MODUL 1

Pendahuluan Struktur Data

Dian Nursantika, S.Kom., M.Cs.
Unggul Utan Sufandi, S.Kom., M.Si.

PENDAHULUAN

Pendahuluan Struktur Data merupakan materi pertama dalam mata kuliah struktur data. Modul ini ditulis untuk menjembatani Anda agar mampu belajar mandiri mengenai struktur data. Pertama-tama akan kita pahami terlebih dahulu mengenai struktur. Apakah struktur tersebut? Tentunya kata struktur bukanlah hal yang baru Anda dengar, karena kata tersebut selalu ada dalam setiap bidang ilmu, seperti misalnya struktur bangunan dalam ilmu sipil, struktur organisasi dalam ilmu politik, struktur organ tubuh dalam ilmu kedokteran, dan struktur lainnya dalam keilmuan lain. Struktur yang akan kita pelajari dalam mata kuliah struktur data adalah struktur dalam data. Apakah yang dimaksud struktur dalam data? Struktur dalam data adalah cara pengolahan data yang terstruktur, yaitu mampu mengolah data sehingga mampu mendukung performa dan organisasi data dalam sebuah aplikasi.

Modul ini tidak hanya membahas struktur data, namun juga berbagai hal yang mendukung struktur data, baik itu algoritma, maupun bahasa pemrograman, dan yang lainnya. Adapun materi yang akan diberikan pada modul 1 ini dibagi dalam tiga kegiatan belajar yaitu:

1. Konsep Struktur Data
2. Matematika dan Struktur Data
3. Bahasa Pemrograman Java

Setelah mempelajari modul ini, diharapkan Anda mampu:

1. Menjelaskan definisi struktur data
2. Menguraikan efisiensi data dalam struktur data
3. Menjelaskan *interface* dalam struktur data
4. Menjelaskan relasi antara eksponensial dengan struktur data
5. Merinci model komputasi struktur data

6. Merinci model kompleksitas komputasi pada struktur data
7. Menjelaskan fungsi *compiler* dalam bahasa pemrograman Java
8. Menjelaskan tipe data pada bahasa pemrograman Java

KEGIATAN BELAJAR 1

Konsep Struktur Data

Pertama kali Anda mendengar kalimat struktur data, apakah yang terlintas dalam benak Anda? Sebagian orang menjawab struktur data seperti struktur–struktur bangunan atau rangka–rangka bangunan seperti kayu atau besi untuk memberikan ruang–ruang pada sebuah bangunan. Adapun jawaban lainnya yaitu seperti struktur organisasi di sebuah organisasi maupun lembaga, seperti ketua, sekretaris, bendahara, anggota dan sebagainya.

A. DEFINISI STRUKTUR DATA

Sebenarnya kita selalu berinteraksi dengan struktur data, misalkan:

1. Membuka *file* dalam komputer. Struktur data pada sistem *file* menggunakan suatu lokasi tertentu untuk menyimpan *file* tersebut dalam tempat penyimpanan (*disk*), sehingga *file* tersebut dapat diambil atau diperlihatkan kepada *user*. Setiap konten pada *file* tersebut dapat disimpan pada berbagai bagian *disk* yang tersedia. Penyimpanan dan pencarian *file* tersebut bukanlah hal yang mudah, karena komputer harus melakukan berbagai proses untuk mendapatkan informasi sesuai dengan keinginan *user*.
2. Melihat kontak pada telepon genggam. Struktur data digunakan untuk melihat nomor telepon yang terdapat pada daftar kontak telepon berdasarkan sebagian data. Sebelum Anda selesai melakukan pengetikan nama yang Anda cari pada telepon, maka telepon Anda akan memberikan beberapa nama yang mirip jika terdapat banyak kemiripan, atau akan langsung memberikan nama yang Anda cari jika tidak terdapat kemiripan dengan nama lainnya dalam telepon Anda.
3. Masuk (*login*) ke dalam media sosial. *Server* akan menggunakan informasi *login* Anda untuk mencari informasi akun Anda. Hal tersebut tidak mudah, karena media sosial yang popular pasti memiliki ratusan, bahkan miliaran *user* yang aktif.
4. Melakukan pencarian di dalam *website*. Mesin pencarian (*search engine*) menggunakan struktur data untuk menemukan *website* berdasarkan kata kunci pencarian yang diberikan *user*. Hal ini pun tidak mudah, karena

lebih dari 8.5 milyar *website* dalam internet dan setiap *website* tersebut memiliki banyak konten yang dapat dijadikan sebagai istilah pencarian tersebut.

5. Telepon layanan darurat (9-1-1). Jaringan layanan darurat akan melihat nomor telepon Anda dalam suatu struktur data yang memetakan nomor telepon Anda dengan alamat Anda, sehingga dapat mengirimkan mobil polisi, ambulan, atau pemadam kebakaran menuju alamat Anda tanpa harus menunggu. Hal ini sangat penting, karena orang yang membuat panggilan tersebut mungkin tidak dapat memberikan alamat yang tepat, sehingga dapat menyebabkan penundaan yang mempertaruhkan antara hidup dan mati.

Berdasarkan beberapa contoh yang telah dipaparkan, maka struktur data sudah menjadi hal yang sangat penting dalam keseharian kita. Dengan demikian kita perlu memahami bagaimana kinerja dari struktur data sehingga mampu menerapkan berbagai konsep dan istilah yang terdapat dalam struktur data. Struktur data adalah cara untuk menyimpan, menyusun, dan mengurutkan data sehingga data tersebut dapat digunakan dengan efisien pada suatu media komputer (Morin, 2012).

Struktur data dapat diterapkan ke dalam bahasa pemrograman untuk mendapatkan pengolahan informasi sesuai dengan tipe dan struktur data yang digunakan. Bahasa pemrograman yang akan digunakan dalam mata kuliah ini adalah bahasa Java, yang merupakan salah satu bahasa pemrograman tingkat tinggi. Pada bab ini akan dibahas mengenai pengenalan Java, tipe data sederhana, deklarasi data, pemetaan ke dalam tempat penyimpanan (*storag*e), organisasi logik dan fisik, serta waktu pelaksanaan program sebagai ukuran data masukan (*input*).

