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

Telkom University

ANALISIS ALGORITMA

Week 02: Kasus Algoritma Sorting Sederhana

PROGRAM PASCA SARJANA INFORMATIKA
FAKULTAS TEKNIK INFORMATIKA
UNIVERSITAS TELKOM

2022/2023

Problem Sorting

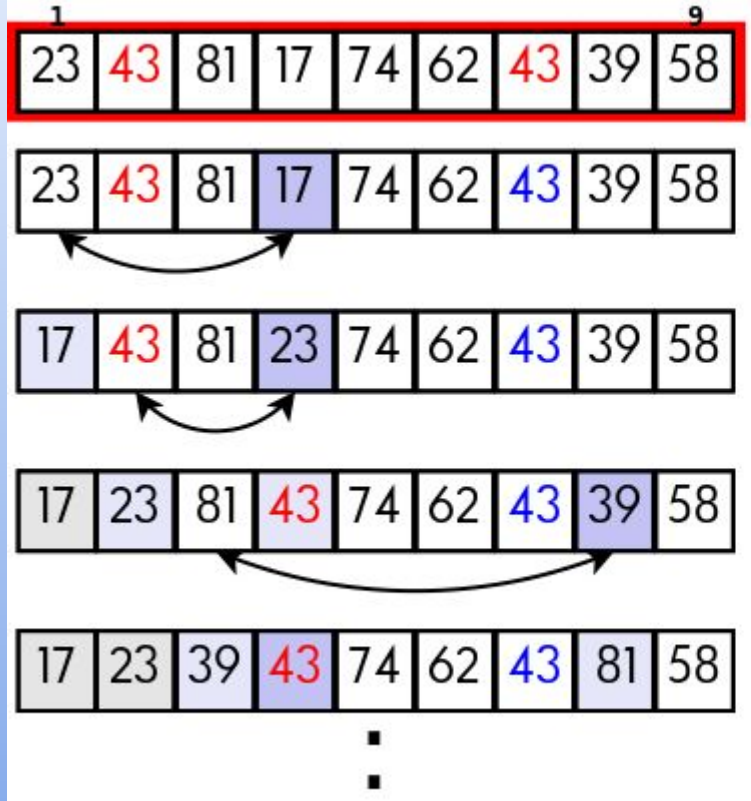
atau Permutasi Data
bagian 1 dari 4

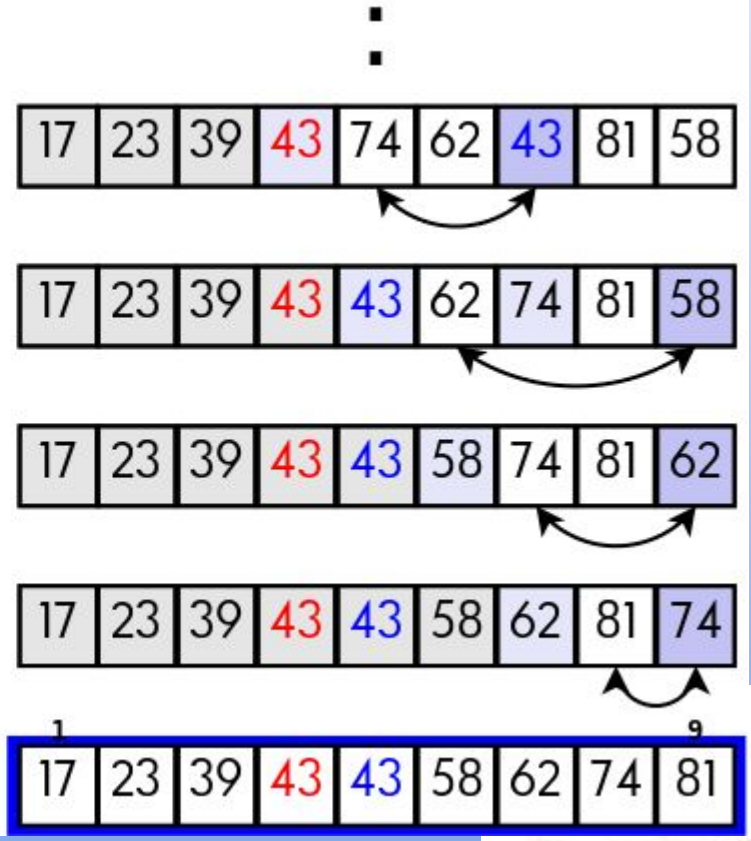
Algoritma Selection Sort
(alias Maximum Sort)

