Sesi 14 **Fungsi Kompleksitas Waktu**

Kuliah Online : Struktur Data

Dosen : Ir. Nixon Erzed MT

Pengertian Algoritma

Hasil analisis terhadap persoalan akan memberikan ***metoda-tepat*** bagaimana masalah akan diselesaikan dengan sistem komputer, yang merupakan logika proses/penyelesaian masalah.

Logika proses tersebut akan dituangkan dalam sekumpulan langkah-langkah terbatas. Setiap langkah mungkin memerlukan satu oprasi atau lebih.

Ciri-ciri algoritma :

1. Input
2. Output
3. Definite
4. Effective
5. Terminate

Input : Terdapat nol atau lebih masukan yang diberikan secara eksternal
Output : Sedikitnya terdapat satu keluaran yang harus dihasilkan
Definite : Harus secara sempurna menyatakan apa yang akan dikerjakan

Contoh :

a. Hitung 5/0
b. Tambahkan 6 atau 7 ke x

Contoh instruksi tersebut tidak dijinkan karena :

Kasus a : tidak jelas hasil operasinya.
Kasus b : tidak jelas yang harus dilakukan (tidak memiliki kepastian).

Effective : Setiap instruksi harus dapat dilakukan secara manual menggunakan pensil dan kertas dalam jumlah waktu yang berhingga.
Terminate : Harus berhenti setelah sejumlah terbatas operasi.

Langkah-langkah penyelesaian masalah yang disusun secara sistematis → urut-urutan logis
Penyajian → menggunakan kalimat-kalimat → algoritma
Penyajian dengan skema/diagram → flow chart

Contoh Algoritma :

Ingin ditemukan pembagi bulat dari sebuah bilangan bulat

```
Algoritma Euclidean (m, n : input,  FPB : output)
Deklarasi :
  r : integer

Algoritma
   While  n <> 0 do
            r ←m mod n
            m  ← n
            n  ← r
    end-while
   FPB  ←  m
 End-algoritma
```

Fungsi Kompleksitas Algoritma

→ MENGUKUR KINERJA ALGORITMA

Untuk sebuah problem → yang akan diprogram
→ Terdapat kemungkinan lebih dari satu algoritma penyelesaian yang benar.
→ Algoritma manakah yang terbaik?
→ Apakah algoritma yang dianggap terbaik → terabaik untuk semua keadaan?

Untuk menyelesaikannya harus diturunkan fungsi kompleksitas waktu untuk setiap algoritma

Dan lakukan analisis keadaan : terbaik (best case) terburuk (worst case,) dan rata-rata (average case)

Contoh :
Mencari nilai rata-rata dari n= 50 data yang tersimpan pada array A

	1 instruksi → 1 satuan waktu (pukul rata)		asumsin waktu eksekusi : Assign, operasi boolean: → ½ satuan waktu Aritmetika: →1 satuan waktu Looping for – next, operasi I/O : → 2 satuan waktu
	Misal n=4	n	
Algoritma Rata-1 : real Begin 1 Jml ⇐ 0 ---------- 2 For i ⇐1 to n ---------- 3 Do jml ⇐ jml + A[i] ---------- End-for 4 Rata_1 ⇐ jml/n ---------- End-Alg1	1 x 1detik 4 x 1detik 4 x 1detik 1 x 1detik	1 x 1 =1 n x 1 =n n x 1 =n 1 x 1 =1 f(n) = 2 + 2n	1 x ½ = ½ n x 2 = 2n n x 1½ = 1½ n 1 x 1½ = 1½ f(n) = 2 + 3½n jumlah ruang : jml, i, rata_1, n → @ 1 A[i] → n ruang Total ruang = 4 + n Perbaikan f(n) = 6 + 4½n
Algoritma Rata-2 : real Begin 1 i ⇐ 1 ---------- 2 Jml ⇐ 0 ---------- 3 While i <= n ---------- 4 Do jml <= jml + A[i] ---------- 5 i ⇐ i + 1---------- End-while 6 Rata_2 ⇐ jml/n ---------- End-Alg2	1 x 1 1 x 1 5 x 1 4 x 1 4 x 1 1 x 1	1 x 1 1 x 1 (n+1) x 1 n x 1 n x 1 1 x 1 f(n) = 4 + 3n	1 x ½ = ½ 1 x ½ = ½ (n+1) x ½ = ½ n + ½ n x 1½ = 1½ n n x 1½ = 1½ n 1 x 1 ½ = 1 ½ f(n) = 3 + 3 ½ n total ruang = 4 + n perbaikan f(n) = 7 + 4½ n

