Laprak 3
Analisis Algoritma

Archi Cantona Rusanggara

140810180050

TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2020

PENDAHULUAN

Setelah mengenal macam-macam kompleksitas waktu algoritma (best case, worst case, dan average case), dalam analisis algoritma kita selalu mengutamakan perhitungan worst case dengan alasan sebagai berikut:

- Worst-case running time merupakan upper bound (batas atas) dari running time untuk input apapun. Hal ini memberikan jaminan bahwa algoritma yang kita jalankan tidak akan lebih lama lagi dari worst-case
- Untuk beberapa algoritma, worst-case cukup sering terjadi. Dalam beberapa aplikasi pencarian, pencarian info yang tidak ada mungkin sering dilakukan.
- Pada kasus average-case umumnya lebih sering seperti worst-case. Contoh: misalkan kita secara random memilih angka dan mengimplementasikan insertion sort, average-case = worst-case yaitu fungsi kuadratik dari n.

Perhitungan worst case (upper bound) dalam kompleksitas waktu asimptotik dapat menggunakan Big-O Notation. Perhatikan pembentukan Big-O Notation berikut!

Misalkan kita memiliki kompleksitas waktu T(n) dari sebuah algoritma sebagai berikut:

$$T(n) = 2n^2 + 6n + 1$$

- Untuk yang besar, pertumbuhan $T(n)$ sebanding dengan n^2
- Suku $6n + 1$ tidak berarti jika dibandingkan dengan $2n^2$, dan boleh diabaikan sehingga T(n) = 2 + suku-suku lainnya.
- Koefisien 2 pada $2n^2$ boleh diabaikan, sehingga T(n) = O(n^2) -> Kompleksitas Waktu Asimptotik

DEFINISI BIG-O NOTATION

Definisi 1. $T(n) = O(f(n))$ artinya $T(n)$ berorde paling besar $f(n)$ bila terdapat konstanta C dan sedemikian sehingga

$$T(n) \leq C.f(n)$$

Untuk $n \geq n_0$

Jika dibuat semakin besar, waktu yang dibutuhkan tidak akan melebihi konstanta C dikalikan dengan $f(n)$, -> $f(n)$ adalah upper bound.

Dalam proses pembuktian Big-O, perlu dicari nilai dan nilai C sedemikan sehingga terpenuhi kondisi $T(n) \leq C.f(n)$.

SOAL 1

Tunjukan bahwa, $T(n) = 2n^2 + 6n + 1 = O(n^2)$

Penyelesaian:

Kita mengamati bahwa $n \geq 1$, maka $n \leq n^2$ dan $1 \leq n^2$ sehingga

$$2n^2 + 6n + 1 \leq 2n^2 + 6n^2 + n^2 = 9n^2, untuk\ n \geq 1$$

Maka kita bisa mengambil C=9 dan =1 untuk memperlihatkan:

$$T(n) = 2n^2 + 6n + 1 = O(n^2)$$

BIG-O NOTATION DARI POLINOMIAL BERDERAJAT M

Big-O Notation juga dapat ditentukan dari Polinomial n berderajat m, dengan TEOREMA 1 sebagai berikut:

Polinomial berderajat dapat digunakan untuk memperkirakan kompleksitas waktu asimptotik dengan mengabaikan suku berorde rendah

Contoh: $T(n) = 3n^3 + 6n^2 + n + 8 = O(n^3)$, dinyatakan pada

TEOREMA 1

Bila $T(n) = a_m n^m + a_{m-1} n^{m-1} + a_1 n + a_0$ adalah polinom berderajat m maka $T(n) = O(n^m)$

Artinya kita mengambil suku paling tinggi derajatnya ("Mendominasi") yang diartikan laju pertumbuhannya lebih cepat dibandingkan yang lainnya ketika diberikan sembarang besaran input. Besaran dominan lainnya adalah:

- Eksponensial mendominasi sembarang perpangkatan (yaitu, $y^n > n^p, y > 1$)
- Perpangkatan mendominasi $\ln n$ (yaitu, $n^p > \ln n$)
- Semua logaritma tumbuh pada laju yang sama (yaitua $\log(n) = b \log(n)$)
- log tumbuh lebih cepat daripada tetapi lebih lambat dari n^2

Teorema lain dari Big-O Notation yang harus dihafalkan untuk membantu kita menentukan nilai Big-O dari suatu algoritma adalah:

TEOREMA 2

Misalkan $T_1(n) = O(f(n))\ dan\ T_2(n) = O(g(n)), maka$

$(a)(i)\ T_1(n) + T_2(n) = O(\max(f(n), g(n)))$

$(ii) T_1(n) + T_2(n) = O(f(n) + g(n))$

(b) $T_1(n).T_2(n) = O(f(n))O(g(n)) = O(f(n).g(n))$

(c) $O(cf(n)) = O(f(n)), c\ adalah\ konstanta$

(d) $f(n) = O(f(n))$

Berikut adalah contoh soal yang mengaplikasikan Teorema 2 dari Big-O notation:

SOAL 2

Misalkan, $T_1(n) = O(n)\ dan\ T_2(n) = O(n^2), dan\ T_3(n) = O(mn)$

(a) $T_1(n) + T_2(n) = O(\max(f(n, n^2) = O(n^2))$……………………………………Teorema 2(a)(i)

(b) $T_2(n) + T_3(n) = O(n^2 + mn)$…………………………………………………………..Teorema 2(a(ii)

(c) $T_1(n).T_2(n) = O(n.n^2) = O(n^3)$……………………………………………………..Teorema 2(b)

SOAL 3

(d) $O(5n^2) = O(n^2)$………………………………………………………………………………..Teorema 2(c)

(e) $n^2 = O(n^2)$………………………………………………………………………………………..Teorema 2(d)

Aturan Menentukan Kompleksitas Waktu Asimptotik

- Cara I

  Jika kompleksitas waktu T(n) dari algoritma sudah dihitung, maka kompleksitas waktu asimptotiknya dapat langsung ditentukan dengan mengambil suku yang mendominasi fungsi T dan menghilangkan koefisiennya (sesuai TEOREMA 1)

  Contoh:

  Pada algoritma cariMax, $T(n) = n - 1 = O(n)$
- Cara 2

  Kita bisa langsung menggunakan notasi Big-O, dengan cara: Pengisian nilai (assignment), perbandingan, operasi aritmatika (+,-,/,*, div, mod), read, write, pengaksesan elemen larik, memilih field tertentu dari sebuah record, dan pemanggilan function/void membutuhkan waktu O(1)

SOAL 4

Tinjau potongan algoritma berikut:

read(x)	O(1)
x ← x + 1	O(1) + O(1)=O(1)
write(x)	O(1)

Kompleksitas waktu asimptotik algoritmanya (1)+(1)+(1)=(1)

Penjelasan:

$O(1) + O(1) + O(1) = O(\max(1,1)) + O(1)$ Teorema 2(a)(i)

………………………………….$= O(1) + O(1)$

………………………………..$= O(\max(1,1))$ Teorema 2(a)(ii)

………………………………..$= O(1)$

DEFINISI BIG-Ω DAN BIG-Θ NOTATION

Notasi Big-O hanya menyediakan batas atas (upper bound) untuk perhitungan kompleksitas waktu asimptotik, tetapi tidak menyediakan batas bawah (lower bound). Untuk itu, lower bound dapat ditentukan dengan Big-Ω Notation dan Big-θ Notation.

Definisi Big-Ω Notation:
$T(n) = \Omega(g(n))$ yang artinya $T(n)$ berorde paling kecil $g(n)$ bila terdapat konstanta C dan n_0 sedemikian sehingga

$$T(n) \geq C.(g(n))$$

Untuk $n \geq n_0$

Definisi Big-θ Notation:
$T(n) = \theta(h(n))$ *yang artinya* $T(n)$ *berorde sama dengan* $h(n)$ *jika* $T(n) = O(h(n))$ *dan* $T(n) = \Omega(g(n))$

SOAL 5

Tentukan Big-Ω dan Big- Θ Notation untuk $T(n) = 2n^2 + 6n + 1$

Penyelesaian:
Karena $2n^2 + 6n + 1 \geq 2$ untuk $n \geq 1$, dengan mengambil C=2, kita memperoleh

$$2n^2 + 6n + 1 = \Omega(n^2)$$

Karena$2n^2 + 6n + 1 = O(n^2)$ dan $2n^2 + 6n + 1 = \Omega(n^2)$, maka $2n^2 + 6n + 1 = \theta(n^2)$

Penentuan Big-Ω dan Big- dari Polinomial Berderajat m

Sebuah fakta yang berguna dalam menentukan orde kompleksitas adalah dari suku tertinggi di dalam polinomial berdasarkan teorema berikut:

TEOREMA 3

Bila $T(n) = a_m n^m + a_{m-1} n^{m-1} + a_1 n + a_0$ adalah polinom berderajat m maka $T(n) = n^m$

Contoh soal 6:
Bila $T(n) = 6n^4 + 12n^3 + 24n + 2$,
maka T(n) adalah berorde n^4, yaitu $O(n^4)$, $\Omega(n^4)$, $\Theta(n^4)$.

Latihan Analisa

Minggu ini kegiatan praktikum difokuskan pada latihan menganalisa, sebagian besar tidak perlu menggunakan komputer dan mengkoding program, gunakan pensil dan kertas untuk menjawab persoalan berikut!

