

Week - 13 - DSA I

- Data Structure ✓
- Algorithms ✓
- memory allocation ✓
- memory leak ✓
- complexity analysis ✓
- asymptotic analysis ✓
- Big-O notation ✓
- Concepts of array ✓
- Concepts of linked list ✓ (set, insert, delete, init)
- Singly linked list ✓
- Doubly linked list ✓
- Concepts of string ✓
- Binary search ✓
- Linear search ✓
- Recursion ✓ (string reverse)
- application of all structures ✓
- Linear vs non-Linear ✓
- + types of ds ✓
- types of memory allocation ✓
- ✓ Contiguous and non contiguous
- ✓ Garbage collection and working
- ✓ adv and disadv of linked list
- ✓ mutable and immutable
- hierarchical DS
- Tagged array.
- adv and disadv of recursion
- List functions.
- virtual memory.
- ✓ circular reference.
- ✓ heterogeneous array.
- ✓ sort, ascending (find middle), descending, linked list, Reverse doubly.
- null termination.
- LF, CR LF
- ✓ unique.

Week 14 . Data structure II

- ✓ Bubble sort $O(n^2)$
 - ✓ Insertion sort $O(n^2)$
 - ✓ Selection sort $O(n^2)$
 - ✓ Quick sort $O(n \log n)$, $O(n^2)$
 - ✓ Merge sort $O(n \log n)$
 - ✓ Stack (types) adv/dis
 - ✓ Queue (types) adv/dis
 - ✓ Push and pop in stack
 - ✓ Enqueue, Dequeue in Queue
 - ✓ Hash tables
 - ✓ Applications all structures
 - ✓ Peek
 - ✓ Stable and unstable
 - ✓ deterministic and undeterministic
 - ✓ Stack underflow and stack overflow
 - ✓ divide and conquer strategy.
 - ✓ adv & dis using stack and queue.
 - ✓ double ended queue.
 - ✓ Hash Functions.
 - ✓ Hash values.
 - ✓ collision.
 - ✓ method to prevent collision.
 - ✓ chaining
 - ✓ linear probing
 - ✓ Quadratic, &
 - ✓ double hashing
 - ✓ Load factor.
 - ✓ array vs hashtable
 - ✓ adv and disadv of hashtable
 - ✓ ~~Stack~~ Bubble sort conditions
 - ✓ Lifo queue
 - ✓ Priority queue
- ### Practicals

 - ✓ reverse string in stack
 - ✓ find long char in string
 - ✓ String Reverse using stack
 - ✓ "aabcdbbefgccc" output: a:1, b:4, c:3
 - ✓ stack second largest
 - ✓ frequency of string using hash map.
 - ✓ delete middle element of stack
 - ✓ valid parenthesis {[]{} }
 - ✓ queue reverse in recursion.
 - ✓ find duplicates using hash table