



Ahmad Nizar Sauki - 2306152046 Nizar ▾



Home > My courses > PROG. S1 FAK. REGULER > REG - Gasal 2024/2025 > [Reg] Struktur Data & Algoritma (A,B,C,D,E,F) ... > Pekan 1-2: Pengantar & Analisis Algoritma > CP01 Analisis Algoritma

Started on Tuesday, 3 September 2024, 1:04 PM

State Finished

Completed on Tuesday, 3 September 2024, 8:13 PM

Time taken 7 hours 8 mins

Grade **8.00** out of 10.00 (**80%**)

Question 1

Correct

Mark 1.00 out of 1.00

Algoritma Binary Search berikut (dari materi kuliah) melakukan proses pencarian data x berdasar data array integer a, sebagai berikut.

```
public static int binarySearch (int[] a, int x ) throws ItemNotFound {
    int low = 0;
    int high = a.length - 1;
    int mid;

    while( low <= high ) {
        mid = (low + high) / 2;    // mendapatkan titik tengah
        if (a[mid].compareTo (x) < 0) {
            low = mid + 1;
        } else if (a[mid].compareTo (x) > 0) {
            high = mid - 1;
        } else {
            return mid;
        }
    }
    throw new ItemNotFound( "BinarySearch fails" );
}
```

Jika pada method binarySearch pencarian X dilakukan pada array yang berukuran satu triliun (10^{12}), berapa kalikah perintah "mid = (low + high) / 2;" dilakukan hingga eksekusi selesai (jawab untuk kasus **terbaik** dan kasus **terburuk**) ?

- ☐ Terbaik dan terburuk sama yaitu sekitar setengah triliun kali.
- ☐ Terbaik dan terburuk sama yaitu sekitar 40 kali.
- ☐ Terbaik 1 kali terburuk satu triliun kali.
- ☒ Terbaik 1 kali terburuk sekitar 40 kali. ✓

Your answer is correct.

Tujuan: memahami mengapa Binary Search bersifat logaritmis.

Penjelasan Solusi: Yang terbaik, langsung ketemu, sementara yang terburuk, diiterasi hingga interval tinggal 1 data. Untuk terburuk ini terjadi iterasi halving yang bersifat logaritmis. Dari 1 triliun, $\log_2(10^{12})$ adalah sekitar 40.

Note: Menghitung log2 angka besar dengan mudah sbb.

Ingat bahwa $\log_2(10^3) \approx 10$.

Dan, $\log_2(10^{12}) = \log_2(10^3 \cdot 10^3 \cdot 10^3 \cdot 10^3)$
 $= \log_2(10^3) + \log_2(10^3) + \log_2(10^3) + \log_2(10^3) \approx 40.$

Jika dihitung yang lebih akurat, $\log_2(10^{12}) = 39.86314$

The correct answer is: Terbaik 1 kali terburuk sekitar 40 kali.

Question 2


Correct

Mark 1.00 out of 1.00

Diketahui terdapat beberapa algoritma untuk tujuan komputasi yang sama dan diimplementasikan serta diukur waktu running timenya terhadap data yang berukuran N adalah sebagai pada tabel berikut.

Nama Algoritma	Fungsi waktu eksekusi terhadap ukuran data N
Algo1	$2N^3 + 10N^2 + 200$
Algo2	$1000N^2 + 50N + 5000$
Algo3	$2N^3 + 200N + 10$

Kesimpulan manakah yang **benar**?

- ☒ Ketika N diperbesar beberapa kali dari sebelumnya maka waktu eksekusi Algo1 dan Algo3 keduanya mengalami peningkatan yang sama walaupun waktu eksekusinya berbeda. 
- ☐ Ketiganya memiliki kompleksitas waktu eksekusi yang sama.
- ☐ Algo2 akan selalu dieksekusi lebih cepat dari lainnya untuk semua kemungkinan ukuran data N .
- ☐ Untuk setiap ukuran data N yang berbeda Algo1 dan Algo3 memiliki waktu eksekusi yang selalu sama.

Your answer is correct.

Tujuan: Memahami bahwa notasi big-O mencerminkan growth-rate bukannya actual running time.

Penjelasan solusi: Kompleksitas Algo1: $O(N^3)$, Algo2: $O(N^2)$, dan Algo3: $O(N^3)$.

Kesamaan Algo1 dan Algo3 dari growth-rate-nya bukan actual running time.

The correct answer is: Ketika N diperbesar beberapa kali dari sebelumnya maka waktu eksekusi Algo1 dan Algo3 keduanya mengalami peningkatan yang sama walaupun waktu eksekusinya berbeda.

Question 3

Correct

Mark 1.00 out of 1.00

Algoritma-algoritma berdasarkan dua parameter berbeda N dan M memiliki kompleksitas-kompleksitas dalam notasi Big-Oh sebagai berikut.

Nama Algoritma	Kompleksitas Waktu Eksekusi dalam Notasi Big-Oh
Olah1	$O(N \log(M) \log(N^M))$
Olah2	$O(N^2 \log(N) M \log(M^4))$
Olah3	$O(N \log(10^N) M \log(M))$

Jika diurutkan menurut tingkat kompleksitas dari yang paling **rendah** ke yang paling **kompleks** adalah:

- ☒ Olah1 < Olah3 < Olah2 ✓
- ☐ Olah2 < Olah1 < Olah3
- ☐ Olah3 < Olah1 < Olah2
- ☐ Olah2 < Olah3 < Olah1

Your answer is correct.

Tujuan: Memahami perbedaan tingkatan pada kelas-kelas notasi Big-O pada masalah lebih dari satu variabel (suku-suku multi-variabel).

Penjelasan solusi:

$$O(N \log(M) \log(N^M)) \Rightarrow O(N \log(N) M \log(M))$$

$$O(N^2 M \log(N) \log(M^4)) \Rightarrow O(N^2 \log(N) M \log(M))$$

$$O(N \log(10^N) M \log(M)) \Rightarrow O(N^2 M \log(M))$$

The correct answer is: Olah1 < Olah3 < Olah2

Question 4

Correct

Mark 1.00 out of 1.00

Perhatikan method-method berikut yang akan berjalan berdasarkan parameter bilangan bulat N :

```
void proses1(int N) {  
    for (int i = 0; i < N; i+= 5) {  
        for (int j = 1; j < N; j *= 2) {  
            for (k = i; k < N; k++) {  
                performThis();  
            }  
        }  
    }  
}
```

```
void proses2(int N) {  
    for (int i = 0; i < 1000; i++) {  
        for (int j = N*N; j > 0; j -= N) {  
            for (k = i; k < N*N; k++) {  
                performThis();  
            }  
        }  
    }  
}
```

Method `performThis()` yang dipanggil kedua method ini adalah suatu proses konstan (tidak tergantung ukuran data N).

Dari pernyataan-pernyataan berikut ini manakah yang benar?

- ☐ Kedua method memiliki kompleksitas $O(N^2 \log N)$.
- ☐ Kedua method memiliki kompleksitas $O(N^3)$.
- ☐ Kompleksitas `proses1` adalah $O(N^3)$ sementara kompleksitas `proses2` $O(N^2 \log N)$.
- ☒ Kompleksitas `proses2` adalah $O(N^3)$ sementara kompleksitas `proses1` $O(N^2 \log N)$. ✓

Your answer is correct.

Tujuan: Memahami cara mendapatkan $O(g(N))$ dari proses yang berinteraksi secara konstan (tidak bergantung dari n) dan/atau dari proses yang berinteraksi bergantung secara linier dari dan/atau dari proses yang berinteraksi bergantung secara doubling/having (logaritmis) dari N .

Penjelasan solusi:

`proses1` melakukan for-loop dengan variabel i sebanyak $N/5$ berarti linear, lalu for-loop kedua bersifat doubling, dan for-loop terdalam adalah linear. Sehingga, kompleksitas `proses1` menjadi $O(N^2 \log N)$.

Sementara itu, `proses2` melakukan for-loop dalam jumlah iterasi yang konstan, kemudian for-loop kedua melakukan iterasi sebanyak N dan for-loop terdalam sebanyak N^2 . Sehingga kompleksitas `proses2` menjadi $O(N^3)$.

The correct answer is: Kompleksitas `proses2` adalah $O(N^3)$ sementara kompleksitas `proses1` $O(N^2 \log N)$.

Question 5

Correct

Mark 1.00 out of 1.00

Suatu proses **kuadratis** terhadap ukuran datanya, dijalankan pada berbagai ukuran data berukuran 10000 sebanyak 2 detik. Berapa lamanya proses tersebut jika dijalankan dengan data berukuran 25000?

- ☐ 6.25 detik
- ☐ 25 detik
- ☐ 62.5 detik
- ☒ 12.5 detik ✓

Your answer is correct.

Tujuan: Memahami cara memprediksi running time pada suatu ukuran data N , berdasarkan growth rate $O(g(N))$ dan running time pada suatu harga N_0 . **Pembahasan solusi:** karena kuadratis maka jika ukuran data dikalikan faktor x (asumsi: ukuran data besar) maka untuk waktu eksekusi meningkat x^2 kali. Dalam soal ini ukuran data cukup besar. Jika untuk ukuran data semula waktunya 2 detik, maka untuk ukuran data yang dikalikan 2.5 dari ukuran semula, waktu eksekusi dilakukan $2,5^2 = 6.25$ kali. Karena sebelumnya 2 detik, sekarang menjadi 12.5 detik.

