1. 2 sum problem
    a. Sort array and maintain two pointer low and high
    b. If a[low] + a[high] < target

```
        Low++
Else
        High--
```

2. 3 sum problem (all solution without duplicates)
    a. https://leetcode.com/problems/3sum/discuss/7380/Concise-O(N2)-Java-solution
    b. Sort array
    c. For each element at index i
        i. Target = 0 - a[i]
        ii. In range i + 1 to len apply 2 sum problem
    d. Similar problem 3 sum closest https://leetcode.com/problems/3sum-closest/

3. Overlapping Intervals :
    a. Hotel booking / cab booking
        i. https://www.geeksforgeeks.org/find-k-bookings-possible-given-arrival-departure-times/

```
1. Create event points for every interval start, and end.
2. Sort it according to the day.
3. Now, iterate over them one by one. If you encounter a start, then the number of
active guests increase by one. If you encounter an end, the number of active
guests decrease by one.
4. If at any point, the number of active guests exceed K, we know that scheduling
is not possible.
```

    b. https://www.interviewbit.com/problems/merge-intervals/

    c. Min shoots to shoot all balloons
        i. https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/submissions/
        ii. First approach
            1. Sort Intervals by end point
            2. shoot point = endpoint of first interval
            3. Now all the next intervals whose starting point < shoot point will be shot by shoot point
            4. Else new shoot point = current interval's endpoint
        iii. Second approach
            1. Sort intervals in decreasing order by start point
            2. Shoot point = start point of first interval
            3. Now all the next intervals whose end point > shoot point will be shot by shoot point
            4. Else new shoot point = current inteval's start point

```
iv.    1 2 3 4 5 6 7 8 9
v.     Sort by end point          rev sort by start point
vi.    = = = = =                          = = =
vii.     = = =  =  =              = = = = =
viii.  .
ix.    = = =                              = = =
x.            =  = =              = = =
```

    d. Non overlapping Intervals (minimum number of intervals you need to remove to make the rest of the intervals non-overlapping)

https://leetcode.com/problems/non-overlapping-intervals/
https://www.youtube.com/watch?v=_W0NzvQXnHg
    i. Almost same as above problem
    ii. Sort Intervals by end point
    iii. shoot point = endpoint of first interval
    iv. Now all the next intervals whose starting point < shoot point will be intersected by shoot point so will have to remove it hence count++
    v. Else new shoot point = current interval's endpoint
    vi. --------------------------------
    vii. Sort the intervals by their right end ascending.
    viii. Initialized the select intervals as an empty set
    ix. Consider the sorted intervals one by one, add it if it is possible (only need to check the last select interval and the current one).
  e. Merge Intervals
    i. https://leetcode.com/problems/merge-intervals/
    ii. https://www.interviewbit.com/problems/merge-overlapping-intervals/ **
  f. Approach 1
    i. Sort intervals by decreasing order of end point
    ii. X = start point y = end point
    iii. Now all the points whose end point >= x will intersect so new x = min(x, new point start point)
    iv. Else they do not intersect
  g. Approach 2 **
    i. Sort intervals by increasing order of start point
    ii. X = start point y = end point
    iii. Now all points whose start point <= y will internsect so new y = max(y, new point end point)
    iv. Else they do not intersect
  h.
```
=============                    ===                              ==
           ==                              ===                               ==
              ====                                ===                 ==========
```

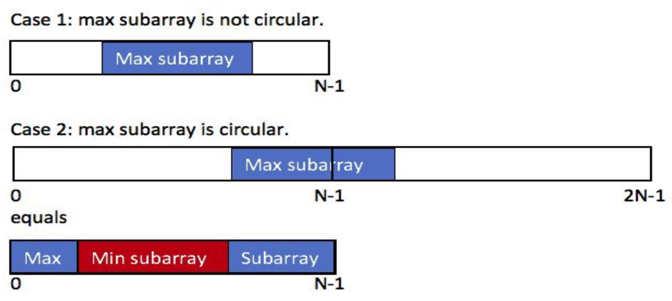
Cant inc sort by end point

4. Next Permutation
  a. https://leetcode.com/problems/next-permutation/solution/
  b. Go from right side find the first i for which a[i] < a[i-1]
  c. Now from right side of a[i] find the number immediate larger than a[i] swap both
  d. Sort right side of a[i]

5. Kth permutation
  a. https://leetcode.com/problems/permutation-sequence/

    b. N = 4 and k = 9
       Take k = 8 (index 0)
       1 2 3 4
       1 {2 3 4}
       2 {1 3 4}
       3 {1 2 4}
       K = 9 index = k/(n-1)! = 9/3! = 1
       new k = k %(n-1)! = 9%3! = 3
       ans = 2 + recursively

6. Max sub circular sub array
   https://leetcode.com/problems/maximum-sum-circular-subarray/submissions/


Case 1: max subarray is not circular.
Max subarray
0
N-1
Case 2: max subarray is circular.
Max subarray
0
N-1
2N-1
equals
Max
Min subarray
Subarray
0
N-1

```
Ans = max(the max subarray sum, the total sum - the min subarray sum)
```

7. Zigzag traversal of string
    a. https://www.interviewbit.com/problems/zigzag-string/
    b. Create vector of vector of size k and keep direction pointer
    c. Change dir at j = 0 or j = k - 1 push s[i] into vec[j]

```
0 P.......A........H.......N
1 ..A..P....L....S....I...I....G
2 ....Y.........I........R
```

8. Moore's Majority element

More than n/2

```
int candidate = 0, count = 0, n = a.size();
        for(int i=0;i<a.size();i++){
            if(count == 0)
                candidate = a[i];
            if(a[i] == candidate)
                count++;
            else
                count--;
        }
        int freq = 0;
        for(int i=0;i<n;i++)
            if(a[i] == candidate) freq++;
        return (freq > n/2) ? candidate : -1;
```

More than n/3

```
count1, count2, candidate1, candidate2 = 0, 0, 0, 1
    for n in nums:
        if n == candidate1:
            count1 += 1
        elif n == candidate2:
            count2 += 1
        elif count1 == 0:
            candidate1, count1 = n, 1
        elif count2 == 0:
            candidate2, count2 = n, 1
        else:
            count1, count2 = count1 - 1, count2 - 1
    return [n for n in (candidate1, candidate2)
                if nums.count(n) > len(nums) //
3]
```

https://leetcode.com/problems/majority-element/
https://leetcode.com/problems/majority-element-ii/
https://gregable.com/2013/10/majority-vote-algorithm-find-majority.html

9. **Search in sorted array
    https://leetcode.com/problems/search-in-rotated-sorted-array/submissions/
    https://leetcode.com/problems/search-in-rotated-sorted-array-ii/submissions/
    https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/
    https://leetcode.com/problems/find-minimum-in-rotated-sorted-array-ii/submissions/

10. https://www.spoj.com/problems/MMASS/
    https://github.com/jiteshsunhala/spoj-solutions/blob/master/MMASS.cpp
    (CH)2((OH2)3H2O)5
    USING stack

11. Maximum Absolute Difference
    a. https://www.interviewbit.com/problems/maximum-absolute-difference/
    b. Four test cases
        i. Case 1: A[i] > A[j] and i > j
           |A[i] - A[j]| = A[i] - A[j]
           |i -j| = i - j
           hence, f(i, j) = (A[i] + i) - (A[j] + j)
        ii.
        iii. Case 2: A[i] < A[j] and i < j
           |A[i] - A[j]| = -(A[i]) + A[j]
           |i -j| = -(i) + j
           hence, f(i, j) = -(A[i] + i) + (A[j] + j)

        iv. Case 3: A[i] > A[j] and i < j
           |A[i] - A[j]| = A[i] - A[j]
           |i -j| = -(i) + j
           hence, f(i, j) = (A[i] - i) - (A[j] - j)

        v. Case 4: A[i] < A[j] and i > j
           |A[i] - A[j]| = -(A[i]) + A[j]
           |i -j| = i - j
           hence, f(i, j) = -(A[i] - i) + (A[j] - j)

           We can construct two arrays with values: `A[i] + i` and `A[i] - i`. Then, for above 2 cases, we find the maximum value possible. For that, we just have to store minimum and maximum values of expressions `A[i] + i` and `A[i] - i` for all `i`.

12. Repeat and Missing Number
    a. https://www.interviewbit.com/problems/repeat-and-missing-number-array/
    b. Expsum - givensum = mis - rep
       Expsquaresum - Givensquaresum = mis^2 - rep^2

13. Spiral Matrix
    a. https://www.interviewbit.com/problems/spiral-order-matrix-ii/

```
[ [ 1, 1, 1, 1, 1, 1 ],        top: c from c1 ... c2
  [ 1, 2, 2, 2, 2, 1 ],        right: r from r1+1 ... r2
  [ 1, 2, 3, 3, 2, 1 ],        bottom: c from c2+1 ... c1+1
  [ 1, 2, 2, 2, 2, 1 ],        left: r from r2+1 ... r1+1
  [ 1, 1, 1, 1, 1, 1 ] ]
```

```
int r1=0,c1=0,r2=n-1,c2=n-1, count = 1;
    while(r1 <= r2){
        for(int j=c1;j<=c2;j++)
            A[r1][j] = count++;
        for(int i=r1+1;i<=r2;i++)
            A[i][c2] = count++;
        for(int j=c2-1;j>=c1;j--)
            A[r2][j] = count++;
        for(int i=r2-1;i>=r1+1;i--)
            A[i][c1] = count++;
        r1++;c1++;
        r2--;c2--;
    }
```