Selection Sort: Ide Solusi

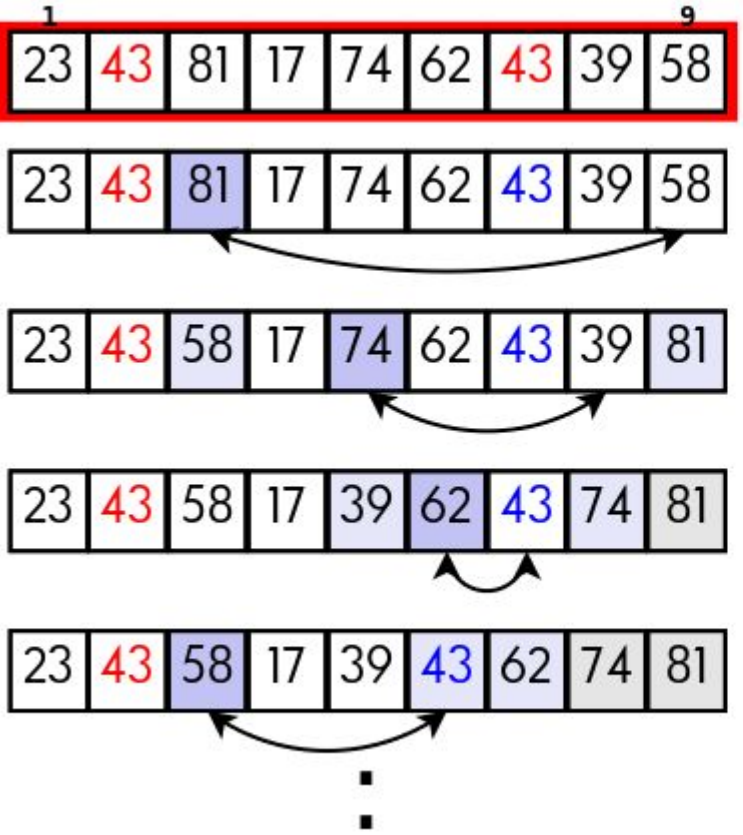
E. Friend (1956) dan D. Bell (1958)

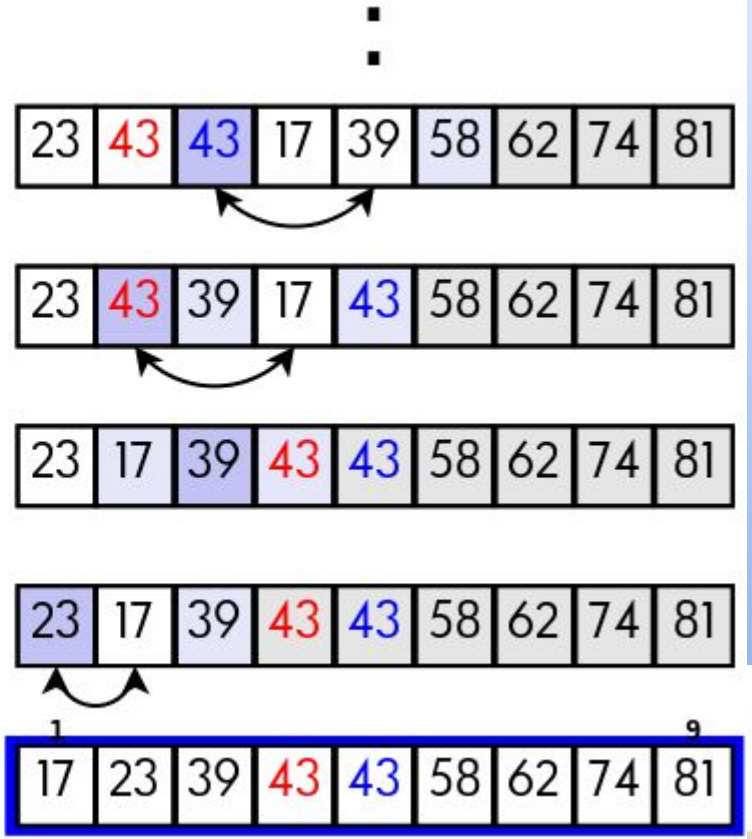
- **Properti** array terurut:
  Array yang terurut mempunyai data terbesar pada indeks terbesar,
- Dan: Indeks ke-2 terbesar berisi data ke-2 terbesar, dst.
- Jadi **ide sorting**nya adalah:
  Cari data terbesar dan tempatkan ditempat seharusnya
- Lakukan iterasi untuk terbesar berikutnya





