Lembar Kerja Responsi 3
Mata Kuliah KOM 401 Analisis Algoritme
Semester Ganjil Tahun Akademik 2020/2021

Asisten Praktikum:

1. **M. Faishal Wicaksono R.**
2. **Zaki Muttaqin**

Nama : Shibgotalloh Sabilana
NIM : G64180002

1. Diberikan tabel yang berisi kompleksitas waktu T(n) yang digunakan oleh sebuah algoritme untuk menyelesaikan sebuah masalah berukuran n. Pilihlah suku yang paling dominan dan tentukan kompleksitas big-O dari setiap algoritme. Tuliskan pula urutan algoritme mulai dari kompleksitas yang paling rendah.

No	T(n)	Suku Dominan	O(...)	Urutan
1	$5 + 10n + n^3$	n^3	$O(n^3)$	4
2	$n^{1.75} + \log n^2 + n \log n$	n^1.75	O(n^1.75)	3
3	$n \log_3 n + n \log_2 n$	$n \log_2 n$	$O(n \log_2 n)$	2
4	$\log_2\log_2\log_2 n + (\log_2 n)^3$	$(\log_2 n)^3$	$O((\log_2 n)^3)$	1
5	$10n! + n^{1000} + 10n$	10n!	O(10n!)	9
6	$\log_{10} n^{1000} + 99n^{10}$	$99n^{10}$	$O(99n^{10})$	5
7	$100n + n^3 + 75$	n^3	$O(n^3)$	4
8	$100^n + n^{100} + 100$	100^n	$O(100^n)$	7
9	$2^n + 3^n + n^{3000}$	3^n	$O(3^n)$	6

10	$n! + 2^n + 15$	$n!$	$O(n!)$	8

2. Apakah pernyataan berikut benar? Jika salah, tuliskanlah pernyataan yang benar.

Pernyataan	**Benar atau Salah**	**Perbaikan**
Aturan penjumlahan: $O(f + g) = O(f) + O(g)$	salah	Harusnya $O(f + g) =$ max $\{O(f), O(g)\}$
Aturan perkalian: $O(f.g) = O(f) . O(g)$	benar	-
Aturan Transivitas: jika $g = O(f)$ dan $h = O(f)$ maka $g = O(h)$	salah	Karena belum pasti jika $g = O(f)$ bakal $g = O(h)$
$5n + 8n2 + 1000n3 = O(n4)$	salah	Harusnya $O(n3)$
$5n + 8n2 + 100n3 = O(n2 \log n)$	salah	Harusnya $O(n3)$

3. Algoritme A dan B memerlukan $T_A(n) = n^2 \log_{10} n$ dan $T_B(n) = 5n^2$ untuk suatu masalah dengan ukuran *n*. Pilihlah algoritme yang lebih baik berdasarkan konsep big-O. Carilah ukuran masalah $n_0$sedemikian sehingga untuk $n > n_0$salah satu algoritme pasti lebih baik daripada algoritme lainnya. Jika ukuran masalah $n \leq 10^9$, algoritme manakah yang lebih baik?

Jika dilihat dari big O algoritma B lebih baik karena memiliki nilai O(n^2) sedangkan algoritma A memiliki nilai O(n^2 log n).

Jika ukuran masalah sebesar 10^9 maka :

$T_A(10^9) = 10^{9\,2}\log_{10} 10^9 = 9 . 10^{18}$

$T_B(10^9) = 5\,10^{9\,2} = 5 . 10^{18}$

Algoritma B tetap lebih baik

Algoritma B lebih baik dari algoritma A saat $n_0 = 10^{50}$

4. Dua buah algoritme A dan B digunakan untuk memproses sebuah pangkalan data yang besar, mengandung 10^{12} *record*. Algoritme memerlukan $T_A(n) = 2.5n$, sedangkan $T_B(n) = 0.1\ n \log_2 n$. Manakah algoritme yang lebih baik sesuai dengan pengertian big-O? Kapan salah satu algoritme lebih baik daripada algoritme yang lain?

Jika dilihat dari **big O** maka algoritma A lebih baik karena memiliki nilai O(n) sedangkan algoritma B memiliki nilai O(n log n).

