Fakultas Informatika
School of Computing
Telkom University
Think Big! Go Forward!

Telkom University

ANALISIS ALGORITMA

Week 02: Kompleksitas Asimtotik

PROGRAM PASCA SARJANA INFORMATIKA
FAKULTAS TEKNIK INFORMATIKA
UNIVERSITAS TELKOM

2022/2023

Analisis Asimtotik Kompleksitas

bagian 4 dari 4

Kompleksitas Asimtotik

Asimptotik adalah

[**Merriam Webster**] A straight line associated with curve such that as a point moves along an infinite branch of the curve, the distance from the point to the line approaches zero and the slope of the curve at the point approaches the slope of the line.

[**Britannica**] In mathematics, a line or curve that acts as the limit of another line or curve. For example, a descending curve that approaches but does not reach the horizontal axis is said to be asymptotic to that axis, which is the asymptote of the curve.

[**Mathworld Wolfram**] An asymptote is **a line or curve that approaches a given curve arbitrarily closely.**

Asimptotik Batas Atas

$\mathbf{O}$(f(n)) adalah fungsi *f(n)* setelah dikali suatu konstanta positif **c** akan selalu berada diatas data pengamatan *g(n)* setelah sejumlah $\mathbf{n_0}$ data.

$$g(n) \leq c.f(n) \text{ saat } \mathbf{n \geq n_0}$$

Kita mencari kurva fungsi f(n) yang dapat memprediksi **seakurat mungkin kinerja (g(n))** algoritma tersebut dengan berbagai variasi data masukan, untuk jumlah data yang relatif besar.

Kompleksitas algoritma umumnya diperoleh dalam bentuk big-Oh ini. Contohnya selection sort $\mathbf{O}(n^2)$.

Asimptotik Batas Bawah

Ω(f(n)) adalah fungsi *f(n)* setelah dikali suatu konstanta positif **c** akan selalu berada dibawah data pengamatan setelah minimum $\mathbf{n_0}$ data.

$$g(n) \geq c.f(n) \quad \text{saat } \mathbf{n \geq n_0}$$

Kita mencari kurva fungsi f(n) yang dapat memprediksi **kebutuhan mininum (g(n))** algoritma/problem tersebut dengan berbagai variasi data masukan, untuk jumlah data yang relatif besar.

Bagi selection sort $\Omega(n^2)$ juga ternyata merupakan batas bawah, lihat kembali ilustrasi data.

Asimptotik Batas Optimal

Θ(f(n)) adalah fungsi *f(n)*

setelah dikali suatu konstanta positif $\mathbf{c_1}$ akan selalu berada diatas data pengamatan (*g(n)*) setelah minimum $\mathbf{n_0}$ data

dan

setelah dikali suatu konstanta positif $\mathbf{c_2}$ akan selalu berada dibawah data pengamatan (*g(n)*) setelah minimum $\mathbf{n_0}$ data

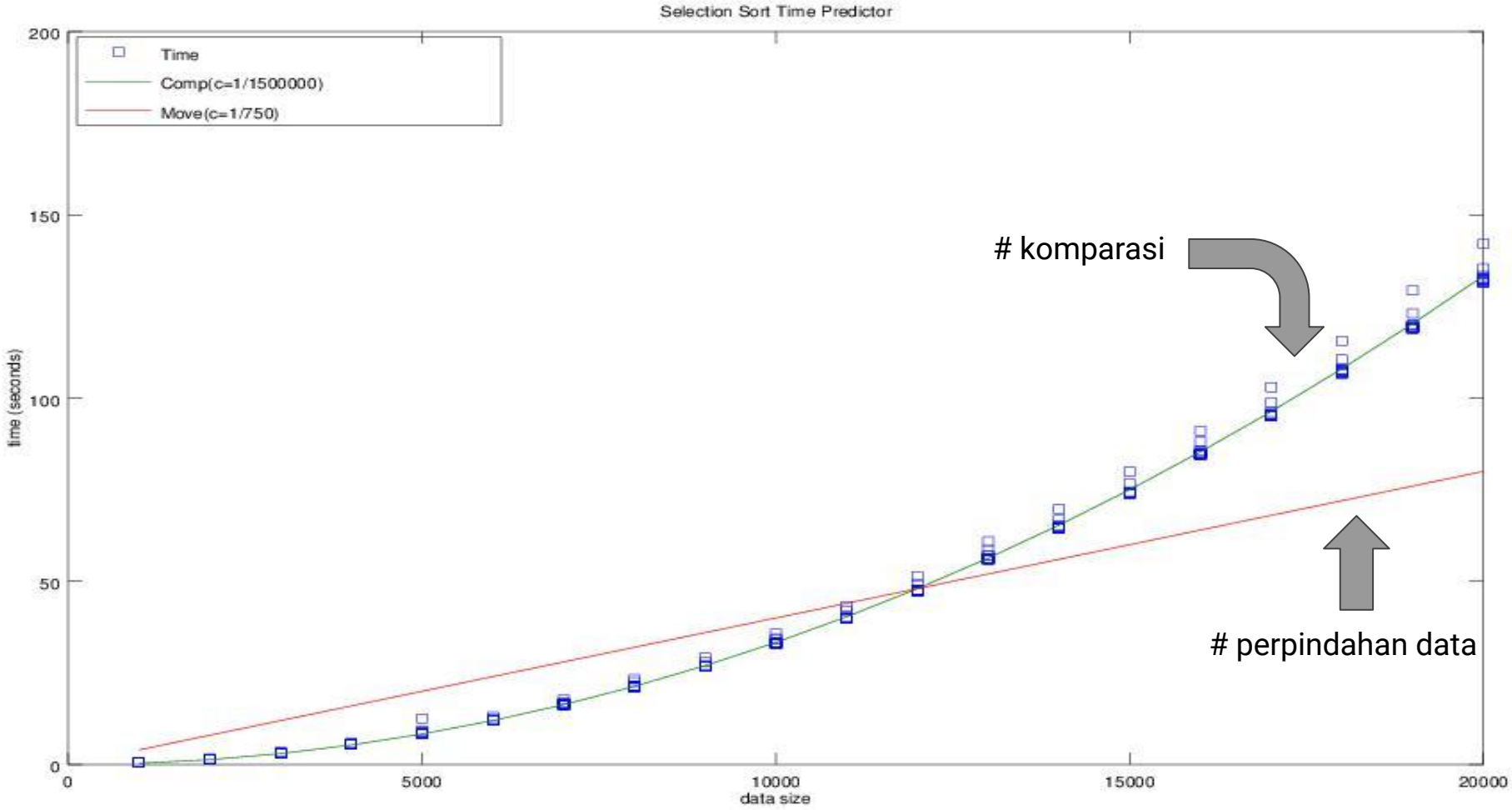
Θ(f(n)) jika Ω(f(n)) dan sekaligus juga O(f(n))

