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

Asisten Praktikum:

1. **Alfian Hamam Akbar**
2. **Hilmi Farhan Ramadhani**

---

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 tinggi.

No	T(n)	Suku Dominan	O(...)	Urutan
1	$5 + 10n + n^3$	n^3	$O(n^3)$	5
2	$n^{1.75} + \log n^2 + n \log n$	$n^{1.75}$	$O(n^{1.75})$	6
3	$n \log_3 n + n \log_2 n$	$n \log_2 n$	$O(n \log_2 n)$	7
4	$\log_2 \log_2 \log_2 n + (\log_2 n)^3$	$(\log_2 n)^3$	$O((\log_2 n)^3)$	8
5	$10n! + n^{1000} + 10n$	$10n!$	$O(n!)$	1
6	$\log_{10} n^{1000} + 99n^{10}$	$99n^{10}$	$O(n^{10})$	4
7	$100n + n^3 + 75$	n^3	$O(n^3)$	5
8	$100^n + n^{100} + 100$	100^n	$O(100^n)$	2
9	$2^n + 3^n + n^{3000}$	3^n	$O(3^n)$	3
10	$n! + 2^n + 15$	$n!$	$O(n!)$	1

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	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	jika g = O(f) dan f = O(h), maka g = O(h)
5n + 8n2 + 1000n3 = O(n4)	Benar	
5n + 8n2 + 100n3 = O(n2 log n)	Salah	$5n + 8n^2 + 100n^3 = O(n^3)$

3. Algoritme A dan B memerlukan $T_A(n) = 0.1n^2 \log_{10} n$ dan $T_B(n) = 2.5n^2$ 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?

Jawaban:

Algoritme B lebih baik jika melihat dari big-O. Algoritme B akan lebih baik dari algoritme A ketika $T_B(n) \leq T_A(n)$, yaitu ketika $2.5n^2 \leq 0.1n^2 \log_{10} n$. Hasilnya adalah $\log_{10} n \geq 25$ atau $n > n_0 = 10^{25}$. Jika $n \leq 10^9$, algoritme yang lebih baik adalah algoritme A.

4. Dua buah algoritme A dan B digunakan untuk memproses sebuah pangkalan data yang besar, mengandung 10^{12} *record*. Algoritme memerlukan $T_A(n) = 0.1\ n \log_2 n$, sedangkan $T_B(n) = 5n$. Manakah algoritme yang lebih baik sesuai dengan pengertian big-O? Kapan salah satu algoritme lebih baik daripada algoritme yang lain?

Jawaban:

Algoritme B yang memiliki kompleksitas O(n) lebih baik daripada algoritme A yang memiliki kompleksitas O(n log n). Algoritme B melebihi algoritme A ketika $n \geq 2^{50} \approx 10^{15}$. Jika data yang diolah 10^{12}, algoritme yang lebih baik adalah algoritme A.

5. Tentukan kompleksitas dari potongan kode berikut:

```
for( i = n; i > 0; i /= 2){                    // log2n
     for( j = 1; j < n; j *=2 ){               // log2n
          for( k = 0; k < n; k += 2){          // n/2
```

```
                ... // operasi yang jumlahnya konstan
            }
        }
}
```

Jawaban: $\log_2 n * \log_2 n * n/2 = O(n(\log_2 n)^2)$

6. Tentukan kompleksitas dari potongan kode berikut:

```
for( bound = 1; bound <= n; bound *= 2){          // log2n
     for( j = 0; j < n; j += 2){                  // n
          ... // operasi yang jumlahnya konstan
     }

     for( j = 1; j < n; j *= 2){                  // log2n
          ... // operasi yang jumlahnya konstan
     }
}
```

Jawaban: $\log_2 n * \max(n, \log_2 n) = \log_2 n * n = n \log_2 n$

7. 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

n_0 dapat dicari dengan mencari nilai n terbesar non negatif pada $f_a(n) = f_b(n)$.

8. 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 $f_b(n) > f_a(n)$?
   1000. Dapat dicari dengan mencari n terbesar non negatif pada $f_b(n) = f_a(n)$.

b. Apabila anda memiliki ukuran data sebanyak 10 000, algoritme manakah yang lebih baik untuk digunakan?
   $f_a(10\,000) = 10(10\,000 - 1000) = 90\,000$
   $f_b(10\,000) = \frac{1000\,000}{100(10\,000) + 1} = 0.99$
   Dari jumlah proses di atas dapat disimpulkan bahwa jika terdapat data sebanyak 10 000 lebih baik menggunakan algoritma B.

c. Misalkan algoritme tersebut akan dibuat menjadi program yang akan berjalan di komputer yang mampu mengeksekusi 10^8 instruksi 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}	4.9×10^{-8}
5 000	4×10^{-1}	1.9×10^{-8}
10 000	9×10^{-1}	0.9×10^{-8}
75 000	7.4	0.1×10^{-8}
1 000 000	99.9	9.9×10^{-11}