Mencari berapa banyak bilangan yang bernilai < x pada array A[i]

Algoritma Cari Nilai_lk_x Begin 1 i ⇐ 1 ---------- 2 Jml ⇐ 0 ---------- 3 While i <= n ---------- Do 4 IF A [i] < x 5 Then jml <= jml + 1 End-if 6 i ⇐ i + 1---------- End-while 7 Write (jml) End-Alg2	1 x ½ = ½ 1 x ½ = ½ (n+1)x ½ = ½n+½ n x ½ = ½n 50%n x 1½ = ¾ n n x 1½ = 1½n 1 x 2 = 2 f(n) = 3¼ n + 3½	Analisa keadaan Keadaan rata-rata kondisi IF (4) bisa benar, bisa salah maka instruksi baris 5 memiliki peluang → 50% f(n) = 3¼ n +3½ (AVERAGE CASE) Jika tidak ada data yang lebih kecil dari x, sehingga kondisi IF selalu salah maka intsruksi 5 tdk pernah dikerjakan → f(n) = 2½ n + 3½ (BEST CASE) Jika semua data lebih kecil dari x, sehingga kondisi IF selalu benar maka intsruksi 5 selalu dikerjakan (100%) → f(n) = 4n + 3½ (WORSTCASE)

Koreksi terhadap asumsin waktu eksekusi :→ mesti dengan argumentasi yang benar

Assign, operasi boolean : ½ satuan waktu
Aritmetika : 1 satuan waktu
Looping for – next : 2 satuan waktu

Apa yang dimaksudkan sebagai fungsi kompleksitas waktu ?

Fungsi yang ekivalen dengan : jumlah pelaksanaan instruksi dan pemakaian space memory

Masalah :

1. waktu yang diperlukan untuk melaksanakan instruksi → tidak selalu sama → diperlukan pemahaman tentang bagaimana prosesor melaksanakan instruksi
2. jumlah pemakaian space

Langkah-langkah analisis :

1. Definisikan, kelas-kelas instruksi dan asumsi waktu eksekusi
2. hitung kebutuhan ruang memory dan konversi ke waktu eksekusi
3. rumuskan kebutuhan waktu → menjadi fungsi kompleksitas waktu

Kelas-kelas instruksi	Asumsi waktu
1. instruksi matematis 2. instruksi logika, assign 3. instruksi I/O dan increment otomatis	1 satuan waktu ½ satuan waktu 2 satuan waktu 1 space ≡ 1 satuan waktu

Semakin detil dan teliti pengelompokan dan pendefinisian asumsi waktu
→ semakin teliti f(n) yang dihasilkan

Instruksi	Kali	waktu	Hasil
Algoritma Rata-1 : real Begin Jml <= 0 For i <= 1 to n Do jml <= jml + A[i] End-for Rata_1 <= jml/n End-Alg1	 1 n n 1	 ½ 2 1 ½ 1 ½	Space => 3+n f_1 (n) = ½ +2n+1½ n+1½ +3+n = 5 + 4½ n
Algoritma Rata-2 : real Begin i <= 1 Jml <= 0 While i ≤ n Do jml <= jml + A[i] i <= i + 1 End-while Rata_2 <= jml/n End-Alg2	 1 1 n+1 n n 1	 ½ ½ ½ 1 ½ 1 ½ 1 ½	Space => 3+n f_2(n) = ½ + ½ + ½ (n+1) +1½n+1½ n+1½ +3+n = 6 + 4½n