B. PERMASALAHAN, ALGORITMA DAN PEMROGRAMAN

Permasalahan adalah suatu kondisi yang harus diselesaikan atau ditemukan solusinya. Contoh permasalahan adalah keluhan konsumen kepada pihak *onlineshop* mengenai pelayanan yang kurang profesional saat melakukan pemesanan produk yang akan dibeli oleh konsumen (Morin, 2012). Pihak konsumen mengeluh bahwa saat konsumen memesan produk melalui *chat*, respon dari penjual sangat lama. *Chat* tersebut baru dibalas oleh penjual lebih dari 1 jam, padahal pihak konsumen ingin segera mengetahui apakah

produk yang dijual oleh *onlineshop* tersebut tersedia atau tidak. Langkah apa yang harus ditempuh pihak penjual agar pihak konsumen tidak mengeluhkan hal tersebut?

Langkah yang perlu ditempuh oleh pihak penjual adalah mempersiapkan seluruh data yang ada pada *onlineshop* tersebut. Data tersebut dapat berupa kode produk, nama produk, stok produk, dan sebagainya. Selanjutnya melakukan pengecekan secara *real time* pada menu *chat* dengan pihak konsumen, sehingga saat konsumen bertanya apakah produk tersebut tersedia atau tidak, maka pihak penjual akan melakukan pemeriksaan terhadap data yang telah disiapkan, kemudian langsung memberikan jawaban kepada pihak konsumen mengenai ketersediaan produk tersebut.

Hal lain yang dapat dilakukan pihak penjual yaitu menyediakan lebih dari satu orang operator pelayanan pihak konsumen pada bagian *chat*. Dengan demikian, meskipun terdapat banyak pihak konsumen yang secara bersamaan menghubungi pihak penjual, maka hal tersebut dapat ditangani karena jumlah operator *chat* lebih dari satu orang. Langkah-langkah yang dapat dilakukan untuk melayani pihak konsumen melalui menu *chat* dengan pihak konsumen tersebut dapat menjadi sebuah algoritma**.**

Kita lanjutkan pembahasan mengenai keluhan konsumen tersebut. Jika diimplementasikan dalam sebuah program maka pihak penjual, pihak konsumen, operator pelayanan konsumen, dan produk dari *onlineshop* tersebut dapat berupa variabel, *class*, bahkan *method*. Implementasi algoritma dituangkan dalam bahasa pemrograman, misalkan menggunakan bahasa pemrograman Java. Satu algoritma dapat diimplementasikan dengan menggunakan berbagai macam bahasa pemrograman, tergantung dari kemampuan pembuat program atau *programmer*.

C. EFISIENSI DAN INTERFACE STRUKTUR DATA

Pembahasan mengenai efisiensi dan *interface* struktur data menjadi modal awal dalam memahami struktur data. Dalam pembahasan ini mahasiswa akan diperkenalkan dengan konsep kebutuhan struktur data terhadap perkembangan kemampuan komputer serta operasi dasar yang digunakan dalam implementasi struktur data (Morin, 2012).

1. Kebutuhan Efisiensi

Pembahasan selanjutnya mengenai efisiensi, kemampuan komputer menjadi pertimbangan lain dalam menerapkan struktur data selain kemampuan *programming*. Saat kita berpikir bahwa kecepatan prosesor dan ukuran memori meningkat, maka akan muncul pertanyaan; apakah semua permasalahan yang terkait dengan efisiensi dapat diselesaikan dengan kemampuan *hardware* di masa yang akan datang atau dapat diselesaikan melalui penerapan struktur data?

Efisiensi pada komputer sangat penting, sehingga dibutuhkan efisiensi dalam bentuk ruang dan waktu. Efisiensi dapat terwujud dengan pembentukan struktur data yang tepat. Struktur data menjadi solusi yang utama karena seluruh algoritma ataupun pembuatan program tidak akan terlepas dari struktur data.

Sebuah solusi dikatakan efisien jika solusi tersebut dapat memecahkan masalah dengan mengetahui keterbatasan sumber daya yang dimiliki. Misalkan total ruang penyimpanan yang dapat digunakan oleh *user* untuk menyimpan data. Sehingga memungkinkan memori utama untuk melakukan partisi menjadi beberapa *disk* terpisah.

2. Interface

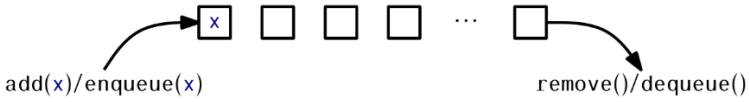
Dalam mempelajari struktur data diperlukan pemahaman mengenai perbedaan antara *interface* dari struktur data dengan implementasinya. *Interface* bukanlah *interface* dalam arti antar muka pengguna, akan tetapi *interface* yang berupa *abstract data type* (tipe data abstrak). Contoh *interface* berupa *queue*, *stack*, dan *deque*. Sedangkan implementasi yaitu representasi struktur data terhadap algoritma dengan menggunakan operasi–operasi yang ada pada *interface*. Misalkan, melakukan *coding* untuk mengimplementasikan *stack* pada bahasa pemrograman tertentu.

Terdapat 4 bagian *interface* (Morin, 2012), yaitu:

a. *Interface Queue, Stack* dan *Deque*
b. *Interface List*
c. *Interface USet* (*Unordered Sets*)
d. *Interface SSet* (*Sorted Sert*)

a. Interface Queue, Stack, dan Deque

Queue. Pernahkah Anda mengantre? Misalkan, mengantre untuk membeli tiket kereta api, maka konsep *queue* dapat Anda simulasikan dengan antrean tiket tersebut. *Queue* (antrean) merupakan *interface* yang merepresentasikan penambahan dan penghapusan elemen dengan menerapkan konsep *First In First Out* (FIFO), yaitu elemen pertama yang masuk ke antrean, akan menjadi elemen pertama yang dihapus pada antrean. Ilustrasi FIFO dapat dilihat pada Gambar 1.1.