Jika ukuran masalah sebesar 10^{12} maka :
$T_A(10^{12}) = 2.5 * 10^{12}$
$T_B(10^{12}) = 0.1 * 10^{12} \log_2 10^{12} = 0.99 * 10^{12}$
Maka algoritma B lebih baik

Algoritma A lebih baik dari algoritma B saat $n_0 = 2^{25}$

5. Tentukan kompleksitas dari potongan kode berikut:

```
for( i = n; i > 0; i /= 2){
 for( j = 1; j < n; j *=2 ){
 for( k = 0; k < n; k += 3){
... // operasi yang jumlahnya konstan
 }
 }
}
```

Kompleksitas yang diperoleh = $\log_2 n * \log_2 n * n/2$

6. Dibawah ini terdapat algoritme X dan algoritme Y. Berapakah nilai n_0 sedemikian sehingga $f_x(n) > f_y(n)$ atau sebaliknya?

$f_x(n)$	$f_y(n)$	n_0
$n^2 + n$	$-n + 80$	8
$2n^2 - 7$	$5n + 5$	4
$2(n^2 - 2n)$	$n^2 + 21$	7
$n(3n - 20n)$	$4n + 27$	9
$(n - 8)^2$	16	12

7. Berikut ini terdapat dua buah algoritme, algoritme tersebut memiliki suatu fungsi untuk melakukan suatu pekerjaan:

$f_a(n) = 10\ 000(n - 1000)$

$f_b(n) = \frac{1000\ 000}{100n + 1}$

a. Berapakah nilai n_0 sedemikian sehingga $f_b(n) > f_a(n)$?

N0 didapat sebesar 1000 dengan cara melakukan pendekatan $f_b(n) = f_a(n)$

b. Apabila anda memiliki ukuran data sebanyak 10 000, algoritme manakah yang lebih baik untuk digunakan?

$f_a(10.000) = 10\ 000(n - 1000) = 90000000$

$f_b(10000) = \frac{1000\ 000}{100(10000) + 1} = 0.99$

Algoritma yang lebih baik digunakan adalah f_b

c. Misalkan algoritme tersebut akan dibuat menjadi program yang akan berjalan di komputer yang mampu mengeksekusi 10^8instruksi dalam waktu satu detik. Berapakah waktu yang diperlukan oleh kedua algoritme tersebut (dalam satuan detik) untuk memproses sebanyak data yang ada pada tabel? Tuliskan jawaban Anda dalam tabel berikut:

n	**Waktu Eksekusi Algoritme A**	**Waktu Eksekusi Algoritme b**
2 000	10^{-1}	$4.9 \cdot 10^{-8}$
5 000	$4 \cdot 10^{-1}$	$1.9 \cdot 10^{-8}$
10 000	$9 \cdot 10^{-1}$	$0.9 \cdot 10^{-8}$
75 000	7.4	$0.1 \cdot 10^{-8}$
1 000 000	99.9	$9.9 \cdot 10^{-8}$

8. Manakah notasi asimtotik yang paling cocok digunakan untuk menunjukkan relasi di antara dua fungsi berikut ini:

No	***f*(*n*)**	***g*(*n*)**	***f*(*n*) = … *g*(*n*)**
1.	1	99	Big theta
2.	n^k, k adalah konstanta > 1	c^n, c adalah konstanta > 1	Big Oh
3.	$n^3 + 2n^2 + 10$	$7500n^2 + 89000n + 50$	Big omega
4.	$\log(n)$	$\log(n^k)$	Big theta
5.	$\sqrt{n} + 2$	$\log(n)$	Big omega
6.	$\log(n)$ + n	$\log(n^2)$	Bog omega
7.	~~$\sqrt{n + 2}$~~	$\sqrt{n} + 2$	Big theta
8.	$n!$	$(n+1)!$	Big theta

9.	c^n, c adalah konstanta > 1	$n!$	Big O
10.	n^n	$n!$	Big omega

9. Identifikasi kondisi yang menyebabkan *worst case, average case*, dan *best case* pada linear search berikut, dengan n adalah ukuran input dan x adalah target pencarian.

```
int search(int arr[], int n, int x)
{
    int i;
    for (i=0; i<n; i++)
    {
       if (arr[i] == x)
         return i;
    }
    return -1;
}
```