Proses dengan mencari yang **terkecil** lebih dulu





Proses dengan mencari yang **TERBESAR** lebih dulu

Selection Sort: Algoritma

```
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
```

Idenya sederhana:

- Array yang terurut mempunyai data terbesar pada indeks terbesar,
- Indeks ke-2 terbesar berisi data ke-2 terbesar, dst.
- Jadi, cari data terbesar dan tempatkan ditempat seharusnya
- Lakukan iterasi untuk terbesar berikutnya
- Satu data tidak perlu diurutkan

Selection Sort: Pembuktian

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

- **Kondisi terminasi**
  - ...
- **Loop Invarian**
  - Refleksi dari perjalanan proses yang sedang berlangsung
- Setelah selesai iterasi sebelumnya
  - Semua data A[1..i] lebih kecil dari A[i+1]
  - Semua data A[i+1.. n] sudah terurut
- Dan kemudian selalu periksa batas persisnya tersebut

Selection Sort: Pembuktian

```
proc SelectionSort( A[1..n] )
     i := n
     {L.I.: A[1..i]≤A[i+1]≤ .. ≤A[n] }
     while i > 1 do
          j := FindMax(A[1..i])
          t := A[j]
          A[j] := A[i]
          A[i] : = t
          i := i - 1
     endwhile
     {Term: i ≤ 1 }
```

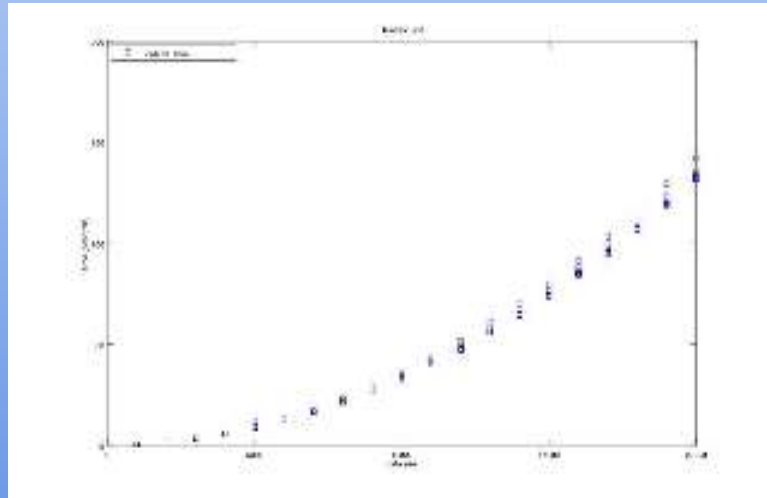
- **Loop Invarian**
  - Semua data A[1..i] lebih kecil dari A[i+1]
  - Semua data A[i+1.. n] sudah terurut
- **Basis**: i==n saat pertama kali memasuki loop
  - Sehingga A[n+1 .. n] berada diluar rentang array
  - (Ingat: op implikasi logis p ⇒ q pada tabel kebenaran)
- **Hipotesis**: Kondisi LI diatas berlaku untuk 2 < i ≤ n
- **Induksi**:
  - Saat j memperoleh maks., kondisi tidak berubah
    - A[1..**i**] ≤ A[j] ≤ A[i+1] ≤ .. ≤ A[n], dimana 1 ≤ j ≤ i
  - 3 instruksi berikutnya terjadi pertukaran data, sehingga:
    - A[1..**i-1**] ≤ A[i] ≤ A[i+1] ≤ .. ≤ A[n]
  - Kondisi L.I. kembali diperoleh setelah i := i - 1
    - A[1..**i**] ≤ A[i+1] ≤ .. ≤ A[n]
- **Terminasi**:
  - i ≤ 1 (karena dekremental, i==1)
  - A[1..i] ≤ A[i+1] ≤ .. ≤ A[n]
  - Sehingga **A[1] ≤ A[2] ≤ .. ≤ A[n] atau terurut** !

- Karakteristik eksekusi?
- Perkiraan untuk jumlah data lebih besar?
- Pengaruh dari platform yang digunakan?


Selection sort
Grafik waktu eksekusi suatu implementasi algoritma selection sort
terhadap jumlah data yang diurutkan
time (seconds)
data size
0
50
100
150
200
0
5000
10000
15000
20000

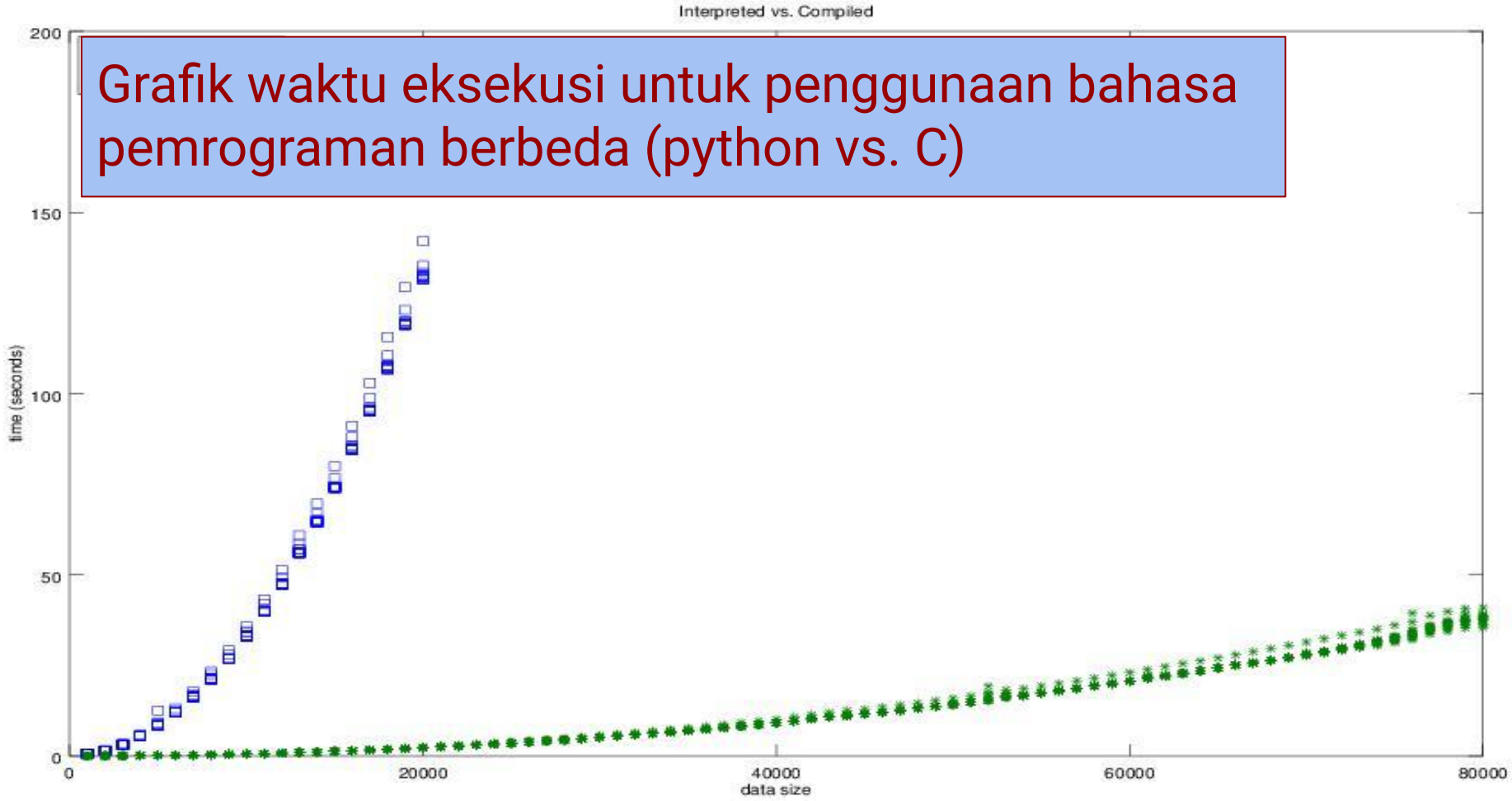
Karakteristik Eksekusi Selection Sort

- Apa yang terjadi untuk data lebih banyak?
- Apakah dapat diprediksi secara matematis?
- Apakah yang memprediksi paling baik? (Apa parameter independen yang cocok?)