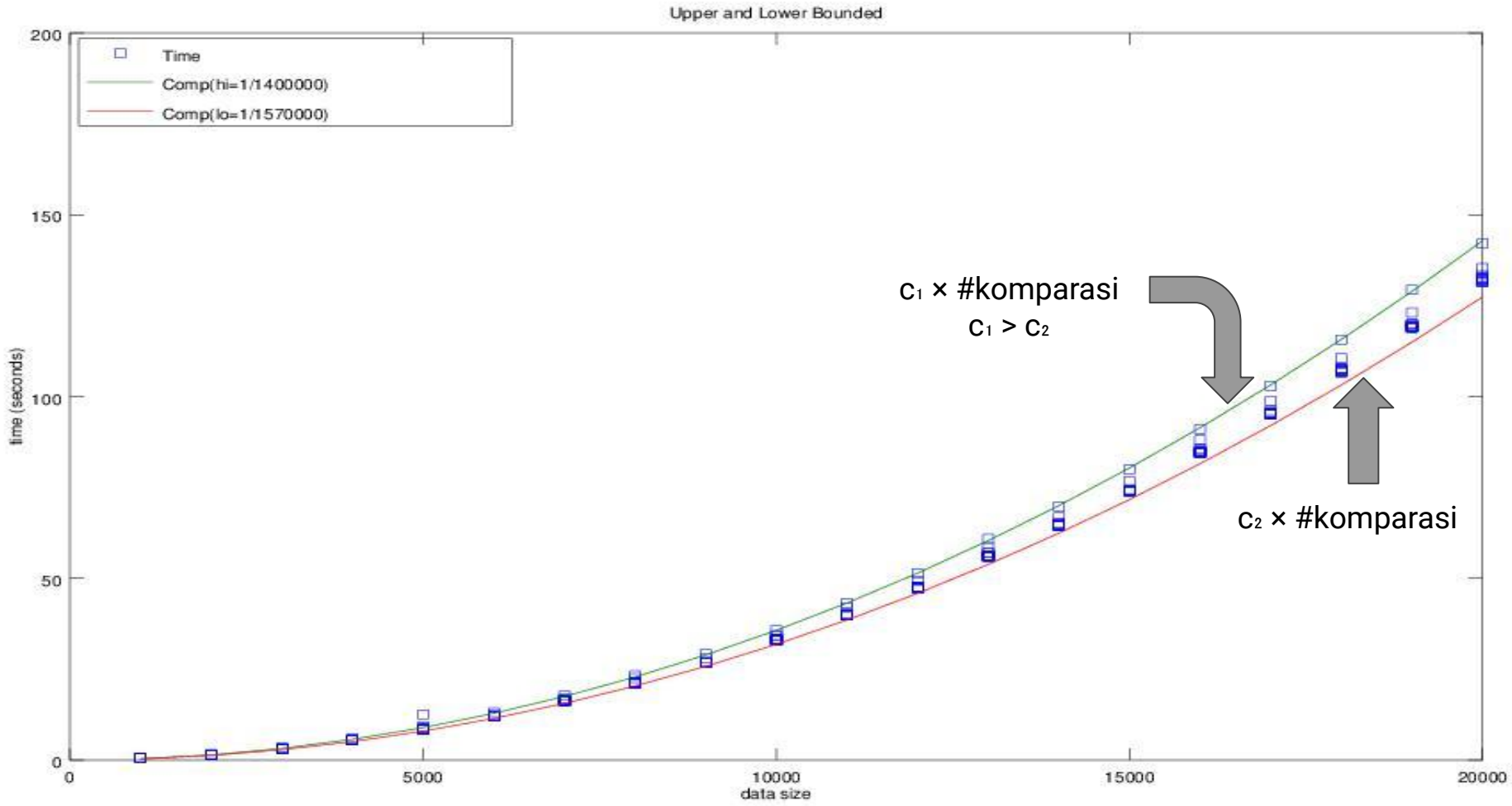
Asimptotik Batas Optimal

Θ(f(n)) adalah fungsi *f(n)*

$$d.f(n) \leq g(n) \leq c.f(n) \text{ saat } \mathbf{n \geq n_0}$$

- Jika batas atas dan batas bawah kompleksitas algoritma sama. Contoh: Selection sort pada $\Theta(n^2)$
- Jika batas atas kompleksitas algoritma sama dengan kebutuhan minimum komputasi problem tersebut, maka disebut problem tersebut disebut sudah mempunyai solusi yang optimal