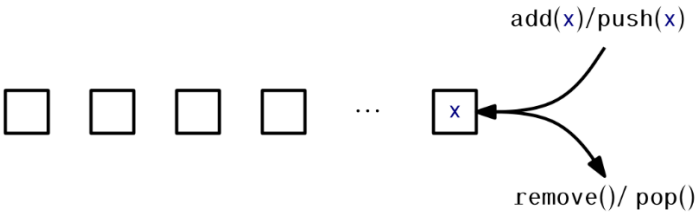
Gambar 1.1
Konsep Proses Queue – FIFO Queue (Morin, 2012)

Operasi yang dimiliki oleh *queue* adalah:

- `add(x)` atau `enqueue(x)`, yaitu operasi menambahkan elemen x pada awal *queue*.
- `remove()` atau `dequeue()`, yaitu operasi menghapus elemen pada akhir *queue*.

Anda dapat menerapkan konsep *queue* pada tumpukan koin yang Anda masukkan ke dalam pipa, pipa tersebut memiliki lubang atas dan bawah. Koin pertama yang dimasukkan ke dalam lubang bagian atas, akan menjadi koin pertama yang akan dikeluarkan dari lubang bagian bawah pipa.

Stack/LIFO Queue. Selanjutnya akan dijelaskan mengenai *stack* (tumpukan), di mana konsep pada *stack* adalah *Last In First Out* (LIFO) seperti yang diilustrasikan pada Gambar 1.2, yang menunjukkan elemen terakhir yang berada pada tumpukan maka akan menjadi elemen pertama yang dihapus jika ada perintah penghapusan. Anda dapat menerapkan konsep *stack* pada tumpukan koin yang Anda masukkan ke dalam pipa, pipa tersebut memiliki lubang bagian atas terbuka dan sementara lubang bagian bawah tertutup. Koin terakhir yang dimasukkan ke dalam lubang bagian atas, akan menjadi koin yang pertama yang akan dikeluarkan dari lubang bagian atas pipa.

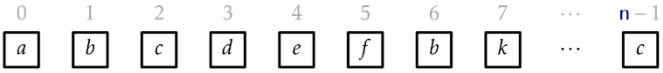


Gambar 1.2
Konsep Proses Stack - LIFO Queue (Morin, 2012)

Operasi yang dimiliki *stack* adalah:

- `push(x)`, yaitu operasi menambahkan elemen x pada awal *queue*.
- `pop()`, yaitu operasi menghapus elemen pada akhir *queue*.

Deque. Penjelasan berikutnya mengenai *deque*, yang memiliki kemampuan untuk menambahkan elemen, baik di depan maupun di belakang antrean. Adapun konsep yang digunakan pada *deque* adalah FIFO dan LIFO, sehingga *deque* memiliki kemampuan *interface queue* maupun *interface stack.*



Gambar 1.3
Konsep Proses Deque (Morin, 2012)

Operasi yang dapat dilakukan oleh *deque* adalah:

- `addFirst(x)`, yaitu operasi penambahan elemen x pada antrean awal.
- `removeFirst()`, yaitu operasi penghapusan elemen pada antrean awal.
- `addLast(x)`, yaitu operasi penambahan elemen x pada antrean akhir.
- `removeLast()`, yaitu operasi penghapusan elemen yang berada pada antrean terakhir.

b. List Interface

Penjelasan berikutnya mengenai *list interface* yang merepresentasikan urutan seperti pada Gambar 1.3. Gambar 1.3 menunjukkan urutan x_0 sampai dengan x_{n-1}. Pemanggilan pada *list* di Gambar 1.3 dengan menggunakan `get(2)` akan mendapat nilai *c*.

Operasi yang dimiliki oleh *list interface* yaitu:

- `size()`: mengembalikan n sesuai dengan panjang list
- `get(i)`: mengembalikan nilai x_i
- `set(i,x)`: himpunan nilai x_i sama dengan x
- `add(i,x)`: menambahkan x pada posisi i, menggantikan x_i, …, x_{n-1};
- `remove(i)`: menghapus nilai x_i, dan menggeser nilai dari x_i, …, x_{n-1};

Berikut adalah contoh penulisan operasi dengan menggunakan *interface deque*:

- `addFirst(x)` ➔ `add(0,x)`
- `removeFirst()` ➔ `remove(0)`
- `addLast()` ➔ `add(size(),x)`
- `removeLast()` ➔ `remove(size()- 1)`

c. Interface USet (Unordered Sets)

USet interface merepresentasikan *unordered set* dari elemen-elemen yang unik. *USet* terdiri dari *n* elemen yang berbeda, tidak ada elemen yang muncul lebih dari sekali, dan elemen-elemen tidak terurut.

Adapun operasi yang terdapat pada *USet* adalah:

- `size()`: mengembalikan jumlah n elemen yang terdapat pada himpunan
- `add(x)`: menambahkan elemen x terhadap himpunan yang belum memiliki elemen
- `remove(x)`: menghapus x dari himpunan
- `find(x)`: menemukan x dari himpunan

d. Interface SSet (Sorted Sets)

SSet interface merepresentasikan *sorted set* dari elemen-elemen. *SSet interface* akan menyortir elemen yang terdapat pada himpunan. Misalkan Anda melakukan proses penyortiran terhadap nilai x dan y, maka dapat dilakukan dengan memanggil sebuah *method* `compare(x,y)`:

$$\text{compare}(x,y)\begin{cases} < 0 & \text{if } x < y \\ > 0 & \text{if } x > y \\ = 0 & \text{id } x = y \end{cases}$$

Jika x lebih kecil dari y, maka hasilnya lebih kecil dari 0, jika x lebih besar dari y, maka hasilnya lebih besar dari 0, jika x sama dengan y, maka hasilnya sama dengan 0.

SSet interface mendukung operasi `size()`, `add(x)`, and `remove(x)` sama seperti pada *USet interface*, perbedaannya terletak pada operasi `find(x)`. Pada *SSet interface*, operasi `find(x)` digunakan untuk menemukan x pada himpunan, sedangkan pada *USet interface*, operasi `find(x)` untuk mencari x pada himpunan yang telah terurut.

LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

1) Jelaskan keterkaitan pencarian kontak telepon pada list kontak telepon dengan struktur data!
2) Jelaskan apa yang dimaksud dengan kebutuhan efisiensi yang terdapat pada struktur data!
3) Jelaskan konsep *stack*, dan berikan contoh dalam kehidupan sehari–hari!

Petunjuk Jawaban Latihan

1) Pelajari bagaimana struktur yang terjadi dalam pencarian kontak telepon tersebut.
2) Pelajari kembali mengenai kebutuhan efisiensi di Kegiatan Belajar 1.
3) Pelajari berbagai contoh kasus yang ada telah diberikan.

RANGKUMAN

Efisiensi merupakan sebuah terobosan dalam komputasi komputer. Efisiensi dapat menghemat penggunaan tempat atau alokasi pada memori. *Interface* yang dibahas merupakan *interface* umum dalam operasi struktur data, yaitu: *queue*, *stack*, *deque*, *list*, *USet* dan *SSet*. Setiap *interface* memiliki karakteristik tertentu sehingga dapat digunakan sesuai dengan kebutuhan program yang akan dibuat.

TES FORMATIF 1

Pilihlah satu jawaban yang paling tepat!

1) Seseorang diberi tugas menduplikasi file yang berekstensi .pdf dari *drive* C menuju *drive* D. Kegiatan tersebut dapat diartikan sebagai ....
   A. algoritma
   B. pemrograman
   C. permasalahan
   D. perumusan

2) Berikut adalah alur kinerja dari pembuatan kopi:
   - Siapkan cangkir kosong, sendok, tempat kopi, tempat gula, termos berisi air putih panas;
   - Masukkan satu sendok kopi dan satu sendok gula ke dalam cangkir;
   - Tuangkan air dalam termos ke dalam cangkir yang telah berisi kopi dan gula;
   - Aduk isi cangkir dengan menggunakan sendok;
   - Secangkir kopi panas siap dihidangkan.

   Serangkaian alur di atas dapat disebut ....
   A. algoritma
   B. pemrograman
   C. permasalahan
   D. perumusan

3) Jika tahapan dari algoritma ingin diimplementasikan ke dalam sebuah komputer, maka hal tersebut disebut dengan ....
   A. algoritma
   B. pemrograman
   C. permasalahan
   D. perumusan

4) Istilah berikut yang bukan termasuk *interface* dalam struktur data, adalah....
   A. *queue*, *stack*, dan *deque*
   B. code
   C. stack
   D. SSet

5) Operasi yang memiliki kemampuan untuk menghapus elemen di akhir *queue* adalah ....
   A. `removeFirst()`
   B. `removeLast()`
   C. `remove()`
   D. `removeOdd()`

6) Operasi untuk menambahkan elemen `x` di awal *stack* adalah ....
   A. `push()`
   B. `add()`
   C. `push(x)`
   D. `add(x)`

7) Operasi untuk menambahkan elemen pada akhir antrean *deque* adalah ....
   A. `addLast(x)`
   B. `addLast()`
   C. `addOdd(x)`
   D. `addOdd()`

8) Penambahan elemen yang dapat dilakukan, baik di depan maupun di belakang antrean, disebut dengan ....
   A. *deque*
   B. *queue*
   C. *uset*
   D. *stack*

9) Konsep dasar dari *stack* adalah ....
   A. FOLY
   B. VIVO
   C. LIFO
   D. FIFO

10) Penerapan efisiensi pada operasi struktur data dapat dibagi dalam dua bagian, yaitu ....
   A. halaman dan tempat
   B. ruang dan waktu
   C. bentuk dan ukuran
   D. panjang dan lebar.

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali
80 - 89% = baik
70 - 79% = cukup
< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 2. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 1, terutama bagian yang belum dikuasai.

KEGIATAN BELAJAR 2

Matematika dan Struktur Data

Pada kegiatan belajar 2 ini akan dibahas pendekatan matematika, yaitu (Morin, 2012):

1. Eksponensial
2. Faktorial
3. Notasi asimtotik
4. Nilai acak dan probabilitas

A. EKSPONENSIAL

Apakah Anda masih ingat pelajaran eksponensial? Eksponensial atau perpangkatan dinyatakan dalam bentuk a^n, di mana a merupakan bilangan pokok atau basis, dan n merupakan bilangan eksponensial, di mana $a, n \in R$. Lebih lanjut, dapat dinyatakan sebagai berikut:

$$a^n = \underbrace{a \times a \times a \times \cdots a}_{\text{sebanyak } n \text{ kali}}$$

Untuk a bilangan real dan $a \neq 0$, n bilangan positif, maka berlaku:

$$a^{-n} = \frac{1}{a^n}$$

Jika $n = 0$, maka $a^0 = 1$.

B. FAKTORIAL

Nilai faktorial digunakan untuk nilai integer positif n, dinotasikan dengan $n!$ seperti terlihat berikut:

$$n! = 1 * 2 * 3 * \ldots * n$$

Perhitungan $n!$ dapat dilakukan dengan pendekatan Stirling's

$$n! = \sqrt{2\mu n}\left(\frac{n}{e}\right)^n e^{\alpha(n)}$$

Di mana

$$\frac{1}{12n+1} < \alpha(n) < \frac{1}{12n}$$

Perhitungan ln(n!) dapat ditulis dengan

$$\ln(n!) = n\ln n - n + \frac{1}{2}\ln(2\pi n) + \alpha(n)$$

Jika dikaitkan dengan fungsi faktorial, maka koefisien binomial untuk non-negatif integer n dan integer $k \in \{0, \ldots, n\}$ dengan notasi $\binom{n}{k}$

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

C. NOTASI ASIMTOTIK

Notasi asimtotik digunakan untuk menghitung waktu tempuh dari operasi yang bervariasi. Adapun notasi yang akan dibahas yaitu *big*-Oh pada sebuah fungsi f(n) dan himpunan dari beberapa fungsi O(f(n)). Notasi *big*-Oh akan diterapkan pada coding untuk menghitung kompleksitas dari algoritma yang terdapat pada coding tersebut. Berikut adalah contoh coding yang dapat dihitung *big*-Oh nya berdasarkan pada n atau banyak iterasi yang dilakukan.

```
void cBigOh(){
  for (int i = 0; i < n; i++)
  a[i] = i;
}
```

- 1 perintah (int i = 0),
- n+1 membandingkan (i < n),
- n perulangan (i++),
- n *array* (a[i]),
- n perintah (a[i] = i)

Running time dari contoh coding di atas adalah:

$$T(n) = a + b(n+1) + cn + dn + en$$

$$T(n) = O(n)$$

Di mana a, b, c, d, dan e adalah nilai konstan yang bergantung pada *machine running* dari sebuah coding, yang merepresentasikan performa waktu dari coding tersebut. Operasi *big*–Oh dapat diterapkan pada berbagai algoritma, terutama untuk mengetahui kompleksitas waktu dari algoritma tersebut.

D. NILAI ACAK DAN PROBABILITAS

Nilai *random* banyak digunakan dalam algoritma untuk mendapatkan nilai acak dari *range* tertentu. Data acak random dengan variabel *X* yang diambil dari semesta pembicara U, ditunjukkan dengan formula:

$$E[X] = \sum_{x \in U} x * \Pr\{X = x\}$$

Dari formula ini terlihat bahwa nilai random *E*[*X*] didapatkan dari penjumlahan terhadap $x * \Pr\{X = x\}$, di mana *x* merupakan anggota dari U. Penjumlahan tersebut diproses dari $x_1, x_2, \ldots, x_n$.

Berikutnya mengenai probabilitas atau peluang. Misalkan saat Anda melempar koin sebanyak k kali, dan Anda ingin menebak berapa kali koin memperlihatkan gambar kepala. Maka jawabannya adalah k/2. Hal tersebut dapat diformulasikan sebagai berikut.

$$E[X] = \sum_{i=0}^{k} i * \Pr\{X = i\}$$
$$= \sum_{i=0}^{k} i * \binom{k}{i} / 2^k$$

Formula tersebut menunjukkan bahwa dilakukan perhitungan peluang sebanyak k kali, sesuai dengan pelemparan koin yang dilakukan.

E. KOMPUTASI STRUKTUR DATA

Proses operasi struktur data tidak terlepas dari proses komputasi. Pada Kegiatan Belajar 2 ini akan diberikan model komputasi struktur data dan istilah terhadap pencapaian proses komputasi pada struktur data. Materi yang akan diberikan mencakup (Morin, 2012):

1. Model Komputasi
2. Kompleksitas Komputasi

1. Model Komputasi

Menganalisa *running times* terhadap operasi struktur data dapat menggunakan model komputasi matematika yaitu w-bit word-RAM (*Random Access Machine*). Nilai random yang digunakan berasal dari *cell* memori (w-*bit word*).

Operasi word-RAM terdiri dari:

- operasi aritmatika (+ , - , * , / , %)
- operasi perbandingn (< , > , = , ≤ , ≥)
- operasi *bitwise boolean* (bitwise-AND, OR dan exclusive-OR)

2. Kompleksitas Komputasi

Setelah Anda mempelajari berbagai hal yang berkaitan dengan struktur data, baik secara konsep maupun algoritma, Anda harus memahami tiga hal penting yang harus selalu diingat dalam mempelajari struktur data yaitu:

- *Correctness*, *interface* struktur data harus dipastikan diimplementasikan sesuai dengan fungsinya.
- *Time complexity* berkaitan dengan *running time* terhadap operasi struktur data, diusahakan memiliki nilai *running time* sekecil mungkin.
- *Space complexity*, struktur data seharusnya menggunakan memori sekecil mungkin.

Definisi *running times* adalah waktu untuk menghitung kinerja dari algoritma yang digunakan. *Running times* terdiri dari tiga jenis, yaitu:

- *Worst-case running times* membuktikan bahwa ada operasi struktur data yang tidak diharapkan atau tidak sesuai dengan target. Hasil *running times* dari operasi struktur data menunjukkan adanya ketidaksesuaian dengan f(n) yang diharapkan. Ketidaksesuaian tersebut memiliki perbedaan yang keluar dari target, sehingga hasil dari *running times* tersebut adalah *worst-case running times*.

- *Amortized running times* merupakan *running times* yang hampir mendekati target, sehingga tidak terlalu melenceng dari target yang diharapkan. Misalkan operasi struktur data yang diharapkan adalah f(n), kemudian hasil dari perhitungan *running times* menunjukkan bahwa *running times* mendekati f(n), sehingga operasi struktur data dinyatakan hampir mendekati target.
- *Expected running times* memiliki perbedaan dengan *running times* yang telah dibahas sebelumnya, karena *expected running times* ini menunjukkan adanya kesesuaian dengan target operasi struktur data yang diharapkan. Misalkan target yang diharapkan struktur data adalah f(n), dan hasil dari operasi struktur data tersebut menunjukkan bahwa f(n) telah tercapai, dengan demikian operasi struktur data tersebut dapat dikatakan *expected running times*.

LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

1) Jelaskan apa yang dimaksud dengan eksponensial!
2) Berikan contoh kasus mengenai probabilitas dalam kehidupan sehari–hari dan jelaskan!
3) Jelaskan dan berikan contoh penerapan operasi *word bitwise boolean* terhadap *programming*!
4) Apa pengaruh kompleksitas waktu terhadap *programming*?
5) Jelaskan perbedaan antara *worst-case* dan *amortized running times*!

Petunjuk Jawaban Latihan

1) Pelajari kembali materi eksponensial dan algoritma.
2) Pelajari kasus yang memiliki istilah peluang.
3) Pelajari operasi dalam berhitung, misalkan penjumlahan, pengurangan, dan lain-lain.
4) Perhatikan bahwa banyaknya coding dapat mempengaruhi kompleksitas waktu.
5) Baca kembali mengenai *running times*.

RANGKUMAN

Struktur data dilahirkan dari konsep matematika. Adapun pembahasan konsep matematika tersebut meliputi; faktorial, notasi asimtotik, nilai acak, dan probabilitas. Eksponensial, faktorial dapat diterapkan pada proses perulangan saat programming. Notasi asimtotik diterapkan saat mencari nilai *big*-Oh untuk menentukan kompleksitas algoritma. Bilangan acak dan probabilitas dapat diterapkan pada proses algoritma tertentu yang memiliki proses random atau probabilitas. Terdapat keterkaitan antara model komputasi dan kompleksitas waktu, karena kompleksitas waktu dapat dilihat dari model komputasi yang diprosesnya. Model komputasi terdiri dari tiga operasi penting, yaitu: operasi aritmatika, perbandingan, dan *word bitwise Boolean*. Kompleksitas komputasi dibagi menjadi dua bagian penting, yaitu kompleksitas dan *running times*. Kompleksitas membahas analisa kesesuaian antara program dengan struktur data yang digunakan dalam pembangunan program tersebut. *Running times* digunakan untuk menganalisa dan memberikan kesimpulan jika terdapat program yang harus dianalisa.

TES FORMATIF 2

Pilihlah satu jawaban yang paling tepat!

1) Terdapat programming seperti berikut:

```
Void printNilaiArray(int array[ ])
{
        cout<<array[0];
}
```

Nilai *big*-Oh dari *programming* di atas adalah ....

A. O( )
B. O(n)
C. O(1)
D. O(2)

2) Terdapat *programming* seperti berikut:

```
Void printNilaiArray(int array[ ])
{
  For (int i=0; i<array.size(); i++;)
  {
       cout<<array[i];
  }
}
```

Nilai *big*-Oh dari *programming* di atas adalah .....

A. O( )
B. O(n)
C. O(1)
D. O(2)

3) Nilai 5 faktorial adalah ....

A. 6
B. 24
C. 120
D. 720

4) Operasi yang termasuk dalam perbandingan adalah ....

A. +
B. AND
C. %
D. >

5) Operasi yang termasuk dalam aritmatika adalah ....

A. +
B. AND
C. %
D. >

6) Operasi yang termasuk dalam operasi *word bitwise Boolean* adalah ....

A. +
B. AND
C. %
D. >

7) Suatu istilah untuk membuat nilai *running time* sekecil mungkin, disebut dengan ....
   A. *correctness*
   B. *time complexctity*
   C. *space compexcity*
   D. *expected poli*

8) Kesesuaian fungsi dan program dalam struktur data perlu dicek kebenarannya. Istilah dari kasus ini adalah ....
   A. *correctness*
   B. *time complexctity*
   C. *space compexcity*
   D. *expected poli*

9) *Running time* yang hampir mendekati target dikenal dengan istilah ....
   A. *corretness*
   B. *expected*
   C. *worst-case*
   D. *amortized*

10) Hasil *running times* yang melebihi target dari pembuatan program disebut....
   A. *correctness*
   B. *expected*
   C. *worst-case*
   D. *amortized*

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali
80 - 89% = baik
70 - 79% = cukup
< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 3. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 2, terutama bagian yang belum dikuasai.

KEGIATAN BELAJAR 3

Bahasa Pemrograman Java

Selanjutnya kita akan mempelajari bahasa pemrograman Java. Apa kaitan antara pemrograman Java dengan struktur data? Pertanyaan tersebut menjadi inti dari Kegiatan Belajar 3 ini, karena diharapkan mahasiswa mampu memahami keterkaitan antara bahasa pemrograman Java dengan struktur data, yaitu mengimplementasikan teori pada struktur data ke dalam bahasa pemrograman. Misalkan dalam mempelajari teori mengenai *stack*, Anda tidak hanya diberikan pemahaman mengenai bagaimana bentuk *stack* secara diagram atau gambar saja, akan tetapi Anda akan diberikan cara bagaimana mengimplementasikannya dengan menggunakan bahasa pemrograman Java.

Pemahaman mengenai struktur data akan lebih mendalam jika Anda sudah dapat mengimplementasikan struktur data ke dalam suatu bahasa pemrograman. Dengan demikian berbagai definisi dan teori yang ada dalam struktur data dapat dijadikan suatu program, yang merupakan hasil implementasi dari bahasa pemrogramannya. Pembahasan struktur data yang disajikan dalam modul ini akan menggunakan bahasa pemrograman Java. Diharapkan dengan menggunakan bahasa pemrograman Java, membuat Anda lebih mudah untuk memahami kinerja dari setiap teori yang ada pada materi struktur data yang diberikan.

A. COMPILER

Pertama-tama akan dibahas mengenai *compiler*. Apakah *compiler* itu? Sebelum dijawab, mari kita lihat pemaparan berikut. Seperti telah kita bahas sebelumnya, bahasa pemrograman Java yang digunakan dalam materi struktur data ini, memiliki peran sebagai alat komunikasi yang akan menghubungkan antara logika manusia dengan program yang akan dibuat. Permasalahannya adalah dimanakah kita dapat menuangkan bahasa pemrograman Java tersebut? Sebuah bahasa pemrograman harus dituangkan dalam tulisan, sehingga dapat menyampaikan apa yang hendak diberikan (Morin, 2012).

Penulisan bahasa pemrograman Java dapat dituangkan dalam *compiler*. *Compiler* adalah sebuah *software* yang dapat mengimplementasikan suatu bahasa pemrograman tertentu. Bahasa pemrograman Java dapat

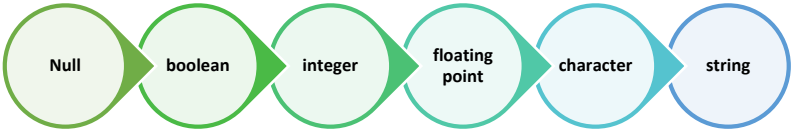
diimplementasikan dalam berbagai macam *compiler*, diantaranya, Netbeans, Eclipse, JCreator, BlueJ, Borland JBuilder, dan sebagainya. Dalam modul struktur data ini kita akan menggunakan *compiler* Netbeans dan *software* JDK, sehingga diharapkan Anda menginstal kedua *software* tersebut.