- Apa yang terjadi jika komputer lebih cepat?
- Ap ayang terjadi jika bahasa pemrograman diganti?




Slower vs. Faster Computers
Grafik waktu eksekusi pada dua komputer dengan kecepatan berbeda
time (seconds)
data size
0
50
100
150
200
0
10000
20000
30000
40000
50000

Grafik waktu eksekusi untuk penggunaan bahasa pemrograman berbeda (python vs. C)



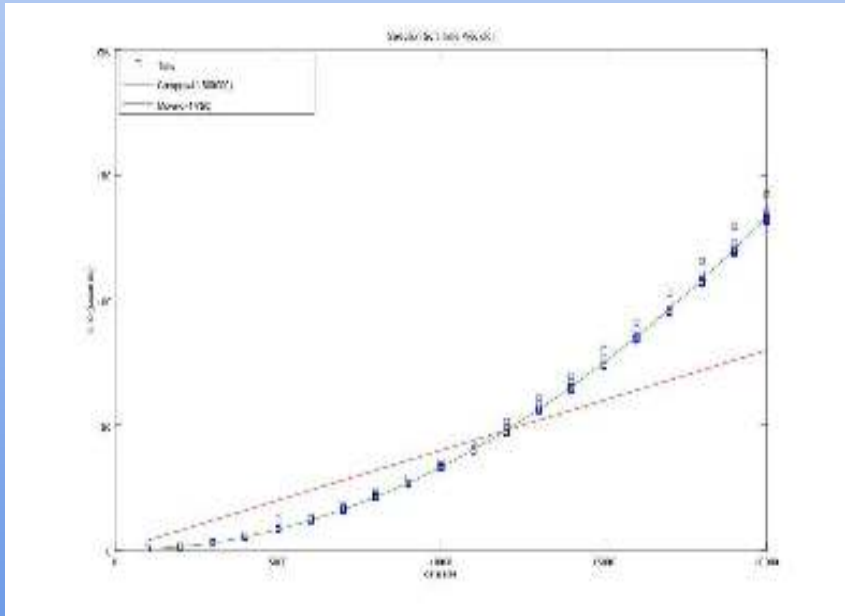
Prediksi Sumber Daya (waktu) Eksekusi

- Jika diketahui kebutuhan sumber daya untuk data sebanyak n_0,
  Berapa kebutuhan sumber daya untuk data sebanyak n, dimana $n > n_0$?