Algoritma 1 lebih baik dari algoritma 2 untuk semua kondisi

Diasumsikan setiap statement butuh 1 satuan waktu	1 space ≡ 1 satuan waktu

Semakin detil dan teliti pengelompokan dan pendefinisian asumsi waktu → semakin teliti f(n) yang dihasilkan

Instruksi	kali	waktu	Hasil
Algoritma Rata-1 : real Begin Jml <= 0 For i <= 1 to n Do jml <= jml + A[i] End-for Rata_1 <= jml/n End-Alg1	 1 n n 1	 1 1 1 1	Space => 3+n f_1 (n) = 1 + n + n + 1+3+n = 5 + 3n
Algoritma Rata-2 : real Begin i <= 1 Jml <= 0 While i ≤ n Do jml <= jml + A[i] i <= i + 1 End-while Rata_2 <= jml/n End-Alg2	 1 1 n+1 n n 1	 1 1 1 1 1 1	Space => 3+n f_2(n) = 1 + 1 + (n+1) +n+n+1 +3+n = 7 + 4n

Algoritma 1 lebih baik dari algoritma 2 untuk semua kondisi

Kelas-kelas instruksi 1. instruksi matematis 2. instruksi logika, assign 3. instruksi I/O dan increment otomatis	Asumsi waktu 1 satuan waktu ½ satuan waktu 2 satuan waktu 1 space ≡ 1 satuan waktu		
Instruksi	**Kali**	**waktu**	**Hasil**

Procedure BubleSort_max_min;
Const n
Var i, j : integer
Temp : integer
Begin
For i ⇐ 1 to n -1 do ----------
For j ⇐ 1 to n -1 do
If Data[j] < Data [j+1]
Then
Temp ⇐ data[j]
data [j] ⇐ data [j+1]
data [j+1] ⇐ temp
end-if
end-for
end –for
end-procedure
	
	n -1
	(n-1)(n-1)
	(n-1)(n-1)
	
	(n-1)(n-1).50%
	(n-1)(n-1).50%
	(n-1)(n-1).50%
		
		2
		2
		1½
		
		½
		1½
		1½
			Space => 4 + n
			
			2n – 2
			$2(n-1)^2$
			$1½(n-1)^2$
			
			$¼(n-1)^2$
			$¾(n-1)^2$
			$¾(n-1)^2$
			
			f(n) = $5¼(n-1)^2 + 2n - 2$
			= $5¼(n^2-2n+1) + 2n - 2$
			= $5¼n^2 - 10½n + 5¼ + 2n - 2$
			= $5¼n^2 - 8½n + 3¼$
			= $5{,}25\ n^2 - 8{,}5n + 3{,}25$
			
			Jika space memory diperhitungkan → (4+n)
			
			f(n) = $5{,}25\ n^2 - 7{,}5n + 7{,}25$
			
			best case :
			jika If Data[j] < Data [j+1] selalu salah → x 0%
			
			f(n) = $3{,}5\ n^2 - 5n + 1{,}5 + 4 + n$
			= $3{,}5\ n^2 - 4n + 5{,}5$
			
			worst case :
			jika If Data[j] < Data [j+1] selalu BENAR → x 100%
			
			f(n) = $7\ n^2 - 12n + 2{,}5 + 4 + n$
			= $7\ n^2 - 11n + 6{,}5$
			
			f(n) = $5{,}25\ n^2 - 7{,}5n + 7{,}25$
			f(n) = $1{,}375n^2 + 3{,}375n + 3{,}75$

Procedure Selection_max_min;
Const n
Var i, j, idxMax : integer
temp : integer
Begin
For i ⇐ 1 to n-1 do
idxMax ⇐ i
For j ⇐ i+1 to n do
If Data[j] > Data[idxMax]
Then
idxMax ⇐ j
end-if
end-for
 	
If idxMax ≠ i
Then
Temp ⇐ data[idxMax]
data [i] ⇐ data [idxMax]
data [idxMax] ⇐ temp
end-if
end –for
end-procedure
	
	n-1
	n-1
	(n-1)n/2 = ½ (n^2 – n)
	(n-1)n/2 = ½ (n^2 – n)
	
