DSA..  
  
notes continues. Check other

Lnked lists

Head-first node-always with a reference

Tail- last node-always with a reference for quick additions/removals

Pointers – references, 64bit – 8 bytes, 32bit – 4 bytes

Nodes are object containing data and pointers,

Singly and doubly linked list

Single - refers the next node,pro = less memory, simple,con= no prev access

Double – refers both next and previous node, pro = prev , remove in constant time, con= 2x memory

Linked list operation-

(add & remove)

Singly linked  
reference head,   
traverse to the node prev to operation node,  
 create new node,  
 reference the new node to the next node,  
 then change reference the prev node to the new node,   
flat.

Reference head pointer 1,  
reference heads next node pointer2,  
traverse pointer 2 to the node to be removed,  
traverse both nodes simultaneous where pointer 1 is traversed to the prev node to the node to be removed,  
refer a new pointer to the node to be removed,  
traverse pointer2 to the next node,  
change reference of the node of pointer1 from node to be removed to the node of pointer2,  
the node to be removed pointed by pointerT can be removed and cleaned to avoid memory leaks,  
flat.

Doubly linked  
Reference head,  
 traverse to the node prev to operation node,  
 create new node,  
 reference the new node to the next node and also refer it to the prev node,  
 now refer the next node to refer new node,  
 and refer prev node to refer new node,  
flat.

Refer head,  
traverse to node to be removed,  
change prev node (nodeP) pointer from node to be removed (nodeR) to the next node to the nodeR (nodeN),  
and change the pointer of nodeN from nodeR to nodeP,  
remove nodeR,  
flat.  
  
complexity…  
singly doubly  
search – O(n) O(n)  
insert at head/tail – O(1) O(1)  
remove at head – O(1) O(1)  
remove at tail – O(n) O(1)  
remove in middle – O(n) O(n)

--------------------------------------+----------------------------------------------------

STACK  
always check if the STACK is empty

One ended linear data structure – push/pop LIFO  
Python – pop(),append()  
eg(undo),tower of Hanoi, Depth First Search (DFS) graph.

Complexity  
Pushing – O(1)  
pop-O(1)  
peeking-O(1)  
searching-O(n)  
size-O(1)

Arrays,singly or doubly – stacks

Singly stack-  
..pushing  
head- null  
refer new head node pointing to null  
refer new head node pointing to the prev node  
and so on making new heads  
..popping  
traverse to the next node and refer it as head, and de-allocate prev node,  
traverse, re-reference, and de-allocate and so on

-----------------------------------------------------------------------+------------------------------------

QUEUES

Linear - enqueue, dequeue  
front:dequeue and back:enqueue

Enqueue/adding/offering – adding elements to the back- (queue,data)  
Dequeue/polling/removing – removing elements from the from- (queue)

USAGE:

Web server request management, Breadth First Search(BFS) graph traversal.  
BFS- In a network of nodes (graph), visiting a node and visiting all its neighbors and mark them as the frontier,  
visit all the neighbors of the frontiers and mark them as frontiers.  
repeat until every node is visited

COMPLEXITY

Enqueue – O(1)  
Dequeue –O(1)  
Peeking – O(1)  
Contains – O(n) – scan to check data  
Removal– O(n) – scan and remove  
is Empty – O(1)