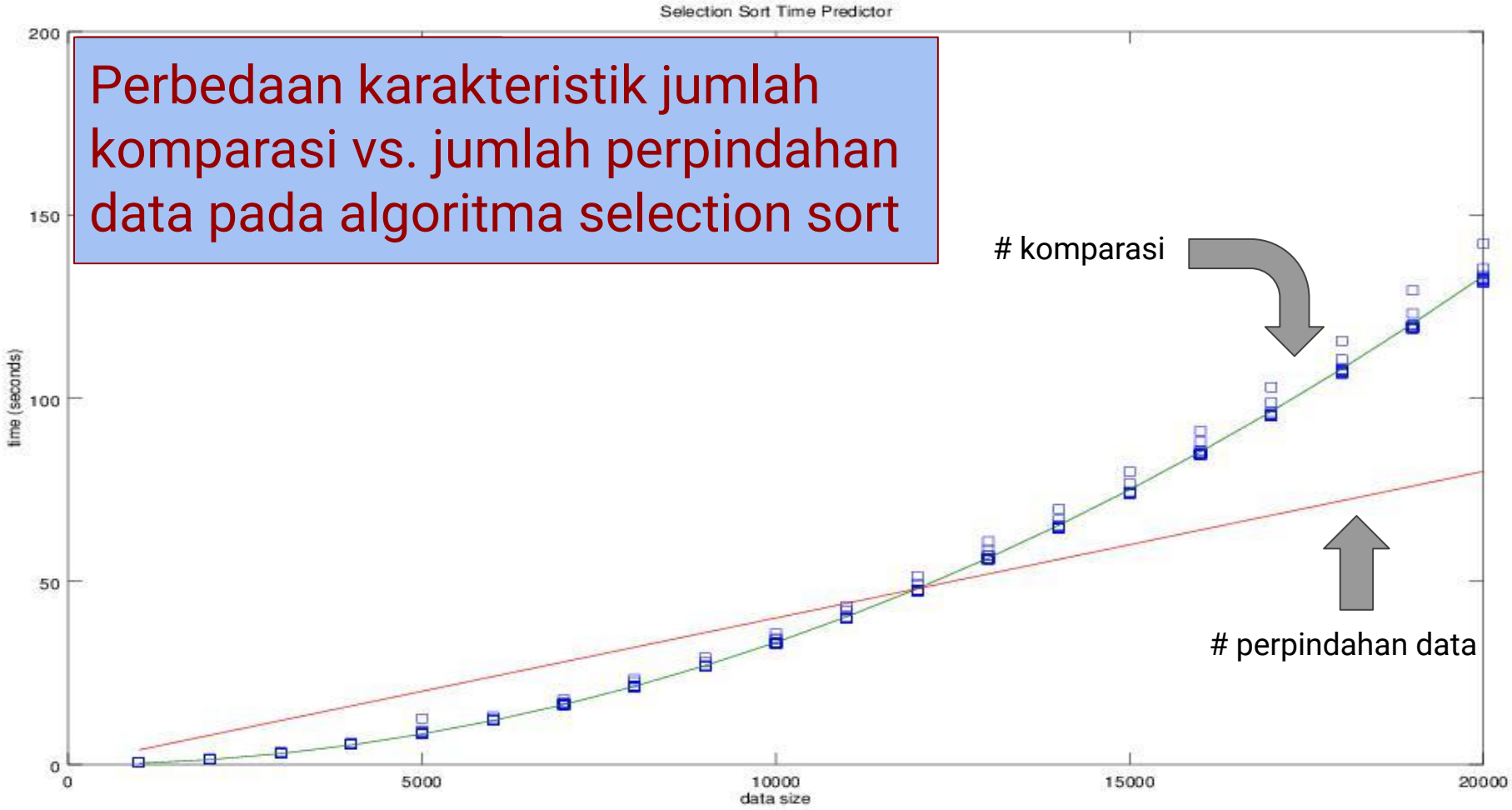
- Apa alat prediksi yang paling cocok?
- Apa yang diprediksi:
  - Batas atas: Kebutuhan sumber daya terbesar,
  - Rerata: Prediksi rerata kebutuhan sumber daya,
  - Batas bawah: Paling sedikit sumber daya yang harus disiapkan?

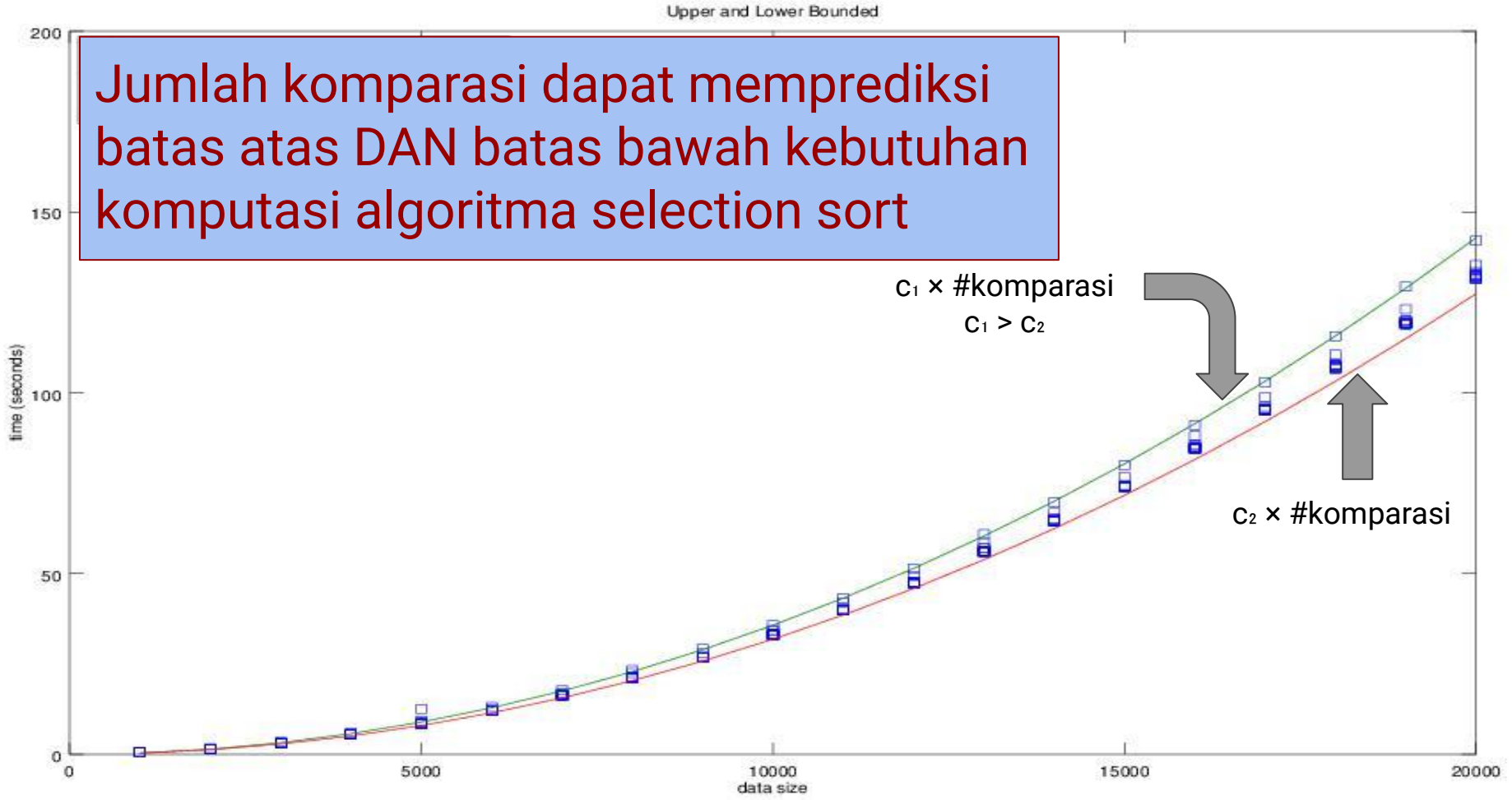
Memprediksi Karakteristik Selection Sort

- Beberapa operasi utama sebagai representatif algoritma keseluruhan
- Operasi yang sering diperhatikan adalah:
  - Perpindahan/pertukaran data
  - Pembandingan data



Perbedaan karakteristik jumlah komparasi vs. jumlah perpindahan data pada algoritma selection sort




Upper and Lower Bounded
Jumlah komparasi dapat memprediksi batas atas DAN batas bawah kebutuhan komputasi algoritma selection sort
c1 × #komparasi
c1 > c2
c2 × #komparasi
time (seconds)
data size
0
50
100
150
200
0
5000
10000
15000
20000

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,

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

Akhir Bagian 1 dari Topik Minggu 02