Dsa lab 8

Arrays

1)1464

class Solution {

    public int maxProduct(int[] nums) {

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

            for(int j=i+1; j<nums.length; j++){

                if(nums[i] > nums[j]){

                    int temp = nums[i];

                    nums[i] = nums[j];

                    nums[j] = temp;

                }

            }

        }

        return (nums[nums.length-1] -1) \* (nums[nums.length-2] - 1);

    }

}

2)1046

class Solution {

    public int lastStoneWeight(int[] stones) {

      Arrays.sort(stones);

        ArrayList<Integer> list = new ArrayList<>();

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

            list.add(stones[i]);

        }

        while(list.size() > 1) {

           int val1 = list.get(list.size()-1);

           int val2 = list.get(list.size()-2);

           list.remove(list.size()-1);

           list.remove(list.size()-1);

           if(val1>val2 || val2>val1) {

                list.add(Math.abs(val1-val2));

           }

           Collections.sort(list);

        }

        if(list.size()==1) {

            return list.get(0);

        }

        else {

            return 0;

        }

    }

}

3)2231

class Solution {

    public int largestInteger(int num) {

     int[] ans = new int[10];

        int x = num;

        while (x != 0) {

            ans[x % 10]++;

            x /= 10;

        }

        x = num;

        int result = 0;

        int t = 1;

        while (x != 0) {

            int v = x % 10;

            x /= 10;

            for (int y = 0; y < 10; ++y) {

                if (((v ^ y) & 1) == 0 && ans[y] > 0) {

                    ans[y]--;

                    result += y \* t;

                    t \*= 10;

                    break;

                }

            }

        }

        return result;

    }

}

4)506

class Solution {

    public String[] findRelativeRanks(int[] score) {

      String[] ans1 = new String[score.length];

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

        int max = -1;

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

             if(max < score[i]){

                 max = score[i];

             }

        }

          int[] arr = new int[max+1];

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

            arr[score[i]]++;

        }

        int j = 1;

        for(int i=arr.length-1; i>=0; i--){

            if(arr[i] != 0){

                map.put(i,j++);

            }

        }

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

               int pos = map.get(score[i]);

               if(pos == 1){

                   ans1[i] = "Gold Medal";

               }else if(pos == 2){

                    ans1[i] = "Silver Medal";

               }else if(pos == 3){

                   ans1[i] = "Bronze Medal";

               }else{

                   ans1[i] = "" + pos;

               }

        }

        return ans1;

    }

}

5)215

class Solution {

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

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

        for (int x : nums) {

            p.offer(x);

            if (p.size() > k) {

                p.poll();

            }

        }

        return p.peek();

    }

}

6)703

class KthLargest {

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

    int k;

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

        this.k = k;

        for (int num : nums)

            store.add(num);

        Collections.sort(store);

    }

    public int add(int val) {

        int pos = binarySearch(val);

        store.add(pos, val);

        return store.get(store.size() - k);

    }

    public int binarySearch(int val) {

        int low = 0, high = store.size() - 1, res = high + 1;

        while (low <= high) {

            int mid = (high - low) / 2 + low;

            if (store.get(mid) == val)

                return mid;

            else if (store.get(mid) > val) {

                res = mid;

                high = mid - 1;

            } else {

                low = mid + 1;

            }

        }

        return res;

    }

}

7)846

class Solution {

    public boolean isNStraightHand(int[] hand, int groupSize) {

        Arrays.sort(hand);

       int len = hand.length;

       boolean[] visited = new boolean[len];

       for(int i = 0;i < len;i++){

           if(visited[i]) continue;

           visited[i] = true;

           int j = i + 1;

           int count = 1;

           int num = hand[i] + 1;

           for(count = 1;j < len && count < groupSize;j++){

               if(visited[j]) continue;

               if(hand[j] > num) break;

               if(hand[j] == num ) {

                   num++;

                   visited[j] = true;

                   count++;

               }

           }

           if(count < groupSize) return false;

       }

       return true;

    }

}

8)355

class Twitter {

    ArrayList<Tweet> tweets;

    HashMap<Integer, User> users;

    public Twitter() {

        tweets = new ArrayList<>();

        users = new HashMap<>();

    }

    public void postTweet(int userId, int tweetId) {

        if(!users.containsKey(userId))

            users.put(userId, new User(userId));

        tweets.add(new Tweet(tweetId, userId));

    }

    public List<Integer> getNewsFeed(int userId) {

        if(!users.containsKey(userId))

            users.put(userId, new User(userId));

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

        int i = tweets.size()-1;

        while(ans.size()<10 && i>=0) {

            if(tweets.get(i).userId==userId || users.get(userId).follows.contains(tweets.get(i).userId))

                ans.add(tweets.get(i).tweetId);

            i--;

        }

        return ans;

    }

    public void follow(int followerId, int followeeId) {

        if(!users.containsKey(followerId))

            users.put(followerId, new User(followerId));

        if(!users.containsKey(followeeId))

            users.put(followeeId, new User(followeeId));

        User user = users.get(followerId);

        user.follows.add(followeeId);

    }

    public void unfollow(int followerId, int followeeId) {

        if(!users.containsKey(followerId))

            users.put(followerId, new User(followerId));

        if(!users.containsKey(followeeId))

            users.put(followeeId, new User(followeeId));

        User user = users.get(followerId);

        user.follows.remove(followeeId);

    }

}

class Tweet {

    int tweetId;

    int userId;

    Tweet(int t, int u) {

        tweetId = t;

        userId = u;

    }

}

class User {

    int userId;

    HashSet<Integer> follows;

    User(int u) {

        userId = u;

        follows = new HashSet<>();

    }

}

9)703

class KthLargest {

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

    int k;

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

        this.k = k;

        for (int num : nums)

            store.add(num);

        Collections.sort(store);

    }

    public int add(int val) {

        int pos = binarySearch(val);

        store.add(pos, val);

        return store.get(store.size() - k);

    }

    public int binarySearch(int val) {

        int low = 0, high = store.size() - 1, res = high + 1;

        while (low <= high) {

            int mid = (high - low) / 2 + low;

            if (store.get(mid) == val)

                return mid;

            else if (store.get(mid) > val) {

                res = mid;

                high = mid - 1;

            } else {

                low = mid + 1;

            }

        }

        return res;

    }

}

10)347

class Solution {

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

     Map<Integer, Integer> map = new HashMap<>();

        for (int i : nums)

            map.merge(i, 1, Integer::sum);//For Getting Frequency

        List<Integer> list = new ArrayList<>(map.keySet());

        list.sort((a, b) -> map.get(b) - map.get(a)); //Sort by Frequency in descending order

        int res[] = new int[k];

        for (int i = 0; i < k; ++i)

            res[i] = list.get(i);

        return res;

    }

}