**SDA Cheatsheet**

1. **BST**

InOrder = left, tengah, right

PreOrder = tengah, left, right

Post Order = left, right, tengah

Remove :

* Leaf langsung hapus
* Satu child lagsung 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

1. **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

1. **Binary-Heap**

Root = 0

Index kiri 2i + 1

Index kanan = 2i + 2

Parent = 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)

1. **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

1. **Hash Tables**

Penyimpanan data menggunakan pemetaan fungsi

Ukuran hash biasa lebih besar dari jumlah data

Load factor = n(size data) / m (size hash)

Collision resolution :

* Closed Hashing
* Linear Probing

Index = H + I, i (1,2,…,n)

Masalah : primary clustering (sel penggati sama)

Kompleksitas => load factor

Load factor > 0,5 tidak disarankan

Disarankan ukuran table 2 x data

* Quadratic Probing

Index = H + i2

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 != 0

Ukuran hashtable prime

* Open Hashing
* Linked list untuk setiap index

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generatedA picture containing text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated