DIY:

- https://leetcode.com/problems/next-greater-element-ii/ (https://leetcode.com/problems/next-greater-element-ii/)
- https://leetcode.com/problems/power-of-two/ (https://leetcode.com/problems/power-of-two/)
- https://leetcode.com/problems/decode-string/ (https://leetcode.com/problems/decode-string/)

In [ ]:

Discuss solution for
https://leetcode.com/problems/next-greater-element-ii/ (https://leetcode.com/problems/next-greater-element-ii/)

In [ ]:

Merging 2 sorted arrays

given 2 sorted arrays, create a third array, which contains elements from both the arrays in ascending order.

a1 [1 3 5 8 10 20] i a2 [2 3 6 7] j size: m , n

a3: size m + n

Solution-1:
copy a1 into a3, copy a2 into a3 after a1. => O(m+n) then sort a3 => O((m+n) log (m+n))

Solution-2:
Use one pointer in each of the array, pick the smaller current element from one the arrays and advance the pointer to next position. 1 2 3 3 5 6 7 8 10 20 O(m + n)

```
vector<int> merge(vector<int> a1, vector<int> a2) {
    vector<int> res(a1.size() + a2.size());

    int i = 0, j = 0, k = 0;

    // merge
    while(i < a1.size() && j < a2.size() ) {
            if (a1[i] < a2[j]) {
                res[k] = a1[i];
                i++;
            } else  {
                res[k] = a2[j];
                j++;
            }
            k++;
    }

    // copy over remaining data
    while(i < a1.size() ) {
            res[k] = a1[i];
            i++;
            k++;
    }

    while(j < a2.size() ) {
            res[k] = a2[j];
            j++;
            k++;
    }

    return res;
}
```

In [ ]:

Question-1
https://leetcode.com/problems/merge-sorted-array/ (https://leetcode.com/problems/merge-sorted-array/)

In [ ]:
```java
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        int i = m-1;
        int j = n-1;
        int k = m + n - 1;
        //System.out.println(k);
        while(i>=0 && j>=0){
            if(nums1[i] > nums2[j]){
                nums1[k]=nums1[i];
                i--;
            } else{
                nums1[k]=nums2[j];
                j--;
            }
            k--;
        }

        while(j>=0){
            nums1[k]=nums2[j];
            j--;
            k--;
        }

    }
}
```

In [ ]:
```java
        int i=m-1;
        int j=n-1;
        int k=nums1.length-1;

        while(j>=0){
            if(i>=0 && nums1[i]>nums2[j]){
                nums1[k]=nums1[i];
                k--;
                i--;
            }else{
                nums1[k] = nums2[j];
                k--;
                j--;
            }
        }
```

In [ ]:
```javascript

var merge = function (nums1, m, nums2, n) {
    let i = m - 1; // nums1 size 2
    let j = n - 1; // nums2 size
    let k = nums1.length - 1; // nums1 total length
    while (j >= 0) {
        console.log(i,j,nums1[i] , nums2[j])
        if (i >= 0 & j >= 0 && nums1[i] > nums2[j]) {
            nums1[k--] = nums1[i--];
        }
        else {
            nums1[k--] = nums2[j--];
        }

    }
};
```

In [ ]:
```
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        int p1 = m-1;
        int p2 = n-1;
        for(int i =nums1.length-1;i>=0 ;i--){
            int valAtP1 = p1>=0 ? nums1[p1] : Integer.MIN_VALUE;
            int valAtP2 = p2>=0 ? nums2[p2] : Integer.MIN_VALUE;
            if(valAtP1> valAtP2){
                nums1[i] = valAtP1;
                p1--;
            }else{
                nums1[i] = valAtP2;
                p2--;
            }
        }
    }
}
```

