

Struktur Data



Definisi

- Cara penyimpanan data pada komputer sehingga data tersebut dapat digunakan secara efisien
- Pemilihan struktur data yang tepat umumnya memungkinkan diterapkannya algoritma yang lebih efisien

Strategi Pemilihan

- Salah satunya adalah dengan mempertimbangkan kompleksitas waktu operasi-operasi dasarnya

- **function** `find(T: tipe_data) → element`
 - mencari nilai T pada struktur data yang bersangkutan
- **function** `find_extreme() → element`
 - mencari nilai ekstrem (maksimum/minimum) pada struktur data
- **procedure** `add(T: tipe_data)`
 - menambah sebuah elemen ke struktur data
- **procedure** `remove(E: element)`
 - menghapus sebuah elemen sembarang dari struktur data
- **procedure** `remove_extreme()`
 - menghapus elemen bernilai ekstrem dari struktur data

3

Struktur Data Dasar

= *Fundamental/Foundational Data Structure*

- *Building blocks* struktur data yang lebih *advanced*
- Umumnya merupakan koleksi/kumpulan tipe data primitif (integer, boolean, char, real)
- Beberapa struktur perlu dipahami dengan baik untuk keperluan pemrograman Olimpiade Informatika (OI)
 - List
 - Set

4

List

= Senarai

- Kumpulan data yang memiliki ciri-ciri khusus
 - *Ordered* (urutan data dari masukan dipertahankan – perhatikan bahwa *ordered* ≠ *sorted*)
 - Elemennya tidak harus unik (boleh ada elemen-elemen bernilai sama)
- Ada 2 model implementasi yang umum dipakai
 - Array
 - Linked List
- Pada bahasa-bahasa pemrograman prosedural, tipe data elemennya biasanya harus seragam

5

Array

= Larik

indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
data	c	h	a	r	a	c	t	e	r	s	a	r	r	a	y

`array [1..15] of char`

- Definisi matematis: rangkaian (berhingga) elemen-elemen yang masing-masing dapat diakses secara langsung
- Kumpulan data direpresentasi secara kontigu (bersambungan, antareleman tidak bercelah)

6

Array

- Umumnya, tipe dan ukuran elemen seragam
- Elemen diacu berdasarkan posisinya dalam rangkaian
- Representasi posisi berupa indeks (disebut juga *subscript*), yaitu anggota dari himpunan nilai ordinal (umumnya integer)
- Sangat efisien dalam hal *random access* (pengaksesan acak)
 - Kompleksitas waktu pengaksesan elemen ke-*i* konstan, $O(1)$

7

Array

Karakteristik Operasi Dasar

- `find(T) : $O(n)$`
 - harus menelusuri seluruh elemen array
- `find_extreme() : $O(n)$`
 - juga harus menelusuri seluruh elemen array
- `add(T) : $O(1)$`
 - tambahkan pada akhir array
- `remove(E) : $O(1)$`
 - penghapusan elemen menghasilkan lubang
 - pindahkan elemen terakhir untuk menutup lubang
- `remove_extreme() : $O(n)$`
 - `find_extreme() + remove(E)`

8

Array

Dalam Bahasa Pascal

■ Deklarasi

- **var**
A: **array** [*rentang_ordinal*] **of** *tipe_data*;
- **type**
tabel = **array** [*rentang_ordinal*] **of** *tipe_data*;
var
A: tabel;
- **contoh**
 - **var** X: **array** [-10..10] **of** integer;
 - **type** tab_char = **array** [char] **of** real;
var Y: tab_char;

9

Array

Dalam Bahasa Pascal

■ Cara akses elemen

- ke-*i*: X[i]
 - contoh pemakaian
 - WriteLn(X[i]);
 - ReadLn(Y['A']);

- Ukuran bersifat statis (tetap, tidak berubah-ubah), yaitu sesuai ukuran yang dispesifikasikan pada bagian *rentang_ordinal* pada bagian deklarasi

10

Array

Contoh Soal

- Sebagian besar soal OI, di antaranya soal-soal yang termasuk kelas soal *Dynamic Programming*

11

Sorted Array

= Larik Terurut

- Salah satu variasi dari array
- Elemennya terurut berdasarkan kriteria tertentu
 - *ascending* (menaik) : $T_i < T_{i+1}$
 - *descending* (menurun) : $T_i > T_{i+1}$
- Tiap kali operasi dilakukan, keterurutan dipertahankan

12

Sorted Array

Karakteristik Operasi Dasar

- $\text{find}(T) : O(\lg n)$
 - dengan menggunakan *binary search*
- $\text{find_extreme}() : O(1)$
 - elemen pertama dan terakhir merupakan nilai ekstrem
- $\text{add}(T) : O(\lg n + n) \approx O(n)$
 - $\text{find}(T)$ untuk menemukan tempat yang tepat, kemudian geser elemen array satu-per-satu untuk menyediakan tempat bagi elemen yang disisipkan ($O(n)$)
- $\text{remove}(E) : O(n)$
 - penghapusan elemen menghasilkan lubang
 - geser elemen untuk menutup lubang
- $\text{remove_extreme}() : O(1)$ atau $O(n)$
 - bergantung pada jenis keterurutan dan ekstrem yang dihapus: jika penghapusan terjadi pada elemen pertama, lakukan penggeseran ($O(n)$)

13

Sorted Array

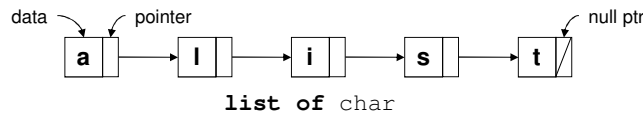
Contoh Soal

- Soal-soal *Greedy Algorithm*
 - Catatan: seringkali keterurutan tiap kali add tidak perlu dijaga, cukup di-sort setelah semua proses add selesai – lebih efisien!

14

Linked List

= Senarai Berantai



- Definisi matematis: rangkaian (berhingga) elemen-elemen yang masing-masing dapat diakses secara serial
- Kumpulan data direpresentasi dengan bantuan *pointer* (penunjuk) yang berfungsi menunjukkan lokasi elemen berikutnya
 - Akibat: tidak perlu melakukan pemesanan memori untuk keseluruhan struktur data sewaktu deklarasi – struktur data dinamis: ukuran sesuai kebutuhan

15

Linked List

- Bersifat serial: elemen tidak dapat langsung diacu, untuk mengakses sebuah elemen perlu dilakukan *traversal* (penelusuran)
 - Kompleksitas waktu *random access* linier, $O(n)$
- Merupakan tipe data self-referential/rekursif: punya pointer ke data lain yang bertipe sama
- Pada praktiknya, linked list sering diacu sebagai list saja
- Elemen linked list disebut juga *node* (simpul)

16

Linked List

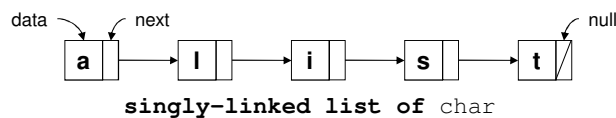
Varian

- Ada beberapa varian dari linked list
 - Linearly-linked list
 - Singly-linked list
 - Doubly-linked list
 - Circularly-linked list
 - Singly-circularly-linked list
 - Doubly-circularly-linked list
 - Linked list dengan sentinel

17

Singly-Linked List

= Senarai Berantai-Tunggal



- Varian linked list paling sederhana
- Tiap node hanya memiliki 1 pointer yang menunjuk ke node berikutnya (next)
 - Pointer node terakhir menunjuk ke nilai null (representasi list kosong)
 - Sewaktu traversal, tidak bisa mundur ke node sebelumnya karena hanya punya next

18

Singly-Linked List

Dalam Bahasa Pascal

- [TBD] deklarasi, implementasi operasi-operasi dasar (e.g. insert, delete)

19

Singly-Linked List

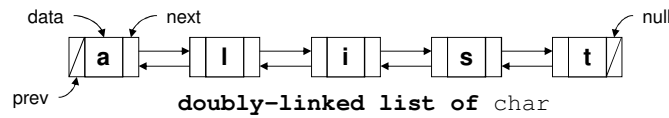
Contoh Soal

- [TBD]

20

Doubly-Linked List

= Senarai Berantai-Ganda



- Lebih rumit daripada singly-linked list
- Tiap node memiliki 2 pointer
 - 1 pointer menunjuk ke node sebelumnya (prev)
 - 1 pointer menunjuk ke node berikutnya (next)
 - Prev dari elemen pertama dan next dari elemen terakhir adalah null
 - Sewaktu traversal, bisa mundur karena punya prev

21

Doubly-Linked List

Dalam Bahasa Pascal

- [TBD]

22

Doubly-Linked List

Contoh Soal

- [TBD]


23

Circularly-Linked List

= Senarai Berantai-Sirkuler

- Singly-circularly-linked list: next dari elemen terakhir (last) menunjuk ke elemen pertama (first)
- Doubly-circularly-linked list: next dari last menunjuk ke first, prev dari first menunjuk ke last
- Kegunaan utama: untuk struktur yang secara natural bersifat sirkuler, misalnya buffer (tempat penyimpanan sementara)
 - Contoh aplikasi: membuat daftar 5 program yang terakhir diakses – circularly-linked list dengan 5 node
 - Dalam kondisi seluruh node terisi, akan terjadi penimpaan data terlama dengan data terbaru

24




Singly-Circularly-Linked List

Dalam Bahasa Pascal

- [TBD]

25




Singly-Circularly-Linked List

Contoh Soal

- [TBD]

26




Doubly-Circularly-Linked List

Dalam Bahasa Pascal

- [TBD]

27




Doubly-Circularly-Linked List

Contoh Soal

- [TBD]

28




Set

= Himpunan

- [TBD]

29



Set

Dalam Bahasa Pascal

- [TBD]

30

Set

Contoh Soal

- [TBD]

31

Tips & Tricks

- Umumnya, struktur data **array** sudah mencukupi untuk menyelesaikan soal-soal IOI
- Gunakan linked list hanya dalam kondisi khusus
 - Deklarasi array tidak mungkin (jumlah data terlalu besar)
 - Gunakan linked list dengan representasi pointer
 - Soal tersebut jauh lebih efisien diselesaikan dengan linked list
 - Misalnya, yang melibatkan banyak operasi insert dan/atau delete

32

Referensi

- Artikel dari [Wikipedia](#):
 - [Data Structure](#), [List of Data Structure](#)
 - [List: Array](#), [Linked List](#)
- Wirth, Niklaus. *Algorithms + Data Structures = Programs*. Eastern Economy Edition, Second Printing. Prentice-Hall of India, SEP 1991
- Max. [Advanced Data Structures](#). [HKOI](#), 5 FEB 2005.
 - Revisi: Kayman. [Advanced Data Structure](#). [HKOI](#), 21 JAN 2006.
- Liem, Inggriani. *Struktur Data Lanjut*. [TOKI-Biro ITB](#), 20 JUN 2005.
- Chionh, Eng Wee. [Data Structures](#). [NUS](#), 21 MAY 2005

Tata Bahasa

- [Panduan Pembakuan Istilah](#). [yLSM](#), 19 SEP 2001