

SDA Cheatsheet

1. BST

InOrder = left, tengah, right

PreOrder = tengah, left, right

Post Order = left, right, tengah

Remove :

- Leaf langsung hapus
- Satu child langsung ganti
- Dua anak successor inorder (terkecil di kanan)
- Dua anak predecessor inorder (terbesar di kiri)

Running time all query =>

- $O(\log n)$ average case,
- $O(n)$ worst case

2. AVL Tree

Balanced BST

Selisih subtree kiri dan kanan maks 1

Pindah kiri : kalau ada anak kiri dijadiin anak

kanan node kirinya, node skrg jadi parent

Pindah kanan : kebalikan kiri

Lengkapin kapan pakai single, double lr

3. Binary-Heap

Root = 0

Index kiri $2i + 1$

Index kanan $= 2i + 2$

Parent $= \text{floor}((i-1)/2)$

Insert -> masukkan ke pos paling akhir,
percolate up

Delete max/min -> ganti root dengan index
terakhir, lalu percolate down

Heapify -> percolate down pada semua
node kecuali leafes

Running time =>

- Insertion $O(\log n)$
- Find min/max $O(1)$
- Delete min/max $O(\log n)$

4. Graph

Simple path => tidak ada dua edge yg sama

DAG => tidak ada cycle

Connected graph => setiap pasang node pasti terhubung

Edge list =>

- Mudah diimplementasikan
- Tidak efisien bila diketahui node

Topo-sort =>

- mengurutkan simpul-simpul dari sebuah directed acyclic graph (DAG) sedemikian hingga jika ada lintasan di dalam graf dari u ke v, maka u akan muncul sebelum v di dalam urutan tersebut
- DAG minimal 1 topo-sort
- Ada cycle, tidak ada topo-sort

MST Prim => berdasarkan nodes yg sudah masuk

MST Kruskal => berdasarkan edge list

5. Hash Tables

Penyimpanan data menggunakan pemetaan fungsi

Ukuran hash biasa lebih besar dari jumlah data

Load factor = $n(\text{size data}) / m(\text{size hash})$

Collision resolution :

- Closed Hashing

- Linear Probing

Index = $H + i$, $i (1, 2, \dots, n)$

Masalah : primary clustering (sel pengganti sama)

Kompleksitas => load factor

Load factor > 0,5 tidak disarankan

Disarankan ukuran table 2 x data

- Quadratic Probing

Index = $H + i^2$

Bermasalah jika table terisi > setengah

Ukuran hash table jangan bil.

Kuadrat, lebih baik prima

Masalah : second clustering

- Double hash

Index = $H + i * H2$

$H2 = y \neq 0$

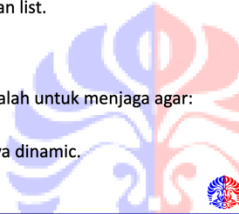
Ukuran hashtable prime

- Open Hashing

- Linked list untuk setiap index

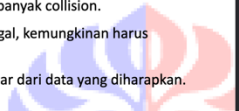
Analisa Open Hash

- Secara umum panjang dari linked list yang dihasilkan sejalan dengan nilai λ .
- Kompleksitas insertion bergantung pada fungsi hash dan insertion pada linked-list.
- Untuk pencarian, kompleksitasnya adalah waktu konstan dalam mengevaluasi fungsi hash + pembacaan list.
- *Worst case* $O(n)$ untuk pencarian.
- *Average case* bergantung pada λ .
- Aturan umum untuk *open hashing* adalah untuk menjaga agar: $\lambda \approx 1$.
- Digunakan untuk data yang ukuran-nya dinamic.



Isu-isu lain

- Hal-hal lain yang umum dan perlu diperhatikan pada metode *closed hashing resolutions*:
 - Proses menghapus agak membingungkan karena tidak benar-benar dihapus.
 - Secara umum lebih sederhana dari pada open hashing.
 - Bagus bila diperkirakan tidak akan terjadi banyak collision.
 - Jika pencarian berdasarkan fungsi hash gagal, kemungkinan harus mencari/membaca seluruh tabel.
 - Menggunakan ukuran table yang lebih besar dari data yang diharapkan.



Rangkuman

- Hash tables: array
- Hash function: Fungsi yang memetakan keys menjadi bilangan [0 \Rightarrow ukuran dari hash table)
- Collision resolution
 - Open hashing
 - Separate chaining
 - Closed hashing (Open addressing)
 - Linear probing
 - Quadratic probing
 - Double hashing
 - Primary Clustering, Secondary Clustering



Rangkuman

- Advantage
 - running time
 - $O(1) + O(\text{collision resolution})$
 - Cocok untuk merepresentasikan data dengan frekuensi insert, delete dan search yang tinggi.
- Disadvantage
 - Sulit (tidak efficient) untuk mencetak seluruh elemen pada hash table
 - tidak efficient untuk mencari elemen minimum or maximum
 - tidak bisa di expand (untuk closed hash/open addressing)
 - ada pemborosan memory/space