14. Diagonal Matrix
    a. https://www.interviewbit.com/problems/anti-diagonals/
    b. Traverse matrix by i and j and push that element in ans vector's index

15. Largest Number that can be form from given numbers
    a. https://leetcode.com/submissions/detail/302890019/
    b. Create a int vector and add comparator

```
static bool mycomp(int a,int b){
        string x = to_string(a);
        string y = to_string(b);
        return x+y>y+x;
    }
```

16. Rotate a Matrix
    a. 90 degree
    https://leetcode.com/problems/rotate-image/discuss/18872/A-common-method-to-rotate-the-image

```
clockwise rotate
 * first reverse up to down, then swap the symmetry
 * 1 2 3     7 8 9     7 4 1
 * 4 5 6  => 4 5 6  => 8 5 2
 * 7 8 9     1 2 3     9 6 3

anticlockwise rotate
 * first reverse left to right, then swap the symmetry
 * 1 2 3     3 2 1     3 6 9
 * 4 5 6  => 6 5 4  => 2 5 8
 * 7 8 9     9 8 7     1 4
```

17. https://www.interviewbit.com/problems/first-missing-integer/

18. Maximum sum subarray in array of concatenation k times
    https://www.codechef.com/LRNDSA06/submit/KCON
        https://www.codechef.com/viewsolution/33364416
        A1 + A2 + A3 + ….. + Ak
        If sum(A) is positive { ans = maxsuffix + sum(A)*(k-2) + maxprefix}
        Else ans = max(maxsuffix + maxprefix, caddens max sum)

19. ATOI
    a. https://www.interviewbit.com/problems/atoi/
    b. Handle overflow
       +7 → 7   +96afaddsf → 96   __-98dfd → -98
       -7 → -7   -78dfe → -78

20. Roman to Num
    a. Traverse in reverse order if a[i] >= a[i+1] then ans += a[i]
       Else ans -= a[i]

```
Input: "LVIII"
Output: 58
Explanation: L = 50, V= 5, III = 3.

Input: "MCMXCIV"
Output: 1994
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.
```

    b. Integer to Roman

```
string M[] = {"", "M", "MM", "MMM"};
string C[] = {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"};
string X[] = {"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"};
string I[] = {"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"};
return M[num/1000] + C[(num%1000)/100] + X[(num%100)/10] + I[num%10];
```

21. Validate sudoku
    a. https://www.interviewbit.com/problems/valid-sudoku/
    b. Create hash for row, column and 3*3 board and check whether no present among any one
22. Group anagram strings together
    a. https://www.interviewbit.com/problems/anagrams/
    b. Create map<vector<string>,i> check if string anagram already exist
23. Max points on a line
    a. https://leetcode.com/problems/max-points-on-a-line/
    b. Take each point and find slop with all other points
24. String concatenation
    a. https://leetcode.com/problems/substring-with-concatenation-of-all-words/
    b. Create a two map mp1 and mp2. Map1 stores freq of dict1 and mp2 while traversing given stg s

```
For each index i in range(0,n-1)
        J = i
        while(j < n)
                If mp1 contains stg and mp2[stg] + 1 <= mp1[stg] than add stg to map2
```

```
                    if(size1 == size2) ans.add(i) break
                Else break
```

25. Kth maximum
    a. Use min priority queue insert element one by one if q.size() > k pop one element
    b. https://leetcode.com/problems/kth-largest-element-in-an-array/
26.
    a. Kth minimum in row wise and column wise matrix
        i. https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/
        ii. Insert first row in priority queue
        iii. Now pop min and insert next element (x+1,y) in priority queue
    b. N pair maximum
        i. https://leetcode.com/problems/find-k-pairs-with-smallest-sums/
27. Rearrange array so no same elements be an adjacent
    a. https://leetcode.com/problems/distant-barcodes/
    b. Approach 1:

        Use a priority queue and insert all element and frequence

        Select two most frequent elements at a time and insert it into ans
    c. Approach 2 : faster O(n)

        Find the most frequent element and fill it at even positions

        Now fill rest of the element in remaining even positions and odd positions