**Famous Algorithms**

**ARRAYS**

* -> prefix sum
* -> SubArrays
* -> Sliding window
* -> Carry forward
* -> middle element algorithm (self made name)
* -> contribution technique
* -> voting algorithm (boyemoore)
* -> kadane algorithm
* -> using prefix sum in another ways [stocks problem]
* -> rain water storage
* -> merge overlapping
* -> index marking

**2D MATRIX**

* -> sorted matrix search  (start from top right)
* -> row wise sorted binary matrix (row with max 1 count )
* -> spiral traversal (using r,c and boundary size)
* -> contribution technique

**STRING**

* All arrays algorithms
* Middle element
* Sorting
* 2 pointer algo

**HASHING (SET and MAP)**

* find the pair sum exist
* subarray sum to k  in O(n)
* find something in O(1)

**BIT MANIPULATION**

left shift by 1 double the value and right shift by 1 divide the value by 2

check even/odd using &

set the ith bit using OR ,

toggle ith bit using XOR

check status of the ith bit using AND

**bit manipulation algorithms**

* using setbit count at ith place to find unique in duplets(triplets, quadreplets etc)
* 2 unique in duplets : 2 groups on the basis of first set bit of the XOR of array
* max AND for an array :  starting from MSB : start discarding numbers that will be irrelevan in max AND
* OR contribution in sum of value of subarrays
* A^C or C^B <= A^B if A<=C<=B
* sum of all XOR pairs

**RECURSSION**

* fast power
* pattern observation

**SORTING**

* Selection sort
* Bubble sort
* Insertion sort
* Merge sort (Inversion pair)
* Count sort (Array as an hashMap)
* Quick sort

**BINARY SEARCH**

* Search space and a target element (BS-1)
* If we find an answer (that will be in a range) and sorting doesn’t effect anything + on the basis of some operation we can reduce our ans range BS-2
* Max of min / min of max (check is possible and apply binary search on search space) BS-3

**STACKS**

* Balanced paranthesis
* Infix to postfix (using precedency in stacks)
* Min element stack
* Two expressions are same or not
* Nearest greater and smaller
* Histogram area
* Sum of ( max – min )of all sub-arrays