The correct answer is: 12.5 detik

Question 6


Incorrect

Mark 0.00 out of 1.00

Suatu algoritma bekerja berdasarkan dua array yang masing-masing berukuran N dan M . Algoritma diimplementasikan dan dijalankan dengan data berukuran sebagai pada tabel berikut serta waktu eksekusinya.

Ukuran N	Ukuran M	Waktu Eksekusi ("~" artinya sekitar)
1.000	1.000	~5 detik
4.000	1.000	~320 detik
2.000	10.000	~533 detik

Berapakah perkiraan waktu eksekusi implementasi algoritma tersebut jika $N = 3.000$ dan $M = 5.000$?

- ☐ ~135 detik
- ☒ ~664 detik 
- ☐ ~67 detik
- ☐ ~830 detik

Your answer is incorrect.

Tujuan: Memahami cara memprediksi running time pada suatu ukuran data N dan M berdasarkan growth rate $O(g(N,M))$ dan running time pada suatu harga N_0 dan M_0 .

Penjelasan Solusi:

Baris kedua tabel dengan yang pertama *growth rate* hanya dipengaruhi oleh N sebanyak 4 kali, waktu eksekusi meningkat dari 5 detik menjadi 320 detik, atau 64 kali (atau 4^3 kali), yaitu secara kubik.

Baris ketiga mengalami kenaikan di kedua parameter. Jika hanya N yang berubah menjadi 2000 (dimana M tetap) maka waktu eksekusinya harusnya sekitar $5 \times 2^3 = 40$ detik. Sementara itu, selain kenaikan N tsb., juga terjadi kenaikan M sebanyak 10 kali, dan waktunya meningkat dari 40 detik tsb menjadi 533 detik, yaitu kenaikan dengan faktor kenaikan $533/40 = 13.4$.

Jika faktor M bersifat linear, perubahan data 10 kali maka faktor kenaikan juga harus 10 kali, sementara jika kuadratis maka faktor kenaikan harus 100 kali. Faktor di antara 10 - 100 ini menunjukkan selain linear ada faktor logaritmis, atau $O(M \log M)$.

Perbedaan harga $1.000 \log_2(1.000) = 9965.78$ menjadi $10.000 \log_2(10.000) = 132877.124$ yaitu faktor $132877.124/9965.78 = 13.334$ membuktikan faktor kenaikan 10 kali dari M menyebabkan kenaikan waktu 13.334 yang bersesuaian dengan sebelumnya.

Jadi kompleksitas waktu algoritma ini $O(N^3 M \log M)$.

Dengan demikian untuk menjawab pertanyaan, faktor N meningkat 3 kali sehingga berkontribusi peningkatan sebanyak 3^3 kali = 27 kali. Sementara, faktor M meningkat 5 kali. Karena memiliki faktor $O(M \log M)$, waktu eksekusi terkait M ini meningkat dengan faktor $(5.000 \log_2(5.000))/(1.000 \log_2(1.000)) = 61438.56 / 9965.78 = 6.165$. Total peningkatan adalah $27 \times 6.165 = 166.45$ kali dari 5 detik menjadi 832.275. Yang paling mendekati adalah 830 detik.

The correct answer is: ~830 detik

Question 7

Correct

Mark 1.00 out of 1.00

Algoritma Binary Search berikut (dari materi kuliah) melakukan proses pencarian data x berdasar data array integer a, sebagai berikut.

```
public static int binarySearch (int[] a, int x ) throws ItemNotFound {
    int low = 0;
    int high = a.length - 1;
    int mid;

    while( low <= high ) {
        mid = (low + high) / 2;    // mendapatkan titik tengah
        if (a[mid].compareTo (x) < 0) {
            low = mid + 1;
        } else if (a[mid].compareTo (x) > 0) {
            high = mid - 1;
        } else {
            return mid;
        }
    }
    throw new ItemNotFound( "BinarySearch fails" );
}
```

Jika method dipanggil dengan array a berukuran 16 yang isinya sebagai berikut:

indeks	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a[indeks]	3	5	12	13	18	29	30	34	43	54	57	60	69	75	83	88

Jika harga x yang dicari adalah 50, berapa kalikan perintah “mid = (low + high) / 2;” akan dilakukan hingga eksekusi selesai?

- ☐ 10 kali.
- ☐ 7 kali.
- ☒ 4 kali. ✓
- ☐ 16 kali.

Your answer is correct.

Tujuan: Memahami proses komputasi dari algoritma Binary Search.

Penjelasan solusi:

low = 0, high = 15, mid = 7 (1 kali)

low = 8, high = 15, mid = 11 (2 kali)

low = 8, high = 10, mid = 9 (3 kali)

low = 8, high = 8, mid = 8 (4 kali)

low = 9, high = 8 => keluar loop while

The correct answer is: 4 kali.

Question 8

Incorrect

Mark 0.00 out of 1.00

Beberapa algoritma memiliki kompleksitas waktu eksekusi terhadap ukuran data N sebagai dalam tabel berikut.

Nama Algoritma	Kompleksitas Waktu Eksekusi dalam Notasi Big-Oh
Cari1	$O(N^3 \log(N^{10}))$
Cari2	$O(N^2 \log(4^N))$
Cari3	$O(2^{\log(N)} N^2)$

Manakah pernyataan yang **benar** mengenai ketiga algoritma tersebut?

- ☐ Yang paling buruk adalah Cari3 karena memiliki faktor eksponensial.
- ☐ Ada dua saja yang memiliki kompleksitas yang sama sementara lainnya berbeda.
- ☐ Ketiganya memiliki growth-rate yang sama.
- ☒ Ketiganya memiliki kompleksitas yang berbeda-beda. ❌

Your answer is incorrect.

Tujuan: Memahami perbedaan tingkatan pada kelas-kelas notasi Big-O pada masalah satu variabel (konstan, logaritmis, linear, $n \log n$, dst..).

Penjelasan: secara matematis

$$\log(N^{10}) = \log(N) \Rightarrow O(\log(N))$$

sementara

$$\log 4^N = N \log 4 \Rightarrow O(N) \text{ dan}$$

$$2^{\log N} = N \Rightarrow O(N).$$

Jadi,

$$O(N^3 \log(N^{10})) \Rightarrow O(N^3 \log(N)),$$

$$O(N^2 \log 4^N) \Rightarrow O(N^3) \text{ dan}$$

$$O(2^{\log(N)} N^2) \Rightarrow O(N^3).$$

The correct answer is: Ada dua saja yang memiliki kompleksitas yang sama sementara lainnya berbeda.

Question 9

Correct

Mark 1.00 out of 1.00

Hal-hal berikut ini adalah tujuan dari adanya pembahasan beberapa versi algoritma solusi untuk masalah **Maximum Contiguous Subsequence Sum** dalam konteks Analisis Algoritma, kecuali satu yang tidak tepat. Manakah itu?

- ☐ Tanpa berpikir dengan matang dan beranalisis lebih dalam seringkali solusi yang diperoleh bersifat *brute-force* (asal jadi dan tidak efisien).
- ☐ Peningkatan efisiensi (penurunan kompleksitas) dari solusi-solusi tersebut berdampak begitu signifikan terhadap performance implementasi.
- ☐ Dengan perubahan sudut pandang pada masalah bisa jadi akan muncul solusi yang jauh lebih efisien, contohnya pada solusi $O(N)$.
- ☒ Mendapatkan algoritma yang lebih efisien cenderung menyulitkan pengembangan aplikasi dan menghasilkan algoritma yang rumit, sehingga boleh dihindari. ✓

Your answer is correct.

Tujuan: Memahami bahwa dengan adanya notasi big-O sebagai ukuran kinerja algoritma, jika memungkinkan dapat memandu mendapatkan algoritma lebih efisien (contoh; pada masalah maximum contiguous subsequence sum).

Pembahasan Solusi: ini sudah jelas tidak perlu dibahas, kan?

The correct answer is: Mendapatkan algoritma yang lebih efisien cenderung menyulitkan pengembangan aplikasi dan menghasilkan algoritma yang rumit, sehingga boleh dihindari.

Question 10

Correct Mark 1.00 out of 1.00

Diberikan method hitung(int N) yang bekerja berdasarkan parameter integer N berikut.

```
void hitung(int N) {  
    for (int i = N; i > 0; i -= 5) {  
        for (int j = N; j > 0; j /= 2) {  
            for (k = 0; k < 1000*N; k += N) {  
                System.out.println(i+ " "+j+ " "+k);  
            }  
        }  
    }  
}
```

Notasi Big-Oh yang mencerminkan kompleksitas waktu eksekusi method di atas adalah:

- ☐ $O(N)$
- ☐ $O(N^2 \log N)$
- ☐ $O(N^2)$
- ☒ $O(N \log N)$ ✓

Your answer is correct.

Tujuan: Memahami cara mendapatkan $O(g(N))$ dari fungsi $f(N)$ secara intuitif (bukan menggunakan Master's theorem yang baru akan diajarkan di DAA).

Penjelasan solusi: loop-for terluar melakukan iterasi decremental yang linear, loop-for kedua bersifat halving berarti logaritmis, dan loop-for terdalam walaupun terkait faktor N tapi incremental dengan harga N juga sehingga banyaknya iterasi menjadi selalu 1000 (konstan). Maka linear - logaritmis - konstan menjadi $O(N \log N)$.

The correct answer is: $O(N \log N)$

◀ 02a. catatan analisis algoritma

Jump to...



03. ADT Collections ▶