B. TIPE DATA DALAM BAHASA PEMROGRAMAN JAVA

Selanjutnya kita akan masuk ke pembahasan pertama yaitu tipe data. Terdapat 3 tipe data dalam bahasa pemrograman Java, yaitu (Morin, 2012):

1. Primitif
2. Abstrak
3. Koleksi

1. Tipe Data Primitif



Gambar 1.4
Tipe Data Primitif

Tipe data primitif termasuk dalam tipe data sederhana, yang sering ditemui dalam berbagai pembuatan program dengan menggunakan bahasa pemrograman Java. Tipe data primitif dibagi menjadi 6 jenis, yaitu:

- *Null*
  Dapat dimiliki oleh setiap variabel.
- *Boolean*
  Bernilai *true* atau *false*.
- *Integer*
  Merupakan nilai bilangan bulat seperti 176 atau -52 yang bertipe int 32-bit. Tipe 64-bit adalah *long integer* yang penulisan nilainya diakhiri dengan menambahkan huruf "L" atau "I", misalkan 176L atau -52I.
- *Floating point*
  Bernilai *double*, misalkan 3.1415 dan 135.23 (tanda titik di sini menunjukkan tanda koma dalam Bahasa Indonesia), jika ingin menandai

bahwa nilai tersebut bernilai *float* maka dapat ditambahkan huruf "F" atau "f". Nilai *floating point* juga dapat ditulis dengan menambahkan asumsi berbasis 10, seperti 3.14E2 atau 19e10.

- *Character*

Seluruh nilai *unicode* alphabet merupakan *character constant*. Secara umum karakter dapat didefinisikan sebagai simbol individu yang bersifat satu, misalkan 'a' atau '?'. Java memiliki *character constant* khusus seperti berikut:

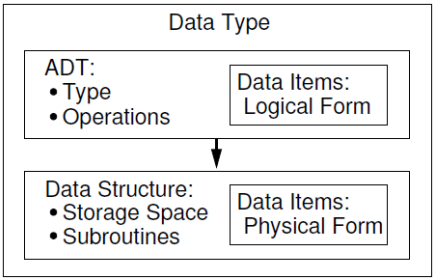
- '\n' (*newline*)
- '\b' (*backspace*)
- '\f' (*form feed*)
- '\' (*single quote*)
- '\t' (*tab*)
- '\r' (*return*)
- '\\' (*backslash*)
- '\' (*double quote*)

- *String*

  Merupakan rangkaian karakter yang berada dalam kutip dua, contohnya: "*dog cannot climb trees*".

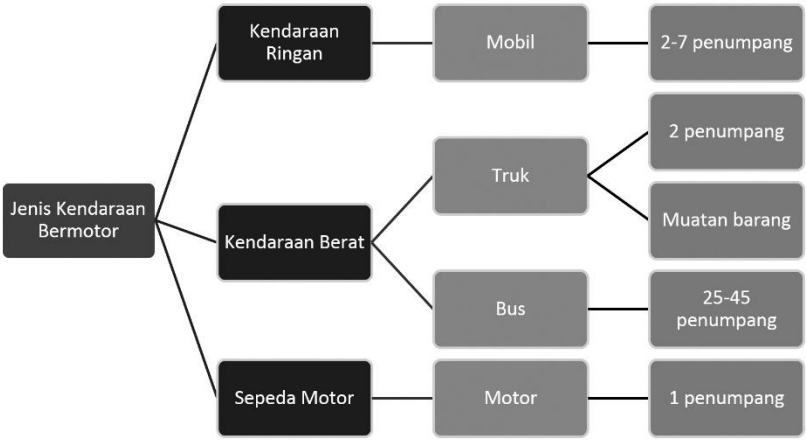
2. Tipe Data Abstrak

Selanjutnya adalah tipe data abstrak atau lebih dikenal dengan istilah *Abstract Data Types* (ADT). ADT bergantung pada tipe data dan serangkaian operasi yang dilakukan sesuai dengan input dan output pada *software* tersebut. Relasi antara ADT dan struktur data dapat dilihat pada Gambar 1.5 berikut.



Gambar 1.5
Abstract Data Types

ADT mendefinisikan bentuk logis dari tipe data, sedangkan struktur data mengimplementasikan bentuk fisik dari tipe data. Implementasi ADT dalam bahasa pemrograman Java dapat diterapkan pada pembuatan *class*. *Class* adalah sebuah tempat yang mampu menampung berbagai anak *class* dan berbagai *object* yang memiliki *method* tertentu yang disesuaikan dengan anak *class* dan *object* yang dimiliki class tersebut. Gambar 1.6 memberikan ilustrasi mengenai *class*, anak *class*, objek, dan *method*.



Gambar 1.6
Ilustrasi Class

Berdasarkan Gambar 1.6, *class* utama adalah "Jenis Kendaraan Bermotor". *Class* tersebut memiliki anak *class* yang terdiri dari 3 yaitu:

a. Kendaraan Ringan
   Mobil sebagai objek dalam Kendaraan Ringan, memiliki *method* atau kemampuan menampung 2-7 penumpang.
b. Kendaraan Berat
   Truk dan Bus sebagai objek dalam Kendaran Berat. Truk memiliki 2 *method* atau kemampuan, yaitu menampung 2 penumpang dan memuat barang. Bus memiliki kemampuan menampung 25-45 penumpang.
c. Sepeda Motor
   Motor sebagai objek dalam Sepeda Motor, memiliki *method* untuk menampung 1 orang penumpang saja.

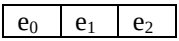
Demikianlah pembahasan mengenai Tipe Data Abstrak. Diharapkan Anda mampu memahami berbagai hal yang berkaitan dengan tipe data abstrak, seperti adanya *class*, anak *class*, objek, dan *method*.

3. Tipe Data Koleksi

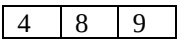
Berikutnya adalah tipe data koleksi yang merupakan tipe data dengan jumlah data yang dinamis. Misalkan tipe datanya adalah *integer*, yang terdiri dari banyak variabel. Kemudian variabel-variabel tersebut disimpan dalam suatu tempat. Dalam bahasa pemrograman Java, tempat tersebut disebut dengan *array* atau *list*.

