3. Find the "Kth" max and min element of an array

Method 1 : Sort and return kth element

Method 2 : Using Set

- Set in c++ is implemented using BST
- All operations take O(log n) time
- Insert All the element in set and return kth element

8 / 13. Find Largest sum contiguous Subarray / Kadane's Algo

- Store max in variable and currentMax in another variable
- Traverse over the array, add element to currentMax
- If currentMax is larger assign it to max
- If currentMax is negative make it 0

10. Minimum no. of Jumps to reach end of an array

- If First element is 0 can't reach end so ans -1
- If no. elements is < 2 ans is 0

Step is no of steps we can move from current element
maxReach is the max index where we can move from current index
Jump is number of jumps = 1
For i = to n - 1
    If n == n-1 return jumps
    Calculate max reach for current element
    Decrease steps
    If step == 0
        Increase jump
        Step = maxReach - i
        If step < 0 return -1
return -1

11. Find duplicate in an array of N+1 Integers

View Solution in leetcode

12. Merge 2 sorted arrays without using Extra space.

- Traverse first array from end and secont from beginning
- If element in first array is greater than element in 2nd array swap them
- Sort both the array

14. Merge Intervals

- Sort the Intervals
- traverse over the intervals
- if i0 >= (i+1)0 merge them
- else add to the ans

15. Next Permutation

- Find index of first decreasing element from the end
- find element just greater than that element and swap them
- reverse the elements after that index

16. Count Inversion

- Divide and conquer (Merge Sort)
- Explanation - Count Inversion - Merge Sort | C++ Placement Course - 19.3

17. Best time to buy and Sell stock

- Track max profit and current profit
- Current += price[i] - price[i-1]
- If current profit is < 0 make it zero
- Change profit to max of profit and current

18. Find all pairs on integer array whose sum is equal to given number

- Store element and its occurrence in map
- Traverse over the array
    - Find target = k - arr[i]
    - Add occurance of target to count
    - Decrease the occurence of current element

19. Find common elements In 3 sorted arrays

Method 1 :

- Find Common Elements of first 2 array
- Then find common from the previously founded common and the 3rd array

Method 2 :

- Keep 3 pointers
- Check for common elements in all array
- Increase pointer of array which has smallest element

20. Rearrange the array in alternating positive and negative items with O(1) extra space

Method 1 : Using Extra space

- Store positive and negative elements in different arrays
- Write them in to the ans array at appropriate position

Method 2 : Without Extra Space

- If element is not at correct position find the correct element for that position and insert that correct element at that position

21. Find if there is any subarray with sum equal to 0

Simple Idea behind this is if the array contains such subarray then adding elements one by one will give a sum that has been seen before.

22. Find factorial of a large number

- Store multiplication in vector or array in descending order --- 120 -> 021
- Write a function to Multiply a vector to number and store it in same way
- Now just multiply vector with [2,N]

23. find maximum product subarray

- Track max and min of products
- Ans is max of ans and max product

24. Find longest coinsecutive subsequence

- Sort the array
- If the previous element is 1 lesser than the current element increase count by 1
- If both are same skip
- Else make count 0
- Update ans by max of count and ans

25. Given an array of size n and a number k, fin all elements that appear more than " n/k " times.

Use HashMap to Store number of occurrence

26. Maximum profit by buying and selling a share atmost twice

27. Find whether an array is a subset of another array

Use Hash Map

28. Find the triplet that sum to a given value

Sort the vector
traverse the vector and fix first element
    if the element is same as previous skip
For other to element
    * 2 pointers - start and end
    * if sum is greater than target decrease the right pointer
    * else increase left pointer
    * if both add up to target add all three element to ans
        * return true
Return false

29. Trapping Rain water problem

Method 1 : Dynamic Programming

- Start searching for max from right and store those values at each index in array named suffix
- Do same from left side and store in array named prefix
- For each element add min(prefix[i],suffix[i]) - arr[i] to the ans
- Return ans

Method 2 : 2 Pointers Approach

30. Chocolate Distribution problem

- Sort the array and use sliding window

31. Smallest Subarray with sum greater than a given value

- 2 pointers init both with 0,start,end
- While end < n
- Add arr[end] to sum till sum is not greater than given value
- Now start subtracting element from sum till sum is greater than given value and update length is sum is > greater than given value each time

32. Three way partitioning of an array around a given value

- 2 Pointer
- smaller = 0 ---> Indicates the position of next smaller element
- end = n - 1---> indicates the position of next greater element
- Traverse the array
    - If the element is smaller than the range swap it with smaller index
    - If the element is greater than the range swap it with greater index
    - and make sure pointer stays at same element

33. Minimum swaps required bring elements less equal K together

Good Element --- less than or equal to k

- Find No of good element - g
- Take a sliding window of size of g.
- Find No of good ele in first g elements
- Start from 1 and find max no of good elements a window have
- Return g - maxgood

34. Minimum no. of operations required to make an array palindrome

35. Median of 2 sorted arrays of equal size

36. Median of 2 sorted arrays of different size