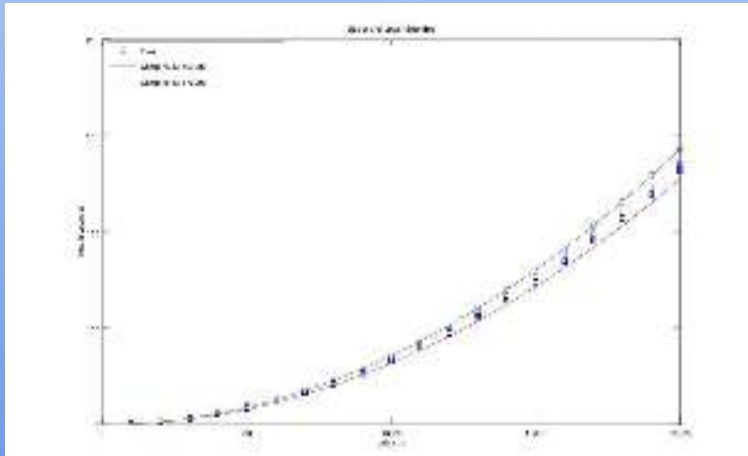

Selection Sort Time Predictor
Time
Comp(c=1/1500000)
Move(c=1/750)
time (seconds)
data size
0
50
100
150
200
0
5000
10000
15000
20000
komparasi
perpindahan data


Upper and Lower Bounded
Time
Comp(hi=1/1400000)
Comp(lo=1/1570000)
time (seconds)
data size
0
50
100
150
200
0
5000
10000
15000
20000
c₁ × #komparasi
c₁ > c₂
c₂ × #komparasi

Kompleksitas Asimptotik Selection Sort

Hasil eksperimen:

- Waktu eksekusi T(k)=c.k dengan konstanta c dan jumlah komparasi k
- Batas atas waktu terdekat dicapai pada c_{atas}.k dimana c_{atas}=1/1,400,000
- Batas bawah waktu terdekat dicapai pada c_{bawah}.k, dimana c_{bawah}=1/1,570,000
- Dimana c_{atas} dan c_{bawah} adalah konstanta hasil pengamatan
- Konstanta bergantung pada bahasa dan komputer yang digunakan
- Cari fungsi dengan prediktor sebagai parameter independen, setelah dikali kostanta, **selalu** diatas (dibawah) waktu setelah jumlah data tertentu.



Utak Atik Hitungan FindMax

```
function FindMax( A[1..n] ) : index
    imax := 1
    i := 2
    while i ≤ n do
        if A[i] > A[imax] then
            imax := i
        endif
        i := i + 1
    endwhile
    return imax
```

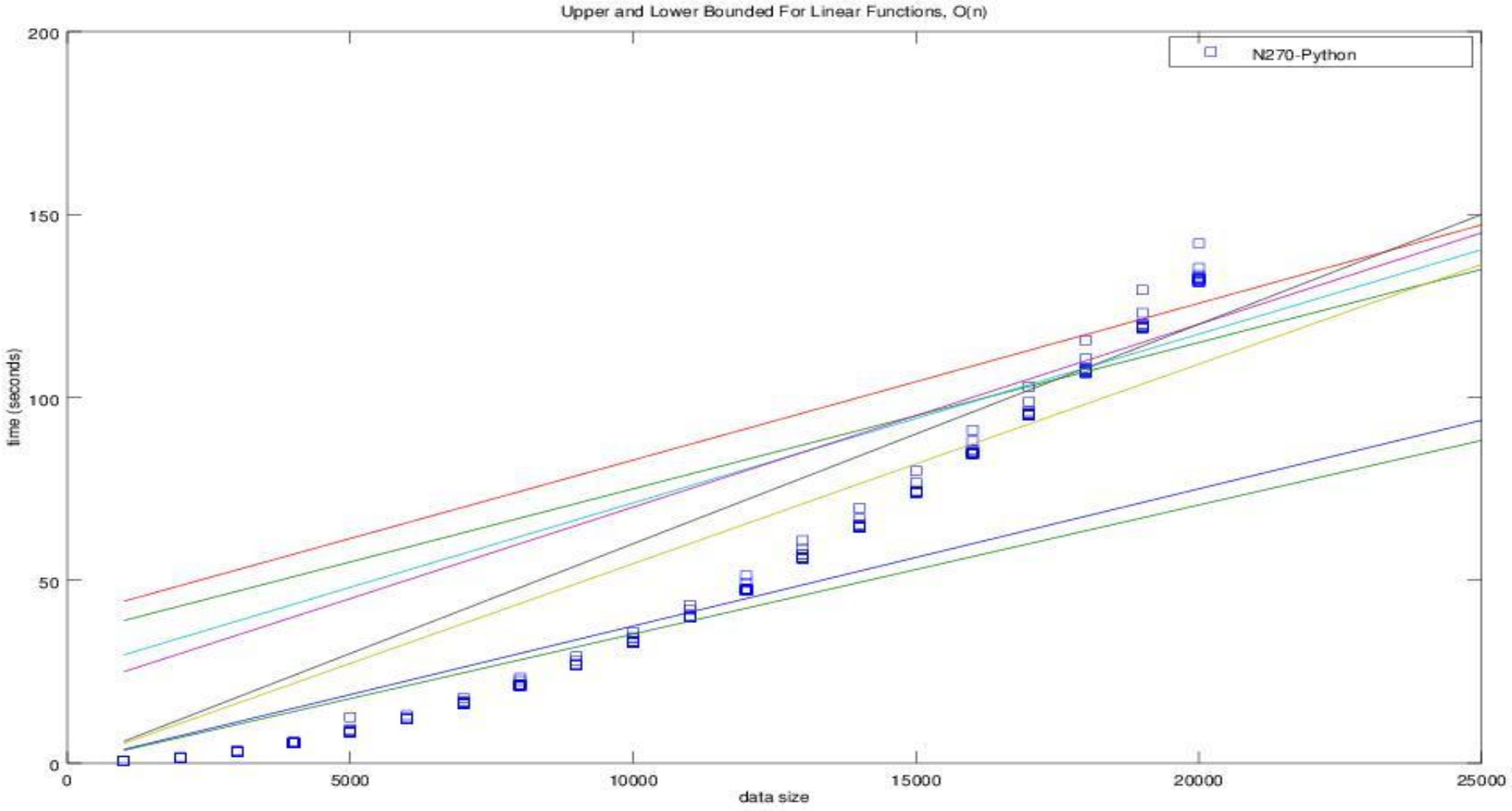
- Komparasi terjadi pada instruksi “**if**”
- Terjadi untuk i=2..n
- Sehingga jumlah komparasi untuk n data, $T_{FINDMAX}(n)=n-1$

