Array Question types---

1. DNF(Dutch National Flag) algo for rearrangement of elements(segregate 0s and 1s, segregate 0s, 1s and 2s in an array). Normal Quick sort partition is single pivot partition, DNF is two pivot three-way partition(we keep indexes for low, high at ends and iterate between them using mid pointer).

2. Elements that occur more than n/2 times(Moor'es voting algo) and n/k times\*\*(generic)

3. Subarray related problem(size variable) : Note that here you can't sort the array.Approaches can be : two pointers approach of extending the subarray towards right and shrinking from left, (this doesn't work if -ve elements are there, use hashing in such case), hashing, DP.

Problems : max sum subrray, K-sum(zero sum) subarray,max length bitonic subarray(DP), max sum circular subarray, max product subarray, (largest)subarray with no of equal 0s and 1s(\*\*), subarray with least avg(use window of size k), count strictly increasing subarrays(using dp and formula approach), smallest subarray with sum greater than K\*\*(use 2 pointer apporach as in K-sum subarray,also tweek to handle for -ve numbers, hashing can't be used here), smallest unsorted subarray that sorts th eentire subarray(2 pointer with starting candindate subarray)

Fixed length subarrays : max of all subarrays of size K(application of Deque), max avg subarray of length k

Based on pairs/triplets/quadraples : pairs/triplets quadraples with sum K and sum zero(sorting & use pointers at start & end, hashing<sum, index> approach), pair with sum closest to K or zero(sorting + 2pointer), pair with given diff(take 2 pointers at first element, if equal increase both, if less increase right else increase left), max diff between 2 pairs(max-min), max diff ordered pair(pair with max diff with j>i), max diff farthest ordered pair(pair(a[i], a[j]) such that a[j]>a[i] and (j-i) is maximum(use of auxiliary arrays), min/max distance beween to given numbers, given 2 sorted subarray, find pair with sum closest to K(2 pointer one at start of fisrt array, other at end of second)

Which require auxiliary arrays : find sorted triplets and inverted triplets, product array puzzle, max diff farthest ordered pair(pair(a[i], a[j]) such that a[j]>a[i] and (j-i) is maximum(2 arrays LMin and RMax and use 2 pointers)\*\*, sorted triplet with max product, rearrange array in max-min form

Missing number search :

case 1 : Numbers in range - one/two number missing...then use sum formula, xor approach, find all missing numbers, find smallest missing in O(n) time constant space

Case 2(no range) : marking by negating and then searching for +ve index. find first missing +ve number

Duplicate finding(in a range) : Specific case(2 elements repeated) - use count array, xor technique, use elements as index. General case(many elements can be repeated) - use absolute val of elemnts as index(prints an element more than once), best approach(linear time constant space - mark visited elements by making them out of range.. index=a[i]%range, a[index] += range. finally check for a[i] with a[i]/range > 1

Binary search based : find peak element in circularly sorted array, search in circularly sorted array, find first/last occurrence of an element, find frequency of an element. searching row/col sorted matrix, row with max no of 1s in a binary matrix, floor/ceilig finding

Based on idea of Merging : max sum path in 2 sorted arrays(can switch at common elements), counting inversions in an array, merge two sorted subarrays, find median of sorted arrays,merge 2 sorted arrays in constant time

On application of stack : implement 2 stacks(k stacks\*\*) in an array, finding next greater element, largest rectangular area in histogram, interval overlap

On application of Queue/Deque : circular tour that visits all petrol pumps, max of all subarrays of size K\*\*(application of Deque)

On application of heap : k largest elements, kth smallest element, median finding in unsorted subarrays, merge k sorted subarrays(k-way merging), online median finding

Counting/Radix sort :

Array rotation - reversal algo(to rotate by k elements....reverse first k, then last n-k, then revrse entire array)

DP based : max non-adjacent sum, max sum subarray, max length chain of pairs, max profit buy/sell stock(2 problems)

Cake-walk : finding leader, equilibrium index, elemnts by frequency\*\* while maintaing the chronological order, consecutiveness check, max-sum-rectangle-in-a-2d-matrix(2D Kadane), nth ugly number,

\*\*Sum of all subarrays = (freq of a[i]) \* a[i]) = sum {(2 ^ (n-1)) \* a[i] } = (2 ^ (n-1)) \* sum(a[i]). Fequency of each element considering all subarrays = 2^(n-1). Other(a bit complex) approach : find sum {(n-i)(i+1) } for i =0...n-1

----------

PENDING : tug of war, petrol pump, pancake sorting, stock buy/sell profit maximize,

http://www.geeksforgeeks.org/find-the-maximum-of-minimums-for-every-window-size-in-a-given-array/

http://www.geeksforgeeks.org/find-the-maximum-of-minimums-for-every-window-size-in-a-given-array/

-------------matrix Problems-------

1.http://www.geeksforgeeks.org/find-median-row-wise-sorted-matrix/

Good problem : https://www.hackerrank.com/challenges/angry-children-2/problem . First sort the array and then find the subarray of size k wih optimal value

----------------Scalability------------

redundancy - replcating

how to solve problem of shared session state(user session) - RAID(Redundant Array of Independent Disks).

load balancer - may usee cookie for storign sesion.

MySQL query cache, memcache

master slave - slave for redundancy(to deal woth single point of failure) and also to distribute read traffic

master-master set up

load balancing +replication : multitiered architecture

replicating loa balancer : active-active, active-passive

db partitioning - for distributing data

geography level load balancing - datacentres