CS2040 Finals Cheatsheet v1.0 (2019-05-01)
by Julius Putra Tanu Setiaji, page 1 of 1

1 Summary

	insert	delete	search
Array	$O(n)$	$O(n)$	$O(n)$
Linked List	$O(1)$	$O(1)$	$O(n)$
BST	$O(n)$ worst, $O(\log n)$ avg	$O(n)$ worst, $O(\log n)$ avg	$O(n)$ worst, $O(\log n)$ avg
AVL Tree	$O(\log n)$	$O(\log n)$	$O(\log n)$
Hash	$O(n)$ worst, $O(1)$ avg	$O(n)$ worst, $O(1)$ avg	$O(n)$ worst, $O(1)$ avg

2 List

2.1 Array

- Best for fixed-size lists.
- Random Access: $O(1)$
- Insertion: $O(n)$
- Random Deletion: $O(n)$, deletion from back: $O(1)$

2.2 Linked List

- Random access: $O(n)$
- Random insertion & deletion: $O(n)$
- Insertion & deletion to head or tail: $O(1)$
- Use `ListIterator` to iterate through the list.
- Variations:
  - **Tailed** linked list: with a pointer to tail
  - **Doubly** linked list: each node with `prev` and `next`
  - **Circular** linked list: pointer on one of the nodes to a prev node

3 Stacks and Queues

3.1 Stack

- LIFO with 2 operations: push and pop
- Implementation using array: one pointer for the top element.

Uses

- **Bracket matching** – for every '(', push; every ')', pop. If doesn't match, underflow, stack not empty, then error.
- **Converting infix to postfix** – print out operands. When encountering ')', pop and print until encountering '(' or stack is empty. Else, push any other operator.

4 Queue

- FIFO with 2 operations: enqueue and dequeue
- Implementation with array: two pointers for front (pointing to front of the queue) and back (pointing to where new element should be inserted).
- Circular array:
  - `front = (front + 1) % maxsize; back = (back + 1) % maxsize`
  - To distinguish full/empty state: either (1) maintain queue size of full status, or (2) leave a gap, so full state `(((B+1) % maxsize) == F)`

Uses

- Print queues
- BFS
- Checking palindromes – a **Stack** reverses order, a **Queue** preserves order

5 Recursion

- Can be visualised using a stack
- **Divide** into sub-problems of the same type and **Conquer** the subproblems with recursion
- Recipe for recursion:
  1. General (Recursive) Case
  2. Base Case
  3. Ensure base case is reached (no infinite recursion)
- **Backtracking**: allows us to exhaustively search all possible results in a systematic manner.

6 Complexity Analysis of Algo

CS1101S!

7 Sorting

7.1 Summary

Sorting	Worst Case	Best Case	In-place?	Stable?
Selection	$O(n^2)$	$O(n^2)$	Yes	NO
Insertion	$O(n^2)$	$O(n)$	Yes	Yes
Bubble	$O(n^2)$	$O(n^2)$	Yes	Yes
Bubble (with flag)	$O(n^2)$	$O(n)$	Yes	Yes
Merge	$O(n \log n)$	$O(n)$	NO	Yes
Radix	$O(n)$	$O(n)$	NO	Yes
Quick	$O(n^2)$	$O(n \log n)$	Yes	NO

8 Trees

8.1 Terminology

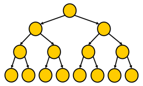
- **Node / Vertex**
- **Edge**
- **Parent**
- **Children**
- **Sibling**
- **Ancestor**
- **Descendant**
- **Root** (has no parent)
- **Internal nodes** (has 1/more children)
- **Leaves** (has no children)
- **Level** of a node (no of nodes of the path from root to node) (1-indexed)
- **Height** of a tree (max level of nodes)
- **Size** of a tree (no of nodes in the tree)

8.2 Binary Tree

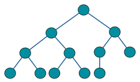
Each node has at most 2 ordered children.

8.2.1 Types

Full BT



Complete BT



- **Full** BT = every node has either 0 or 2 children
- **Complete** BT = Every level except the last is completely filled, and all nodes in the last level are as far left as possible

8.2.2 Properties

- **Full BT**: no of nodes = $N = 2^h - 1$, height = $\log(N + 1)$
- **Complete BT**: $\max(N) = 2^{h-1}$, $\min(N) = 2^h - 1$

8.2.3 BT Traversal

- Post-order
- Pre-order
- In-order
- Level-order: Traverse the tree level by level from left to right. (BFS)

8.3 BST

Property: All keys smaller then root in left subtree, larger in right subtree.

Case of deleting node with 2 children: move smallest node in right subtree to the deleted node.