[i..j] dengan algoritma Insertion Sort.  
  Masukan: Tabel A dengan n elemen  
  Keluaran: Tabel A yang terurut  
}
```

Deklarasi:

```
k : integer
```

Algoritma:

```
if i < j then           { Ukuran(A) > 1 }  
  k ← i  
  Insertion(A, k+1, j)  
  Merge(A, i, k, j)  
endif
```

Prosedur *Merge* dapat diganti dengan prosedur penyisipan sebuah elemen pada tabel yang sudah terurut (lihat algoritma *Insertion Sort* versi iteratif).

Contoh 4.4. Misalkan tabel *A* berisi elemen-elemen berikut:

4 12 23 9 21 1 5 2

DIVIDE, CONQUER, dan SOLVE::

4 12 3 9 1 21 5 2

4 12 3 9 1 21 5 2

4 12 3 9 1 21 5 2

4 12 3 9 1 21 5 2

4 12 3 9 1 21 5 2

4 12 3 9 1 21 5 2

4 12 3 9 1 21 5 2

4 12 3 9 1 21 5 2

4 12 3 9 1 21 5 2

MERGE: 4 12 3 9 1 21 5 2

3 4 12 9 1 21 5 2

3 4 9 12 1 21 5 2

1 3 4 9 12 21 5 2

1 3 4 9 12 21 5 2

1 3 4 5 9 12 21 2

1 2 3 4 5 9 12 21

Kompleksitas waktu algoritma *Insertion Sort*:

$$T(n) = \begin{cases} a & , n = 1 \\ T(n-1) + cn & , n > 1 \end{cases}$$

Penyelesaian:

$$\begin{aligned} T(n) &= cn + T(n-1) \\ &= cn + \{ c \cdot (n-1) + T(n-2) \} \\ &= cn + c(n-1) + \{ c \cdot (n-2) + T(n-3) \} \\ &= cn + c \cdot (n-1) + c \cdot (n-2) + \{ c(n-3) + T(n-4) \} \\ &= \dots \\ &= cn + c \cdot (n-1) + c(n-2) + c(n-3) + \dots + c2 + T(1) \\ &= c \{ n + (n-1) + (n-2) + (n-3) + \dots + 2 \} + a \\ &= c \{ (n-1)(n+2)/2 \} + a \\ &= cn^2/2 + cn/2 + (a-c) \\ &= O(n^2) \end{aligned}$$