Lab 6

(i)705

lass MyHashSet {

    List<Integer> ls;

 public MyHashSet() {

        ls= new ArrayList<>();

    }

    public void add(int key) {

        if(!ls.contains(key)) ls.add(key);

    }

    public void remove(int key) {

        if(ls.contains(key)) ls.remove(Integer.valueOf(key));

    }

    public boolean contains(int key) {

        return ls.contains(key);

    }

}

(ii)706

class MyHashMap {

    private static int[] map = new int[1000001];

    public MyHashMap() {

        Arrays.fill(map, -1);

    }

    public void put(int key, int value) {

        map[key] = value;

    }

    public int get(int key) {

        return  map[key];

    }

    public void remove(int key) {

        map[key] = -1;

    }

}

(iii)1512

class Solution {

public int numIdenticalPairs(int[] A) {

int ans = 0,cnt[]= new int [101];

for (int a: A){

    ans += cnt[a]++;

}

return ans;

}

  }

(iv)645

class Solution {

    public int[] findErrorNums(int[] nums) {

      int N = nums.length, sum = N \* (N + 1) / 2;

        int[] result = new int[2];

        boolean[] seen = new boolean[N+1];

        for (int num : nums) {

            sum -= num;

            if (seen[num]) result[0] = num;

            seen[num] = true;

        }

        result[1] = sum + result[0];

        return result;

    }

}

(v)219

class Solution {

    public boolean containsNearbyDuplicate(int[] nums, int k) {

        HashMap<Integer,Integer> hm=new HashMap<>();

        for(int i=0;i<nums.length;i++){

            if(hm.containsKey(nums[i]) && Math.abs(hm.get(nums[i])-i)<=k)return true;

            hm.put(nums[i],i);

        }

        return false;

    }

}

(vi)2099

class Solution {

    public int[] maxSubsequence(int[] nums, int k) {

        PriorityQueue<Integer> pq=new PriorityQueue<>();

       List<Integer> v=new ArrayList<>();

        for(int x:nums){

            pq.add(x);

            v.add(x);

        }

        int[] ans=new int[k];

        for(int i=0;i<nums.length-k;i++)

            v.remove(pq.remove());

        for(int i=0;i<k;i++)

            ans[i]=v.get(i);

        return ans;

    }

}

(vii)290

class Solution {

    public boolean wordPattern(String pattern, String s) {

     String [] arr = s.split(" ");

        if(pattern.length()!=arr.length)return false;

        HashMap<Character,String> map = new HashMap<>();

        for(int i = 0; i<pattern.length(); i++){

            char ch = pattern.charAt(i);

            boolean containsKey = map.containsKey(ch);

            if(map.containsValue(arr[i]) && !containsKey)return false;

            if(containsKey && !map.get(ch).equals(arr[i]))return false;

            else map.put(ch,arr[i]);

        }

        return true;

    }

}

(viii)1

class Solution {

    public int[] twoSum(int[] nums, int target) {

   int n = nums.length;

        for (int i = 0; i < n - 1; i++) {

            for (int j = i + 1; j < n; j++) {

                if (nums[i] + nums[j] == target) {

                    return new int[]{i, j};

                }

            }

        }

        return new int[]{};

    }

}