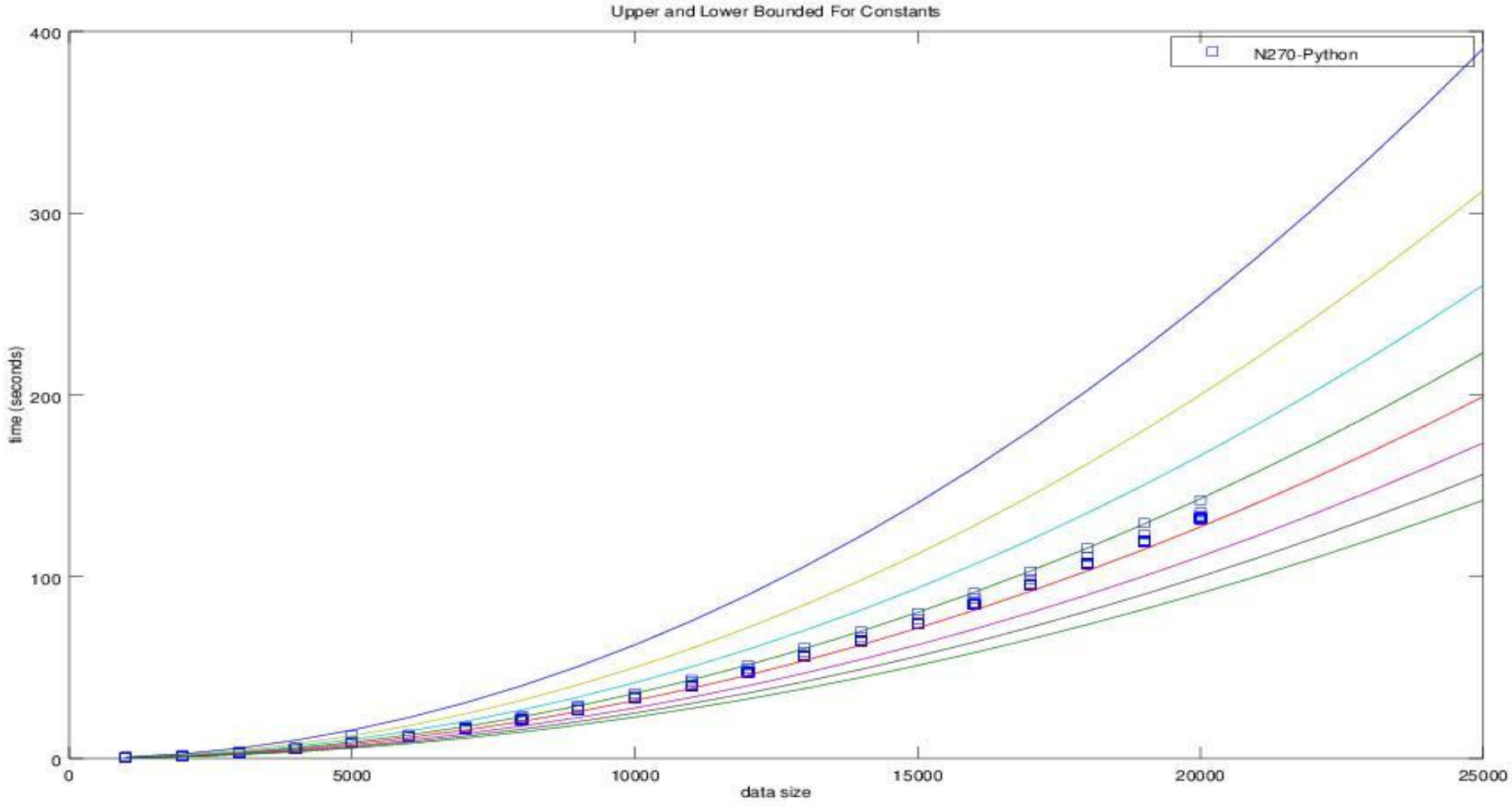
Utak Atik Hitungan Selection Sort

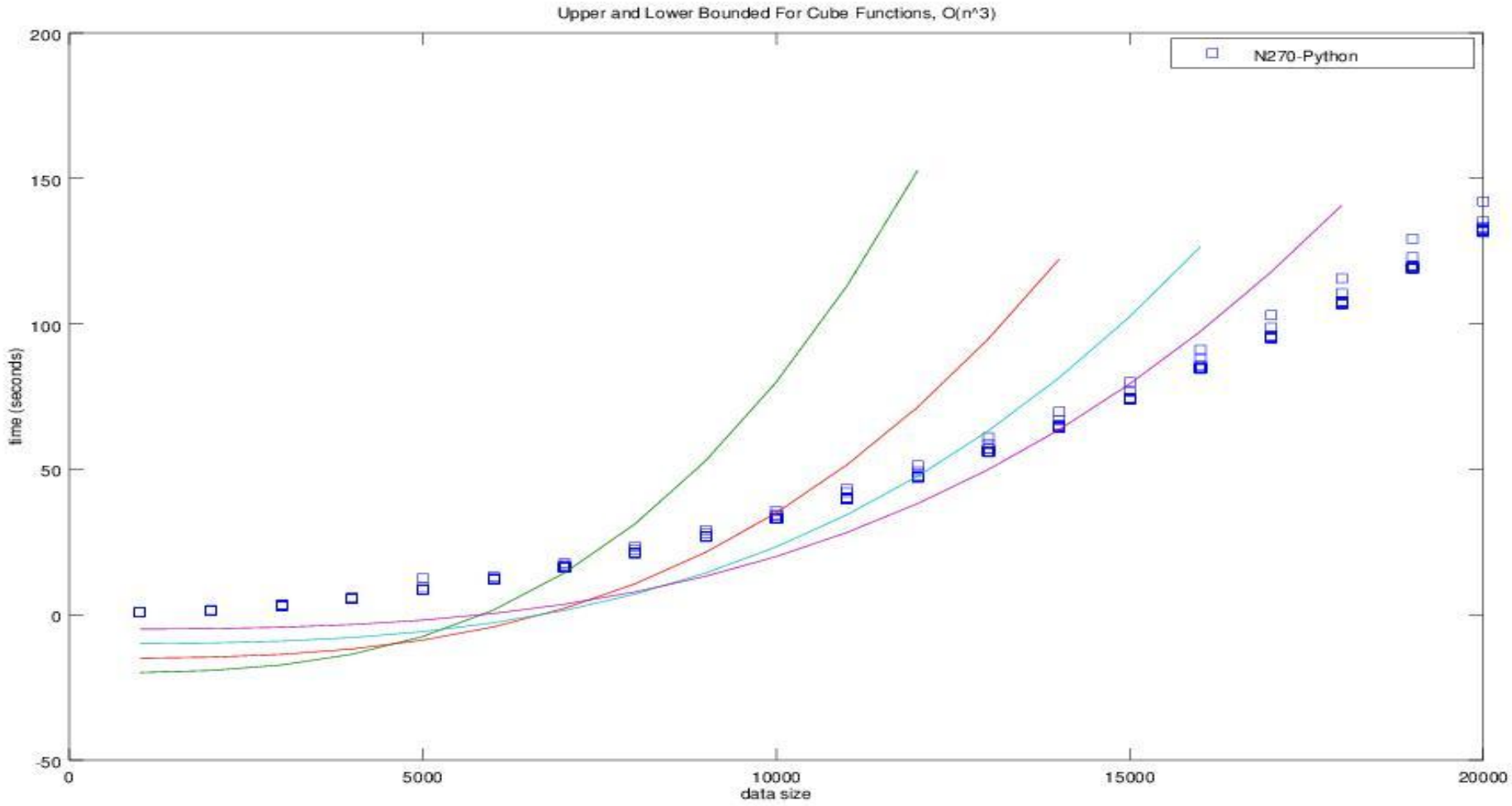
```
proc SelectionSort( A[1..n] )
    i := n
    while i > 1 do
        j := FindMax(A[1..i])
        t := A[j]
        A[j] := A[i]
        A[i] : = t
        i := i - 1
    endwhile
```

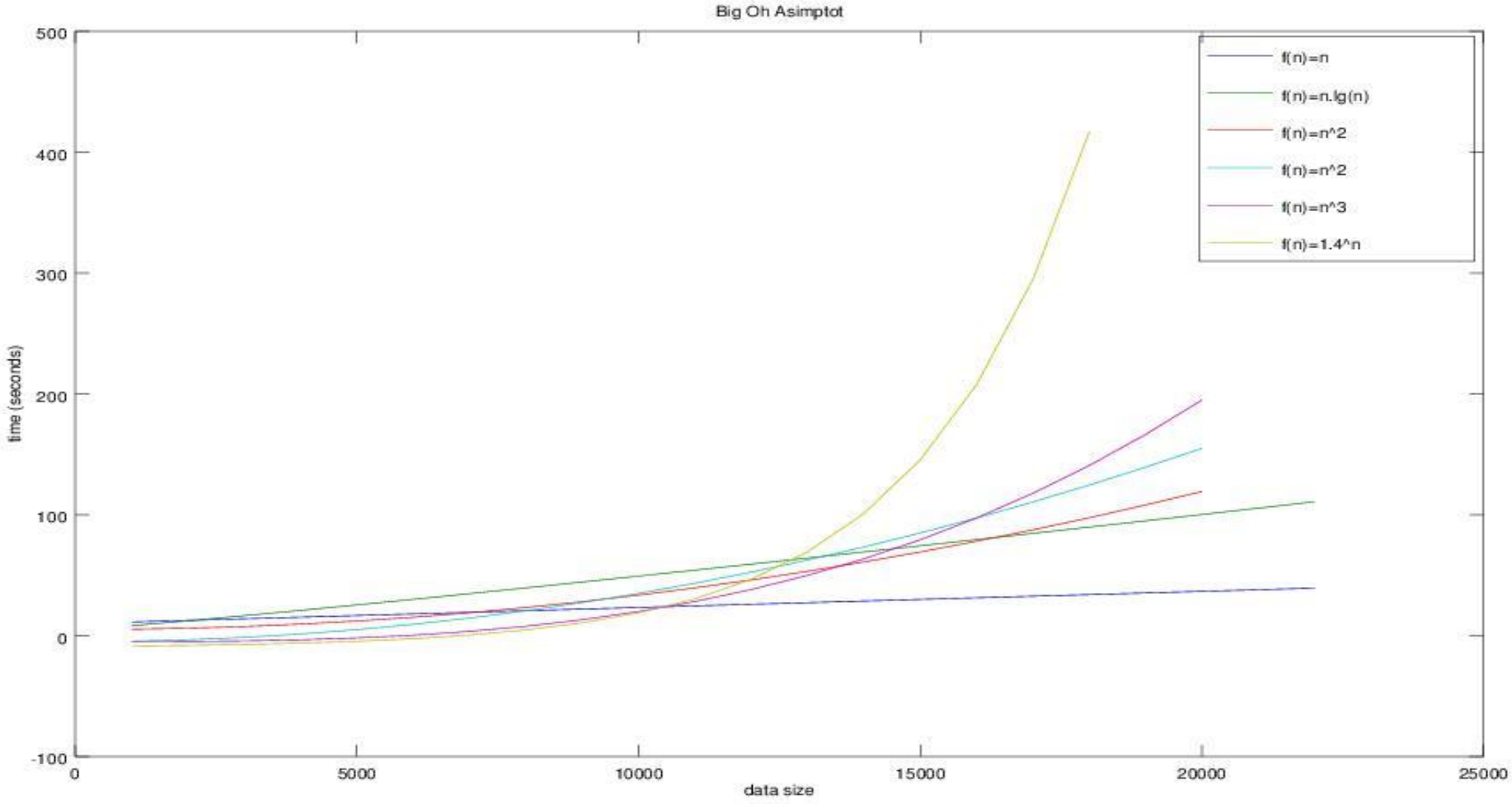
- Komparasi terjadi dalam fungsi **FindMax**
- Sehingga jumlah komparasi,

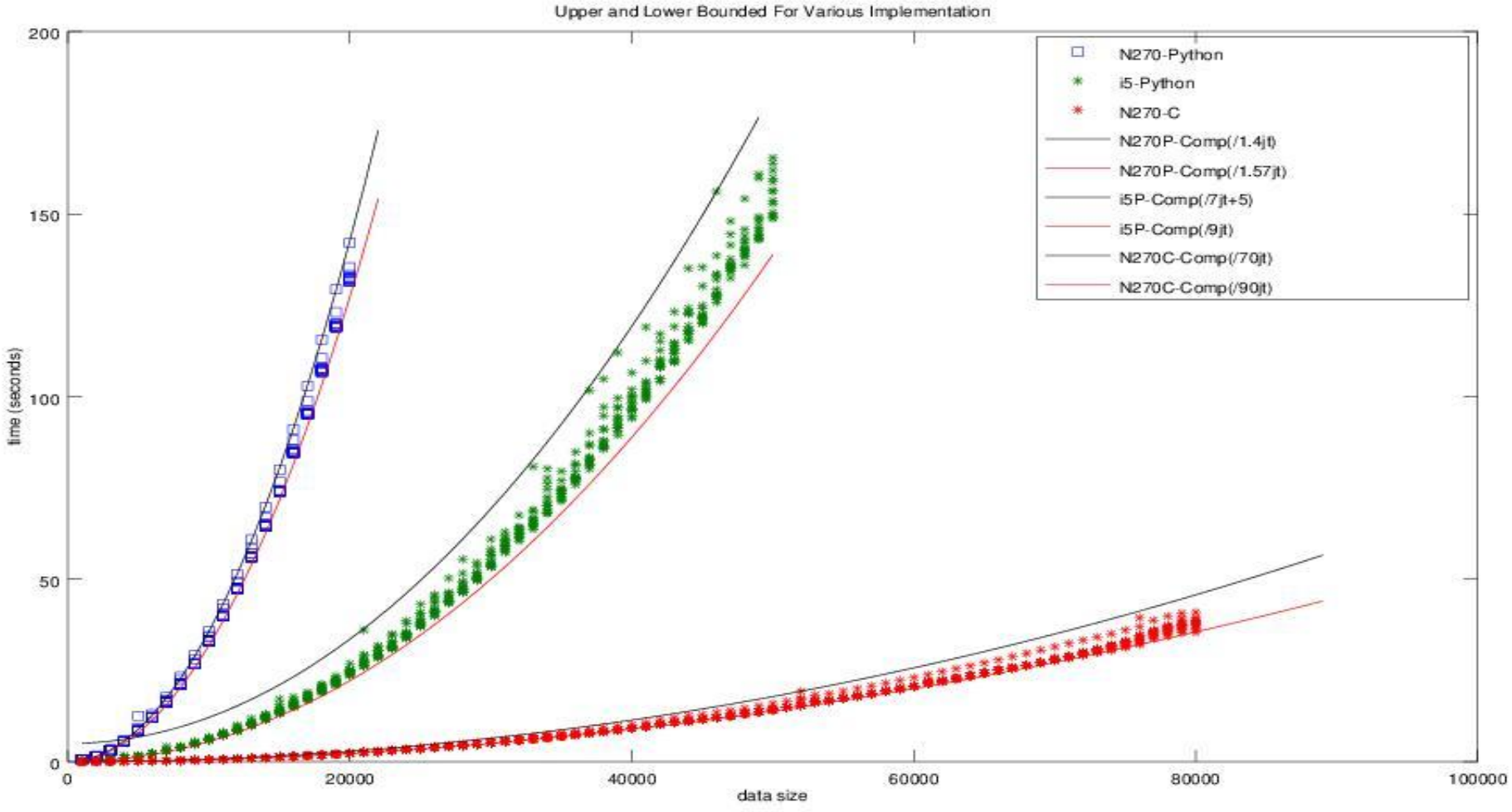
$$
\begin{aligned}
T_{SSORT}(n) &= \Sigma_{i=2..n} T_{FINDMAX}(i) \\
&= \Sigma_{i=2..n}(i-1) \\
&= \Sigma_{i=1..n-1}(i) \\
&= \tfrac{1}{2}(n-1+1)(n-1) \\
&= \tfrac{1}{2}(n)(n-1) \\
&= \tfrac{1}{2}n^2 - \tfrac{1}{2}n
\end{aligned}
$$

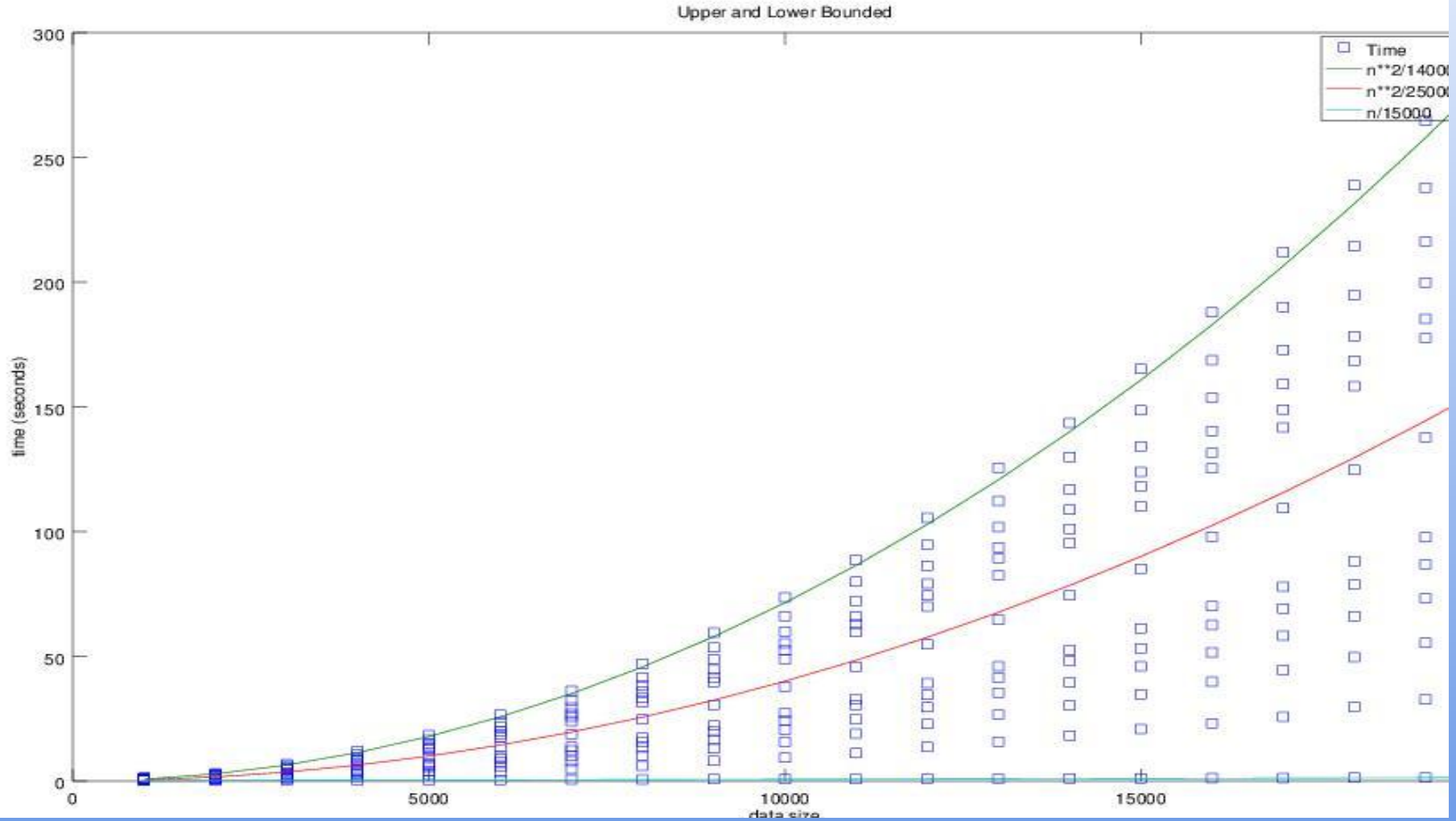

Upper and Lower Bounded For Linear Functions, O(n)
N270-Python
time (seconds)
data size
0
50
100
150
200
0
5000
10000
15000
20000
25000


Upper and Lower Bounded For Constants
N270-Python
time (seconds)
data size
0
100
200
300
400
0
5000
10000
15000
20000
25000


Upper and Lower Bounded For Cube Functions, O(n^3)
N270-Python
time (seconds)
data size
200
150
100
50
0
-50
0
5000
10000
15000
20000


Big Oh Asimptot
f(n)=n
f(n)=n.lg(n)
f(n)=n^2
f(n)=n^2
f(n)=n^3
f(n)=1.4^n
time (seconds)
data size
500
400
300
200
100
0
-100
0
5000
10000
15000
20000
25000

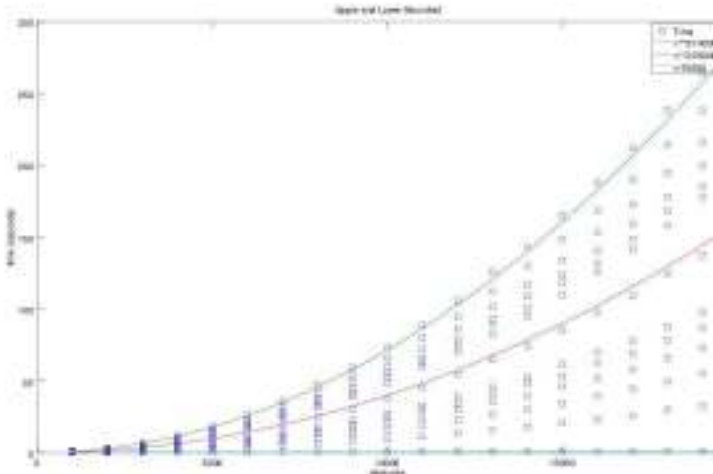

Upper and Lower Bounded For Various Implementation
N270-Python
i5-Python
N270-C
N270P-Comp(/1.4jt)
N270P-Comp(/1.57jt)
i5P-Comp(/7jt+5)
i5P-Comp(/9jt)
N270C-Comp(/70jt)
N270C-Comp(/90jt)
time (seconds)
data size
0
50
100
150
200
0
20000
40000
60000
80000
100000


Upper and Lower Bounded
Time
n**2/1400
n**2/2500
n/15000
time (seconds)
data size
0
50
100
150
200
250
300
0
5000
10000
15000

Asymptotic Complexity for Insertion Sort

Hasil eksperimen:

- Komparasi dan pemindahan data mempunyai kurva yang mirip dengan waktu eksekusi, karena …
- Disparitas batas bawah dan batas atas yang besar, karena …
- Konstanta pengali jumlah komparasi dan pemindahan data bergantung pada bahasa dan hardware yang digunakan
- Cari fungsi dengan prediktor sebagai parameter independen, setelah dikali kostanta, **selalu** diatas (dibawah) waktu setelah jumlah data tertentu.



Utak Atik Hitungan Insertion Sort

```
proc InsertionSort( A[1..n] )
    i := 2
    while i ≤ n do
        t := A[i]
        j := i-1
        while j ≥ 1 and A[j] > t do
            A[j+1] := A[j]
            j := j - 1
        endwhile
        A[j+1] := t
        i := i + 1
    endwhile
```

- Jumlah operasi terdalam (A[j+1] := A[j]) berada dalam 2 nested loops:
  - loop luar: i:=2 … n, atau n-1 iterasi
  - loop dalam j:=i-1 … 1 atau s.d. A[j] ≤ t ≤ A[j+2]
- Worst case, loop dalam berhenti saat j == 0
  - loop dalam i-1 iterasi, sehingga $\mathbf{T_{ISORT}(n)}=\Sigma_{i=2..n}(i-1) = \Sigma_{i=1..n-1}(i)=½(n-1+1)(n-1)=½(n)(n-1) = \mathbf{½n^2-½n}$
- “Best” case, tidak masuk loop karena A[j==i-1] > t
  - hanya loop luar, sehingga $\mathbf{T_{ISORT}(n)}$=n-1
- Average case adalah rata[2] dari semua kemungkinan loop-dalam berhenti.
  - Asumsi probabilitas terjadinya loop-dalam antara 0 kali s.d. i-1 kali adalah sama
  - Sehingga rerata jumlah iterasi dalam adalah $T_j(i) = (\Sigma_{j=0..i-1}(j))/i = (\Sigma_{j=1..i}(j))/i - 1 = ½(i+1)-1 = ½i -½$
  - $\mathbf{T_{ISORT}(n)}=\Sigma_{i=2..n}(½i -½) = ½\Sigma_{i=2..n}(i-1) = ½(½n^2-½n) = \mathbf{¼n^2-¼n}$

Akhir dari Topik Minggu 02