	(n-1)n/2= ½(n^2 –n).50%
	
	n – 1
	
	(n-1) . 50%
	(n-1) . 50%
	(n-1) . 50%
		
		2
		½
		2
		½
		
		½
		
		½
		
		½
		½
		½
			Space => 5 + n
			
			2n – 2
			½ n – ½
			n^2 – n
			¼ n^2 – ¼n
			
			n^2/8 – n/8
			
			½ n – ½
			
			¼ n – ¼
			¼ n – ¼
			¼ n – ¼
			
			f(n) = 11/8 n^2 + 11/8 n - 3¾
			= 1,375n^2 +2,375n -2,75
			
			Jika space memory diperhitungkan → (5+n)
			
			f(n) = 1,375n^2 +3,375n+3,75

Procedure BubleSort_max_min_2;
Const n
Var i, j : integer
Temp : integer
Begin
For i ⇐ 1 to n -1 do ----------
For j ⇐ 1 to n - i do
If Data[j] < Data [j+1]
Then
Temp ⇐ data[j]
data [j] ⇐ data [j+1]
data [j+1] ⇐ temp
end-if
end-for
end –for
end-procedure
	
	n -1
	(n-1)n/2 = ½ (n^2 – n)
	(n-1)n/2 = ½ (n^2 – n)
	
	(n-1)n/2= ½(n^2 –n).50%
	(n-1)n/2= ½(n^2 –n).50%
	(n-1)n/2= ½(n^2 –n).50%
		
		2
		2
		1½
		
		½
		1½
		1½
			Space => 4 + n
			
			2n – 2
			n^2 – n
			¾ n^2 – ¾ n = 0.75n^2 – 0.75n
			
			n^2/8 – n/8 = 0.125n^2 – 0.125n
			3n^2/8 –3n/8=0,375n^2 –0.375n
			3n^2/8 –3n/8=0,375n^2 –0.375n
			
			f(n) = 2.625n^2 –2,625n – 2
			
			Jika space memory diperhitungkan → (4+n)
			f(n) = 2.625n^2 –1.625n + 2

Mencari nilai rata-rata dari n= 50 data yang tersimpan pada array A

Asumsi : 1 instruksi → 1 detik n=4

	1 instruksi → 1 satuan waktu (pukul rata)		asumsin waktu eksekusi : Assign, operasi boolean: → ½ satuan waktu Aritmetika: →1 satuan waktu Looping for – next: → 2 satuan waktu
Algoritma Rata-1 : real Begin 1 Jml ⇐ 0 ---------- 2 For i ⇐1 to n ---------- 3 Do jml ⇐ jml + A[i] ---------- End-for 4 Rata_1 ⇐ jml/n ---------- End-Alg1	 1 x 1detik 4 x 1detik 4 x 1detik 1 x 1detik	 1 x 1 =1 n x 1 =n n x 1 =n 1 x 1 =1 f(n) = 2 + 2n	 1 x ½ n x 2 n x 1½ 1 x 1½ f(n) = 3½ n + 2
Algoritma Rata-2 : real Begin 1 i ⇐ 1 ---------- 2 Jml ⇐ 0 ---------- 3 While i ≤ n ---------- 4 Do jml <= jml + A[i] ---------- 5 i ⇐ i + 1---------- End-while 6 Rata_2 ⇐ jml/n ---------- End-Alg2	 1 x 1 1 x 1 5 x 1 4 x 1 4 x 1 1 x 1	 1 x 1 1 x 1 (n+1) x 1 n x 1 n x 1 1 x 1 f(n) = 4 + 3n	 1 x ½ 1 x ½ (n+1) x ½ n x 1½ n x 1½ 1 x 1½ f(n) = 3½ n + 3

Koreksi terhadap asumsin waktu eksekusi :

Assign, operasi boolean : ½ satuan waktu
Aritmetika : 1 satuan waktu
Looping for – next : 2 satuan waktu