- **Array**

  Array dalam bahasa pemrograman adalah sebuah variabel yang memiliki tipe data sejenis yang berderet sehingga memiliki alamat memori yang bersebelahan atau bersambung. Pada Gambar 1.7 terdapat sebuah ilustrasi mengenai elemen dan indeks pada *array*, yang berukuran 1 baris dan 3 kolom. Terdapat 3 elemen, yaitu elemen e_0, e_1, dan e_2, sedangkan jumlah indeksnya sebanyak 3. Adapun indeks yang aktif yaitu indeks ke-0, ke-1, dan ke-2.



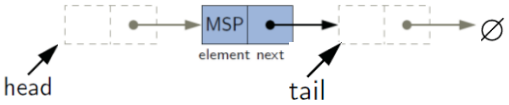
Gambar 1.7
Ilustrasi Array



Gambar 1.8
Ilustrasi Isi Array

- **List**

  Tipe data koleksi berikutnya yang akan dibahas adalah *list*, yang merupakan untaian atau rangkaian dari daftar atau *list* dari data yang dibaca komputer. Implementasi *list* dalam bahasa pemrograman Java berupa *linked list*. Ilustrasi mengenai *list* dapat dilihat pada Gambar 1.9 yang terdiri dari *node*, elemen, pointer. Terdapat 3 *node* yaitu *node head, node next,* dan *node tail*. Setiap *node* berisi 2 kolom, yaitu kolom data atau elemen, dan kolom penunjuk arah yang memiliki tanda panah menuju *node* berikutnya.



Gambar 1.9
Node yang Tersusun dan Menjadi Rangkaian Linked List
(Goodrich, Tamassia, & Goldwasser, 2014)

Demikianlah pembelajaran pada Kegiatan Belajar 3 di modul pertama ini. Semoga Anda mampu memahami secara garis besar berbagai hal yang berkaitan dengan struktur data, bahasa pemrograman Java, dan tipe data pada bahasa pemrograman Java.

LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

1) Jelaskan yang dimaksud dengan tipe data!
2) Jelaskan *compiler* dan berikan contohnya!
3) Berikan penjelasan dalam bentuk gambar dan narasi mengenai hubungan antara struktur data dengan bahasa pemrograman!

Petunjuk Jawaban Latihan

1) Tipe data yang ada dalam bahasa pemrograman Java.
2) *Compiler* secara umum dan yang mengimplementsikan bahasa pemrograman Java.
3) Implementasi struktur data dalam bahasa pemrogaman java.

RANGKUMAN

Dalam bahasa pemrograman Java terdapat 3 jenis tipe data, yaitu tipe data primitif, abstrak, dan koleksi. Ketiga tipe data ini memiliki kemampuan untuk menyimpan data. Tipe data primitif dapat menyimpan data sederhana seperti nilai yang kosong atau *null*, *integer*, *boolean*, *floating point*, *character*, *string*. Tipe data abstrak dapat menyimpan data berupa *class*, objek, *method*. Tipe data koleksi yaitu *array* dan *list*. *Compiler* yang dapat digunakan untuk bahasa pemrograman Java adalah Netbeans, Eclipse. JCreator, BlueJ, Borland JBuilder, dan sebagainya.

TES FORMATIF 3

Pilihlah satu jawaban yang paling tepat!

1) Bahasa pemrograman yang digunakan pada materi struktur data pada modul ini adalah ....
   A. C
   B. C++
   C. Java
   D. Phyton

2) Jenis *compiler* yang digunakan pada materi struktur data pada modul ini adalah ....
   A. Netbeans
   B. Eclipse
   C. Jcreator
   D. BlueJ

3) Data dengan nilai "7,50" termasuk tipe data ....
   A. *integer*
   B. *string*
   C. *floating point*
   D. *character*

4) Data dengan nilai "8" termasuk tipe data ....
   A. *integer*
   B. *string*
   C. *floating point*
   D. *character*

5) Perintah untuk memindahkan kursor ke baris baru adalah ....
   A. '\b'
   B. '\f'
   C. '\n'
   D. '\r'

6) Teks "materi struktur data", termasuk dalam tipe data ....
   A. *integer*
   B. *string*
   C. *floating point*
   D. *character*

7) Jika Anda pahami kembali mengenai *class*, maka Linux dapat dikategorikan ke dalam *class* ....
   A. struktur data
   B. sistem operasi
   C. sistem informasi
   D. struktur komputer

8) Jumlah elemen yang dimiliki array pada gambar berikut adalah ....

Irlandia	Swedia	Norwegia	Islandia	Denmark	Finlandia

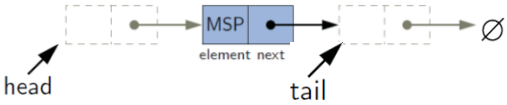
   A. 3
   B. 4
   C. 5
   D. 6

9) Berdasarkan gambar *array* berikut, isi dari *array* adalah ....

satu	dua	tiga

A. satu, dua, tiga
B. satu, tiga
C. dua, satu
D. tiga, dua

10) Berdasarkan gambar berikut, elemen pada *list* adalah ....



A. *head*
B. *tail*
C. MSP
D. Ø

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 3 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 3.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali
80 - 89% = baik
70 - 79% = cukup
< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan modul selanjutnya. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 3, terutama bagian yang belum dikuasai.

Kunci Jawaban Tes Formatif

Tes Formatif 1

1) C
2) A
3) B
4) B
5) C
6) D
7) A
8) A
9) D
10) B

Tes Formatif 2

1) C
2) B
3) C
4) D
5) A
6) B
7) B
8) A
9) D
10) B

Tes Formatif 3

1) C
2) A
3) C
4) A
5) C
6) B
7) B
8) D
9) A
10) C

Daftar Pustaka

Goodrich, M. T., Tamassia, R & Goldwasser, M. H. (2014). *Data Structures & Algorithms*, sixth edition.

Morin, P. (2012). *Open Data Structures* (*in Java*), edition 0.1E. Retrieved from http://opendatastructures.org/ods-java/