CODELABS
BUILD SOCIETY WITH TECHNOLOGY

ANALISIS ALGORITMA

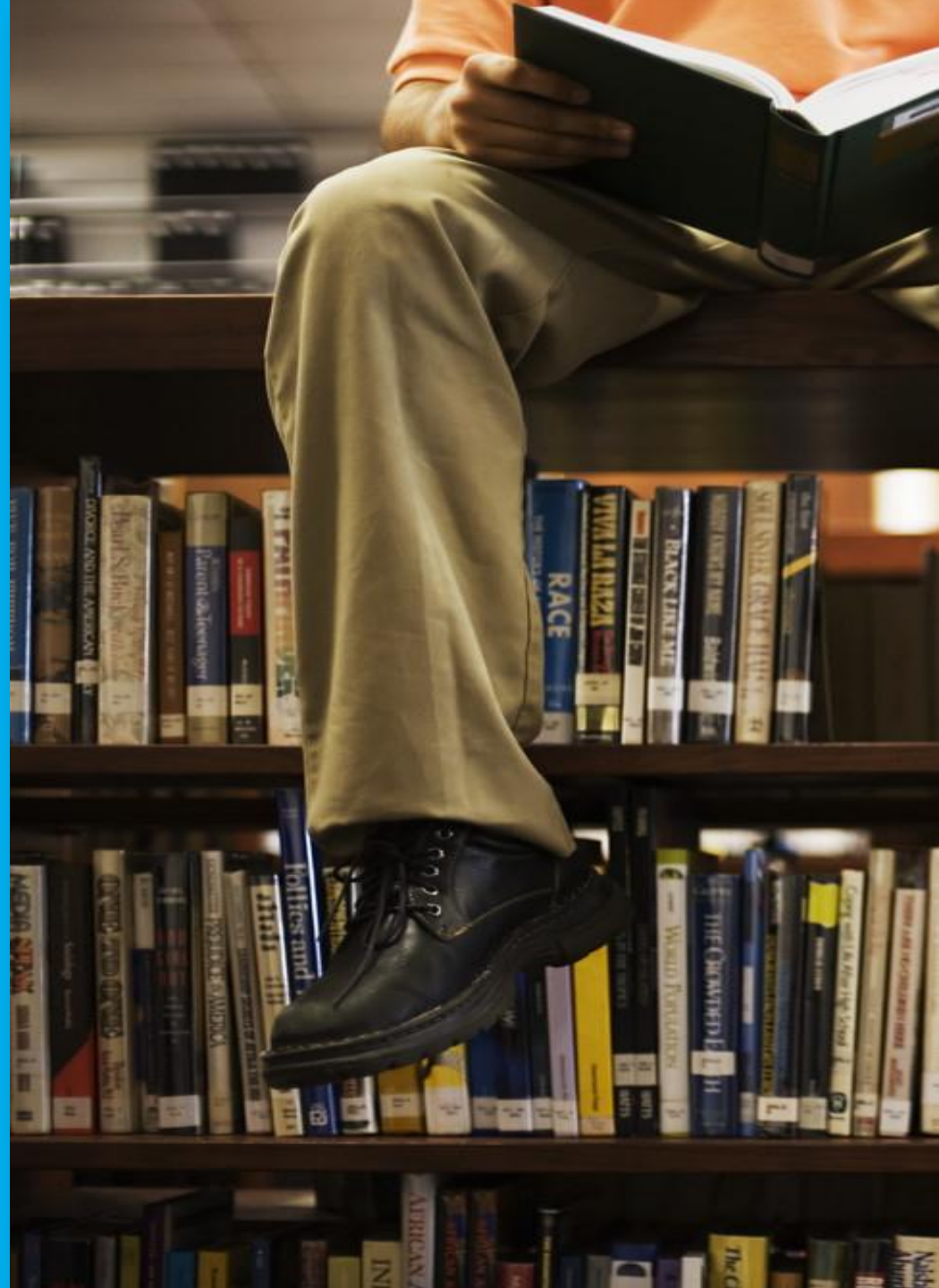
Analisis Masalah dan Running Time

Disusun Oleh:

Adam Mukharil Bachtiar

Teknik Informatika UNIKOM

adfbipotter@gmail.com



AGENDA PERKULIAHAN

- Penjelasan masalah
- Alat ukur algoritma
- Pengukuran Algoritma
- Jenis Kompleksitas Running Time

Penjelasan Masalah

DEFINISI MASALAH

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L} \right), x = 1$$

"Pertanyaan atau tugas yang harus dicari solusinya"

CONTOH MASALAH DAN SOLUSI (1)

Masalah:

[Masalah Pengurutan] Diberikan array (senarai) yang terdiri dari n buah bilangan bulat.