1. Untuk $T(n) = 2 + 4 + 6 + 8 + 16 + \cdots + n^2$, tentukan nilai C, f(n), n_o, dan notasi Big-O sedemikian sehingga $T(n) = O(f(n))$ jika $T(n) \leq C\ untuk\ semua\ n \geq n_0$

2. Buktikan bahwa untuk konstanta-konstanta positif p, q, dan r:
   $T(n) = pn^2 + qn + r$ adalah $O(n^2), \Omega(n^2), dan\ \Theta(n^2)$

3. Tentukan waktu kompleksitas asimptotik (Big-O, Big-Ω, dan Big-Θ) dari kode program berikut:

```
for k ← 1 to n do
    for i ← 1 to n do
        for j ← to n do
            w_ij ← w_ij or w_ik and w_kj
        endfor
    endfor
endfor
```

4. Tulislah algoritma untuk menjumlahkan dua buah matriks yang masing-masing berukuran n x n. Berapa kompleksitas waktunya T(n)? dan berapa kompleksitas waktu asimptotiknya yang dinyatakan dalam Big-O, Big-Ω, dan Big-Θ?

5. Tulislah algoritma untuk menyalin (copy) isi sebuah larik ke larik lain. Ukuran elemen larik adalah n elemen. Berapa kompleksitas waktunya T(n)? dan berapa kompleksitas waktu asimptotiknya yang dinyatakan dalam Big-O, Big-Ω, dan Big-Θ?

6. Diberikan algoritma Bubble Sort sebagai berikut:

```
procedure BubbleSort(input/output a1, a2, ..., an : integer)
{ Mengurut tabel integer TabInt[1..n] dengan metode pengurutan bubble-
sort
  Masukan: a1, a2, ..., an
  Keluaran: a1, a2, ..., an (terurut menaik)
}
Deklarasi
   k : integer     { indeks untuk traversal tabel }
   pass : integer { tahapan pengurutan }
   temp : integer { peubah bantu untuk pertukaran elemen tabel }
Algoritma
    for pass ← 1 to n - 1 do
      for k ← n downto pass + 1 do
        if ak < ak-1 then
            { pertukarkan ak dengan ak-1 }
            temp ← ak
            ak ← ak-1
            ak-1←temp
        endif
      endfor
    endfor
```

(a) Hitung berapa jumlah operasi perbandingan elemen-elemen tabel!
(b) Berapa kali maksimum pertukaran elemen-elemen tabel dilakukan?
(c) Hitung kompleksitas waktu asimptotik (Big-O, Big-Ω, dan Big-Θ) dari algoritma Bubble Sort tersebut!

7. Untuk menyelesaikan problem X dengan ukuran N tersedia 3 macam algoritma:
(a) Algoritma A mempunyai kompleksitas waktu O(log N)
(b) Algoritma B mempunyai kompleksitas waktu O(N log N)
(c) Algoritma C mempunyai kompleksitas waktu O(N^2)
Untuk problem X dengan ukuran N=8, algoritma manakah yang paling cepat? Secara asimptotik, algoritma manakah yang paling cepat?

8. Algoritma mengevaluasi polinom yang lebih baik dapat dibuat dengan metode Horner berikut:

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \ldots + x(a_{n-1} + a_n x)))\ldots))$$

```
function p2(input x : real) → real
{ Mengembalikan nilai p(x) dengan metode Horner}

Deklarasi
  k : integer
  b1, b2, ..., bn : real

Algoritma
  bn ← an
  for k ← n - 1 downto 0 do
     bk ← ak + bk+1 * x
  endfor
  return b0
```

Hitunglah berapa operasi perkalian dan penjumlahan yang dilakukan oleh algoritma diatas, Jumlahkan kedua hitungan tersebut, lalu tentukan kompleksitas waktu asimptotik (Big-O)nya. Manakah yang terbaik, algoritma p atau p2?

Teknik Pengumpulan

- Semua jawaban ditulis di kertas dan dikumpulkan ke asisten praktikum pada akhir praktikum

Jawaban Latihan Analisa

Nama : Archi Cartona R.
NPM : 140810180050
Kelas : A

1) $T(n) = 2+4+6+8+16+ \ldots +2^n$
Deret $= \frac{a(r^n-1)}{(r-1)} = \frac{2(2^n-1)}{2-1} = 2^{n+1}-2$

Notasi big O → $O(2^n)$
$T(n) \leq 2^n$
$2^{n+1}-2 \leq C 2^n$
$2 - \frac{2}{2^n} \leq C$
$n = 1$

↳ $2 - \frac{2}{2} \leq C$
$C \geq 1$

2) Buktikan bahwa P, q, r positif
↳ $T(n) = Pn^2 + qn + r$ adalah
$O(n^2)$ $\Omega(n^2)$ $\Theta(n^2)$

• Big O
↳ $T(n) \leq C \cdot F(n)$
$Pn^2 + qn + r \leq Cn^2$
$\frac{Pn^2}{n^2} + \frac{qn}{n^2} + \frac{r}{n^2} \leq C$
$p + \frac{q}{n} + \frac{r}{n^2}$ → $\frac{1}{1} + \frac{1}{1} + \frac{1}{1} \leq C$
$n = 1$
$p \cdot q \cdot r = 1$
$C \geq 3$

• Big Θ
↳ Karena $\Theta(n^2)$ & $\Omega(n^2)$ terbukti dan berlangsung sama maka $\Theta(n^2)$ benar.

• Big Ω ($\Omega(n^2)$)
↳ $T(n) \geq C\ f(n)$
$Pn^2 + qn^2 + r \geq C \cdot n^2$
$n = 1$ P, q, r = 1
$P + \frac{q}{n} + \frac{r}{n^2} \geq C$
$1 + 1 + 1 \geq C$
$C \geq 3$

3) Tentukan Kompleksitas Waktu
↳ $W_{ij} \leftarrow W_{ik}$ dan W_{kj} berulang
Sebanyak n . n . n
$T(n) = n^3$

• Big O → $O(n^3)$
$n^3 \leq C n^3$
$C \geq 1$

• Big Ω → $\Omega(n^3)$
$n^3 \geq C n^3$
$C \leq 1$

• Big Θ → $\Theta(n^3)$
karena $O(n^3) = \Omega(n^3)$
maka $\Theta(n^3)$

4). Algoritma penjumlahan 2 matriks
```
For i ← 1 to n do
  for j ← 1 to n do
    Mij ← aij + bij
  end for
end for
```

$T(n) = n^2 + O(n^2)$
$n^2 \leq C n^2$
$C \geq 1$

$\Omega(n^2)$
$n^2 \geq C \cdot n^2$
$C \leq 1$

Θn^2 karena $O(n^2) = \Omega(n^2)$

5) Algoritma mencari baik
```
↳ for i ← 1 to n do
    ai ← bi
  end for
```
$T(n) = n$

↳ $O(n)$
$n \leq C \cdot n$
$C \geq 1$

↳ $\Omega(n)$
$n \geq C \cdot n$
$C \leq 1$

↳ Θ karena $O(n) = \Omega(n)$

6) a) Operasi Perbandingan
$T(n) = (n-1) + (n-2) + (n-3) + \ldots + 1$
$= \frac{n(n-1)}{2} = \frac{n^2-n}{2}$

b) Maksimal pertukaran terjadi ketika
↳ $\frac{n(n-1)}{2}$

c) Kompleksitas Waktu
↳ Best Case
Perbandingan → $n(n-1)$ kali
$T(n) = \frac{n(n-1)}{2} = \frac{n^2-n^2}{2}$

↳ Worst Case
Perbandingan → $\frac{n(n-1)}{2}$

Assignment → $\frac{3n(n-1)}{2}$

$T_{max}(n) = \frac{n(n-1)}{2} + \frac{3n(n-1)}{2} = \frac{4n(n-1)}{2}$
$= 2n^2 - 2n$

↳ $O(n^2)$
$2n^2 - 2n \leq C \cdot n^2$
$2 - \frac{2}{n} \leq C$ ex: n=1
$2 - 2 \leq 1$
$C \geq 0$

↳ $\Omega(n^2)$
$\frac{n^2-n}{2} \geq C \cdot n^2$
$\frac{1}{2} - \frac{1}{2n} \geq C$ ex: n=1
$\frac{1}{2} - \frac{1}{2} \geq C$ ↳ $C \leq 0$

↳ $\Theta(n^2)$ karena $O(n^2)$ sama $\Omega(n^2)$

7) Untuk menyelesaikan problem x dengan ukuran N tersedia 3 macam algoritma :
a) Algoritma A mempunyai kompleksitas waktu $O(\log N)$
b) Algoritma B mempunyai kompleksitas waktu $O(N \log N)$
c). Algoritma C mempunyai kompleksitas waktu $O(N^2)$
N = 8
A → $O(\log 8) = O(3 \log 2)$
B → $O(8 \log 8) = O(24 \log 2)$
C → $O(8^2) = O(64)$

↳ maka A tercepat dengan jika $\log 2 = 0.301$

8) Operasi Assignment
↳ P_2
$b_n \leftarrow a_n$ 1 kali
$b_k \leftarrow a_k + b_{k+1} \cdot x$ n kali

$T(n) = 1 + n$
$O(n)$ untuk P_2
Algoritma P
Pertambahan = n kali
Perkalian = n kali
$T(n) = 2n$

↳ maka P2 lebih baik, karena lebih kecil.

Selamat bekerja dengan jujur dan mandiri