Jumlah didapatkan dari n yang dimasukan ke dalam fungsi lalu dibagi 10^8

9. 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	Θ

2.	n^k, k adalah konstanta > 1	c^n, c adalah konstanta > 1	O
3.	$n^3 + 2n^2 + 10$	$7500n^2 + 89000n + 50$	Ω
4.	$^2\log(n)$	$^{10}\log(n)$	Θ
5.	$\sqrt{n} + 2$	$\log(n)$	Ω
6.	$\log(n) + n$	$\log(n^2)$	Ω
7.	$\sqrt{n+2}$	$\sqrt{n} + 2$	Θ
8.	$n!$	$(n+1)!$	Θ
9.	c^n, c adalah konstanta > 1	$n!$	O
10.	n^n	$n!$	Ω

10. 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;
}
```

Jawaban:

- **Worst Case** : Ketika tidak ada data yang sama dengan target, akan dilakukan pencarian dari awal sampai size, kompleksitas O(n).
- **Average Case** : Ketika data berada di bagian tengah program, kompleksitas Θ(n).
- **Best Case** : Ketika data yang sama berada di paling depan (index 0), kompleksitas Ω(1)

11. *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.

Jawaban:

- Worst case: melakukan pencarian elemen yang terletak pada leave tree. Nilai kompleksitas bisa dipengaruhi bentuk tree, namun secara umum O(log n).
- Best case: melakukan pencarian elemen yang terletak pada root tree, O(1).

12. 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.

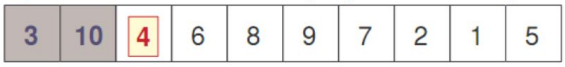
Jawaban:

- Worst case: list atau array terurut dalam kondisi terbalik dari yang diinginkan, contoh list terurut yang diinginkan 1234 namun list dalam kondisi 4321.
- Best case: list atau array sudah dalam kondisi terurut

Berikut visualisasinya:

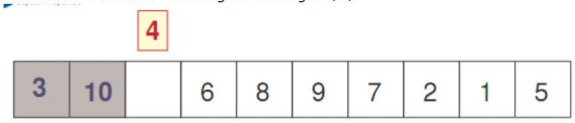
Contoh tahap-tahap insertion sort:

Bagian biru/abu-abu (dua bilangan pertama) sekarang dalam keadaan terurut secara relatif.

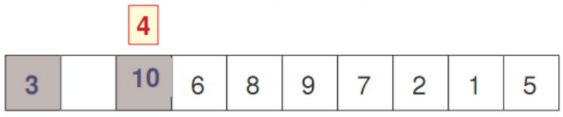


Berikutnya, kita perlu menyisipkan bilangan ketiga (4) ke dalam bagian biru/abu-abu sehingga setelah penyisipan tersebut, bagian biru/abu-abu tetap dalam keadaan terurut secara relatif.

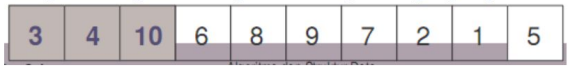
Caranya :
Pertama : Ambil bilangan ketiga (4)



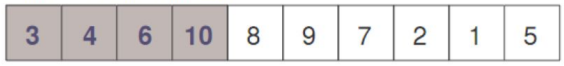
Kedua : Geser bilangan kedua (10) sehingga ada ruang untuk disisipi



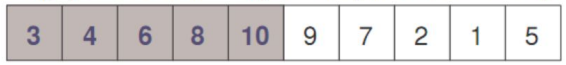
Ketiga : Sisipkan bilangan (4) ke posisi yang tepat

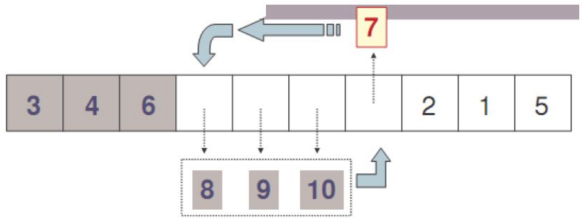


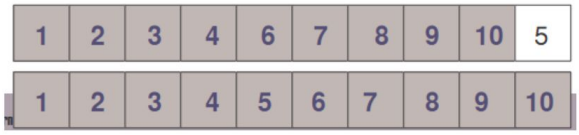
Sekarang ketiga bilangan sudah terurut secara relatif dan kita sisipkan bilangan keempat kepada tiga bilangan pertama tersebut. Setelah penyisipan, empat bilangan pertama haruslah dalam keadaan terurut secara relatif.



Ulangi proses tersebut sampai bilangan terakhir disisipkan







Proses sorting selesai.

sumber : https://sites.google.com/site/riksongultom/algoritma-pemrograman/insertion-sort