Bagaimana mengurutkan n buah bilangan tersebut secara descending?

Solusi:

Barisan bilangan di dalam array yang terurut secara descending.

CONTOH MASALAH DAN SOLUSI (2)

Masalah:

[Masalah Pencarian] Diberikan array (senarai) yang terdiri dari n buah bilangan bulat.

Tentukan apakah bilangan x terdapat pada array tersebut!

Solusi:

Menampilkan “Data ditemukan” jika bilangan x terdapat di array tersebut dan

menampilkan “Data tidak ditemukan” jika bilangan x tidak terdapat di array tersebut.

INSTANSIASI MASALAH

Instansiasi Masalah:

Parameter nilai yang diasosiasikan terhadap masalah. Jawaban dari masalah adalah **solusi**.

Contoh:

Selesaikan masalah pengurutan descending:

Bil={6,2,4,1,3,5} → n=6 //Jumlah input data

Solusi: Bil={6,5,4,3,2,1}

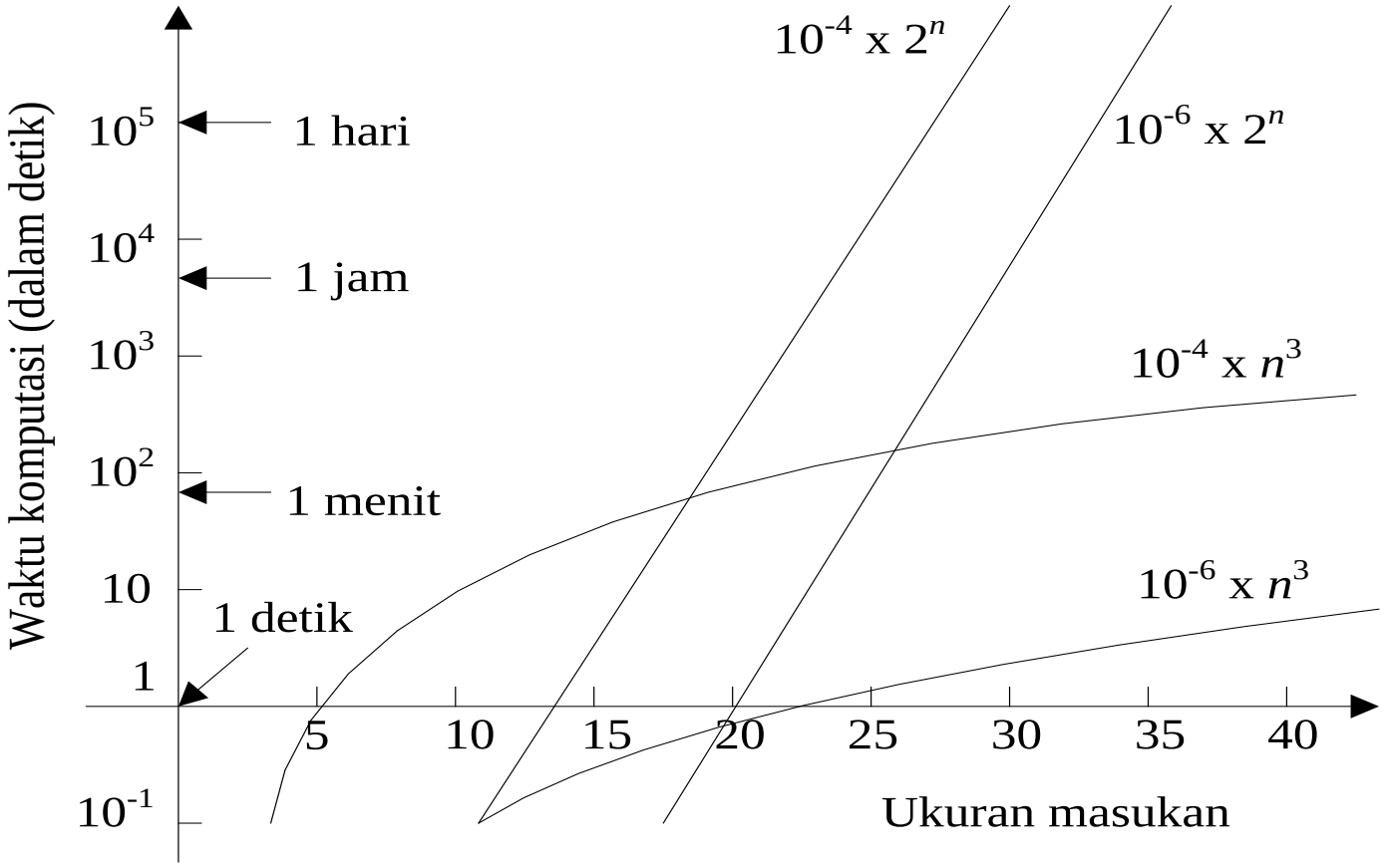
Alat Ukur Algoritma

KENAPA BUTUH ALAT UKUR ALGORITMA

❶ Sebuah algoritma tidak saja harus benar tetapi juga harus mangkus (efisien)

❷ Algoritma yang mangkus adalah algoritma yang meminimumkan penggunaan space (ruang) dan waktu.

HUBUNGAN WAKTU EKSESKUSI TERHADAP INPUT



ALAT UKUR EFISIENSI ALGORITMA

Alat Ukur:

❶ Kompleksitas Waktu → T(n)

❷ Kompleksitas Ruang → S(n)

Keterangan:

n: Ukuran masukan yang diterima oleh algoritma

T(n): Jumlah operasi yang dilakukan untuk menjalankan algoritma sebagai fungsi dari n

S(n): Ruang memori yang dibutuhkan algoritma sebagai fungsi dari n

Pengukuran Algoritma

PENGUKURAN UKURAN INPUT

Definisi:

Jumlah data yang diproses oleh sebuah algoritma. Dalam perhitungan kompleksitas algoritma, ukuran masukan dinyatakan sebagai variabel **n**.

Contoh:

❶ Pengurutan 100 elemen array → n = 100

❷ Pencarian elemen pada array dua dimensi ukuran 4 x 5 → n=20

❸ Penelusuran complete binary tree dengan 10 node secara preorder →n=10

PENGUKURAN RUNNING TIME

❶ Menggunakan satuan waktu standar (second/milisecond).

❷ Tergantung pada:

a. Kecepatan komputer

b. Kualitas program

c. Compiler.

❸ Running time dihitung berdasarkan operasi dasar.

PENJELASAN OPERASI DASAR

Definisi:

Operasi yang paling mendominasi sebagian besar running time suatu algoritma.

Operasi Dasar Bisa Berupa:

❶ Operasi aritmatika (penjumlahan, pengurangan, perkalian, pembagian, dll)

❷ Operasi pengaksesan memori

❸ Operasi input dan output

CONTOH OPERASI DASAR PADA ALGORITMA

❶ Algoritma pencarian elemen dalam array

Operasi khas: Perbandingan nilai antar elemen array.

❷ Algoritma pengurutan elemen dalam array

Operasi khas: Perbandingan nilai antar elemen array dan pertukaran elemen.

❸ Algoritma penjumlahan dua buah array dua dimensi

Operasi khas: Penjumlahan.

❹Algoritma perkalian dua buah array dua dimensi

Operasi khas: Perkalian dan penjumlahan.

CONTOH PERHITUNGAN RUNNING TIME (1)

```
1   Algoritma Deret_Bilangan_Ganjil
2   {I.S: Diinputkan satu nilai akhir oleh user}
3   {F.S: Menampilkan jumlah deret ganjil}
4
5   Kamus:
6      x,akhir:integer
7      jumlah:integer
8
9   Algoritma:
10    input(akhir)
11    jumlah ← 0
12    for x ← 1 to akhir do
13      if x mod 2 = 1 then
14        jumlah ← jumlah + x;
15    endfor
16    output('Jumlah deret ganjil dari 1 - ',akhir,' = ',jumlah)
```

Operasi Dasar:

Perbandingan x mod 2 = 1 (perbandingan sisa bagi)

Kompleksitas Running Time:

T(n) = n, karena operasi dasar diulang dari 1 sampai akhir data

CONTOH PERHITUNGAN RUNNING TIME (2)

```
procedure CariElemenTerbesar(input a1, a2, ..., an : integer, output
maks : integer)
{ Mencari elemen terbesar dari sekumpulan elemen larik integer a1, a2,
..., an.
  Elemen terbesar akan disimpan di dalam maks.
  Masukan: a1, a2, ..., an
  Keluaran: maks (nilai terbesar)
}
Deklarasi
   k : integer

Algoritma
   maks←a1
   k←2
   while k ≤ n do
     if ak > maks then
        maks←ak
     endif
     i←i+1
   endwhile
   { k > n }
```

Operasi Dasar:

Perbandingan a_k > maks (perbandingan elemen yang diperiksa terhadap maks)

Kompleksitas Running Time:

T(n) = n-1, karena operasi dasar diulang dari data ke-2 sampai akhir data

Jenis Kompleksitas Running Time

JENIS KOMPLEKSITAS RUNNING TIME

❶ $T_{max}(n)$

Kompleksitas waktu untuk kasus terburuk (worst case) → kebutuhan waktu maksimum.

❷ $T_{min}(n)$

Kompleksitas waktu untuk kasus terbaik (best case) → kebutuhan waktu minimum.

❸ $T_{avg}(n)$

Kompleksitas waktu untuk kasus rata-rata (average case) → kebutuhan waktu secara rata-rata.

CONTOH PERHITUNGAN (1)

```
1   Procedure SeqSearchBoolean (Input nama_array:tipe_array)
2   {I.S. : elemen array [1..maks_array] sudah terdefinisi}
3   {F.S. : menampilkan data yg dicari ditemukan atau tidak ditemukan}
4   Kamus:
5        i : integer
6        ketemu : boolean
7        data_cari : tipedata
8   Algoritma:
9        input(data_cari)
10       i ← 1
11       ketemu ← false
12       while (not ketemu) and (i ≤ maks_array) do
13           if(nama_var_array(i) = data_cari)
14           then
15                ketemu ←  true
16           else
17                i ← i + 1
18           endif
19       endwhile
20       if (ketemu)
21       then
22           output(data_cari,' ditemukan pada indeks ke-',i)
23       else
24            output(data_cari,' tidak ditemukan')
25       endif
26  EndProcedure
```

Kasus Terbaik:

Apabila $a_1 = x$. $T_{min}(n) = 1$

Kasus Terburuk:

Apabila $a_n = x$ atau x tidak ditemukan. $T_{max}(n) = n$

Kasus Rata-Rata:

Apabila x ditemukan pada posisi ke-j, maka operasi perbandingan akan dieksekusi sebanyak j kali.

$$T_{avg}(n) = \frac{(1+2+3+\ldots+n)}{n} = \frac{\frac{1}{2}n(1+n)}{n} = \frac{(n+1)}{2}$$

CONTOH PERHITUNGAN (2)

```
procedure Urut(input/output a1, a2, ..., an : integer)
Deklarasi
   i, j, imaks, temp : integer

Algoritma
  for i←n downto 2 do    { pass sebanyak n - 1 kali }
    imaks←1
    for j←2 to i do
      if aj > aimaks then
         imaks←j
      endif
    endfor
    { pertukarkan aimaks dengan ai }
    temp←ai
    ai←aimaks
    aimaks←temp

  endfor
```

Kasus Terbaik dan Terburuk:

Sorting tidak terikat pada kondisi awal data terurut atau tidak. Jumlah operasi perbandingan elemen untuk setiap perulangan ke-*i* adalah:

i = n → jumlah perbandingan = n – 1
i = n – 1 → jumlah perbandingan = n – 2
i = n – 2 → jumlah perbandingan = n – 3
Sampai..
i = 2 → jumlah perbandingan = 1

Jumlah seluruh operasi perbandingan elemen-elemen larik adalah:

$$T(n) = (n-1) + (n-2) + \ldots + 1 = \sum_{i=1}^{n-1} n - k = \frac{n(n-1)}{2}$$

LATIHAN PERHITUNGAN

```
procedure Kali(input x:integer, n:integer, output jumlah : integer)
{Mengalikan x dengan i = 1, 2, …, j, yang dalam hal ini j = n, n/2, n/4, …,1
  Masukan: x dan n (n adalah perpangakatan dua).
  Keluaran: hasil perkalian (disimpan di dalam peubah jumlah).
}
Deklarasi
   i, j, k : integer

Algoritma
   j ← n
   while  j ≥ 1 do
     for i ← 1 to j do
       x ← x * i
     endfor
     j ← j div 2
   endwhile
   { j > 1 }
   jumlah←x
```

TERIMA KASIH