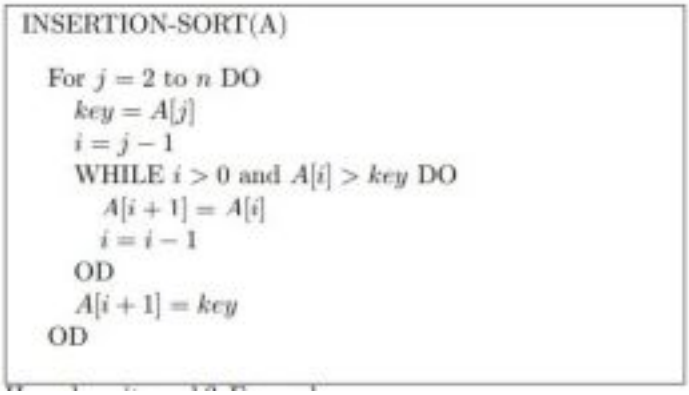
- Worst case = ketika data yang dicari tidak ada sehingga semua proses harus dieksekusi terlebih dahulu. Parameter kompleksitas menggunakan Big O
- Average case = ketika data yang dicari berada di posisi pertengahan. Parameter kompleksitas menggunakan big theta
- Best case = ketika data yang dicari berada di posisi pertama. Parameter kompleksitas menggunakan Big omega

10. *Binary search tree* (BST) adalah salah satu bentuk struktur data yang menggambarkan hubungan hierarki antar elemen-elemennya dengan mengikuti aturan sebagai berikut.
    a. Semua data dibagian kiri sub-tree dari node t selalu lebih kecil dari data dalam node t itu sendiri.
    b. Semua data dibagian kanan sub-tree dari node t selalu lebih besar atau sama dengan data dalam node t.

    Tentukan skenario *worst case* dan *best case* serta kompleksitasnya ketika anda melakukan pencarian suatu elemen pada BST.

- Best case akan terjadi apabila yang dicari terletak di posisi pertama (root)
- Worst case terjadi apabila yang dicari terletak di posisi paling bawah (child)

11. Insertion sort merupakan algoritma pengurutan yang menghasilkan list terurut dengan mengurutkan satu elemen dalam satu waktu. Cara kerja insertion sort mirip seperti pengurutan sebuah deck kartu yang digambarkan oleh pseudocode dibawah ini.



```
INSERTION-SORT(A)

  For j = 2 to n DO
    key = A[j]
    i = j − 1
    WHILE i > 0 and A[i] > key DO
      A[i + 1] = A[i]
      i = i − 1
    OD
    A[i + 1] = key
  OD
```

Jelaskan secara singkat bagaimana algoritma ini bekerja berdasarkan pseudocode di atas dan lakukan analisis skenario yang dapat menyebabkan worst case dan best case pada algoritma ini.

Sebagai contoh kita akan mengurutkan sebuah array yg berisi angka dari angka terkecil ke angka yang besar : [4, 7, 2, 11, 7, 3]

Pertama identifikasi angka urutan ke i + 1 (disini angka 7) lalu bandingkan dengan angka sebelah kirinya, apabila angka i+1 lebih kecil maka akan ditukar dengan angka sebelah kirinya. Apabila tidak maka fungsi berhenti dan dilanjutkan ke list array selanjutnya :
[4, 7, 2, 11, 5, 3] → [4, 7, 2, 11, 5, 3]

Dilanjutkan ke angka i+2 (angka 2) :
[4, 7, 2, 11, 5, 3] → [4, 2, 7, 11, 5, 3] → [2, 4, 7, 11, 5, 3]

Dilanjut angka selanjutnya (angka 11) :
[2, 4, 7, 11, 5, 3] → [2, 4, 7, 11, 5, 3]

Selanjutnya angka 5 :
[2, 4, 7, 11, 5, 3] → [2, 4, 7, 5, 11, 3] → [2, 4, 5, 7, 11, 3]

Dan terakhir angka 3 :
[2, 4, 5, 7, 11, 3] → [2, 4, 5, 7, 3, 11] → [2, 4, 5, 3, 7, 11] → [2, 4, 3, 5, 7, 11] → [2, 3, 4, 5, 7, 11]

Fungsi pun selesai dan output yang dihasilkan adalah [2, 3, 4, 5, 7, 11]

- Kondisi worst case terjadi apabila array yang didapat merupakan kebalikan dari yang seharusnya. Contoh : [11, 7, 5, 4, 3, 2]
- Best case terjadi jika array yang diperoleh sudah dalam bentuk yang benar