In [ ]:
```

```

Sorting Algorithms

Bubble Sort

Iter-1 0 1 2 3 4, size=5
4 3 5 2 1
3 4 5 2 1
3 4 5 2 1
3 4 2 5 1
3 4 2 1 5

iter-2 3 4 2 1 5
3 4 2 1 5
3 2 4 1 5
3 2 1 4 5

iter-3 3 2 1 4 5
2 3 1 4 5
2 1 3 4 5

iter-4 2 1 3 4 5
1 2 3 4 5

Data is sorted in asc order.

```
    void bubbleSort(vector<int> nums) {

        for(int i = 0; i  < nums.size() - 1; i++) {
            for(int j = 0; j < nums.size() - i - 1; j++) {

                if (nums[j] > nums[j+1m]) {
                    int temp = nums[j];
                    nums[j] = nums[j+1];
                    nums[j+1] = temp;
                }

            }
        }
    }
```

If data is already sorted: still has O(n^2) TC.

```
void bubbleSort(vector<int> nums) {

    for(int i = 0; i  < nums.size() - 1; i++) {
        bool sorted = true;

        for(int j = 0; j < nums.size() - i - 1; j++) {

            if (nums[j] > nums[j+1]) {
                int temp = nums[j];
                nums[j] = nums[j+1];
                nums[j+1] = temp;
                sorted = false;
            }

        }

        if (sorted) {
            break;
        }
    }
}
```

In [ ]:

Sliding Window

Question-1
https://leetcode.com/problems/max-consecutive-ones/ (https://leetcode.com/problems/max-consecutive-ones/)

In [ ]:

```C++
class Solution {
public:
    int findMaxConsecutiveOnes(vector<int>& nums) {
        int count=0;
        int maxCount = 0;

        for(int i =0; i < nums.size(); i++) {
            if (nums[i] == 1) {
                count++;
                maxCount = count > maxCount ? count: maxCount;
            } else {
                count = 0;
            }
        }
        return maxCount;
    }
};
```

Question-2
https://leetcode.com/problems/longest-substring-without-repeating-characters/ (https://leetcode.com/problems/longest-substring-without-repeating-characters/)

In [ ]:
```C++
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        // abcabcb
        // 0123456

        // i 0  .... 1
        // j 1 2 ... 2
        if (s.size() == 0) return 0;

        int maxLen = 1;
        for(int i=0; i < s.size(); i++) {
            for( int j = i + 1; j < s.size(); j++) {
                if (isUnique(s, i, j) && (j-i+1) > maxLen) {
                    maxLen = (j-i+1);
                }
            }
        }
        // TC: O(n^3)
    }

    bool isUnique(string s, int start ,int end) { // O(end-start)
        // TODO: implement, start and end are included
        std::set<char> s;
        for(int i = start; i <= end; i++) {
            if (s.find(s.charAt(i)) != s.end()) {
                return false;
            }
            s.insert(s.charAt(i));
        }
        return false;
    }
};
```

In [ ]:
```
abcabcbb
01234567
       i,j

bacbcadab
012345678

i =      0            1       2      3   3
j =      0   1   2    3       4      4   5         6
maxLen = 0   1   2    3                          4
s        {} {b} {b,a} {b,a,c} {b,c} {b,c} {b,c,a} {b,c,a,d}
s = hashmap<chat, bool>{}

while(j < s.size()) { // O(N)
    char c = s.charAt(j);

    if (c is present in set) { // O(1)
        while(s[i] != s[j]) { // O(N) !!!!!!
            set.remove(s[i])
            i++
        }
        i++
    } else {
        s.add(c,true);
        maxLen = max(maxLen, s.size())
    }
    j++;
}

// TC: O(N)
// SC: O(N)
```

Question-3

https://leetcode.com/problems/max-consecutive-ones-ii/ (https://leetcode.com/problems/max-consecutive-ones-ii/) : https://www.lintcode.com/problem/883/ (https://www.lintcode.com/problem/883/)

In [ ]:

Question-4

https://leetcode.com/problems/max-consecutive-ones-iii/ (https://leetcode.com/problems/max-consecutive-ones-iii/)

In [ ]:

In [ ]: