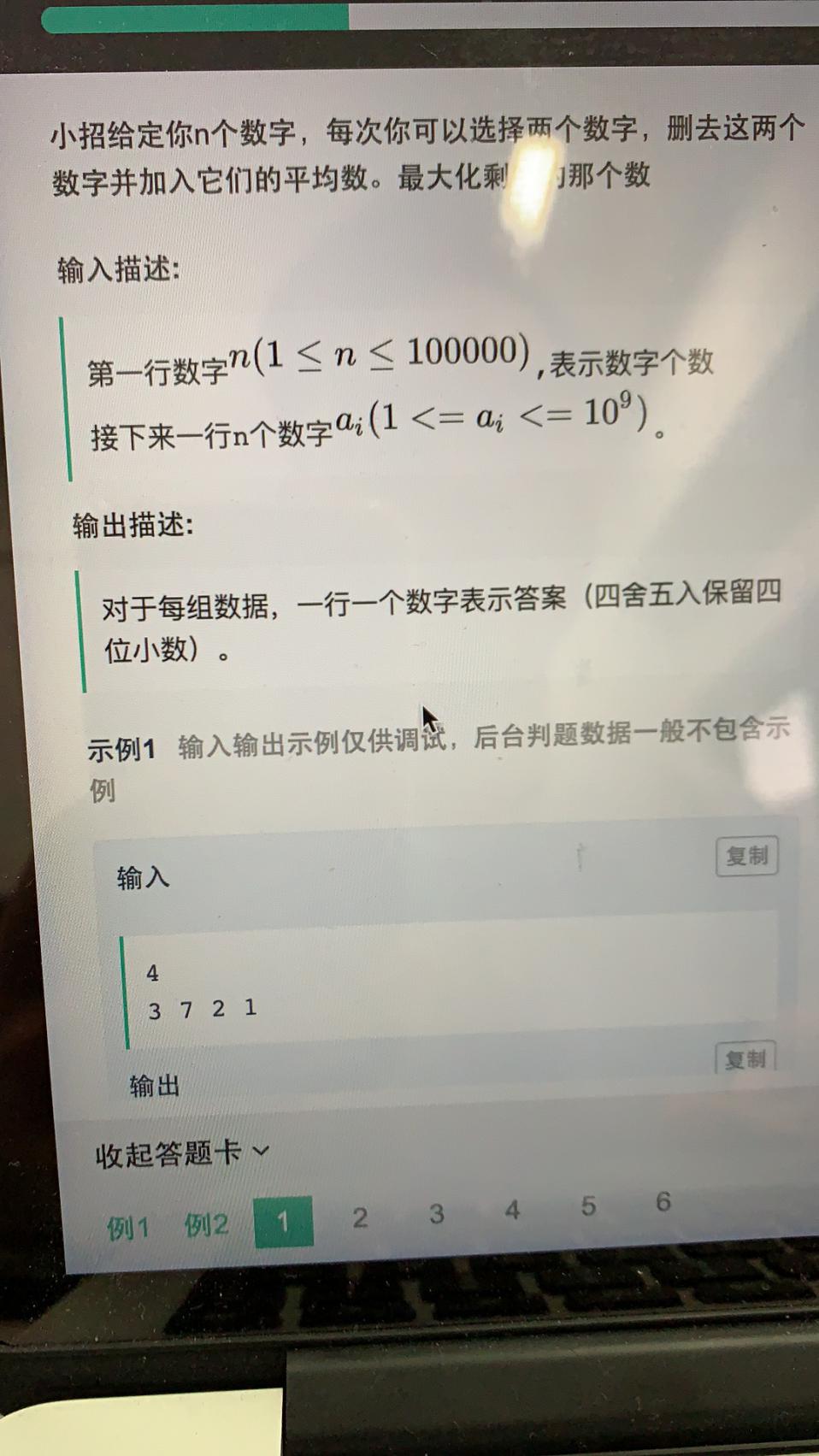
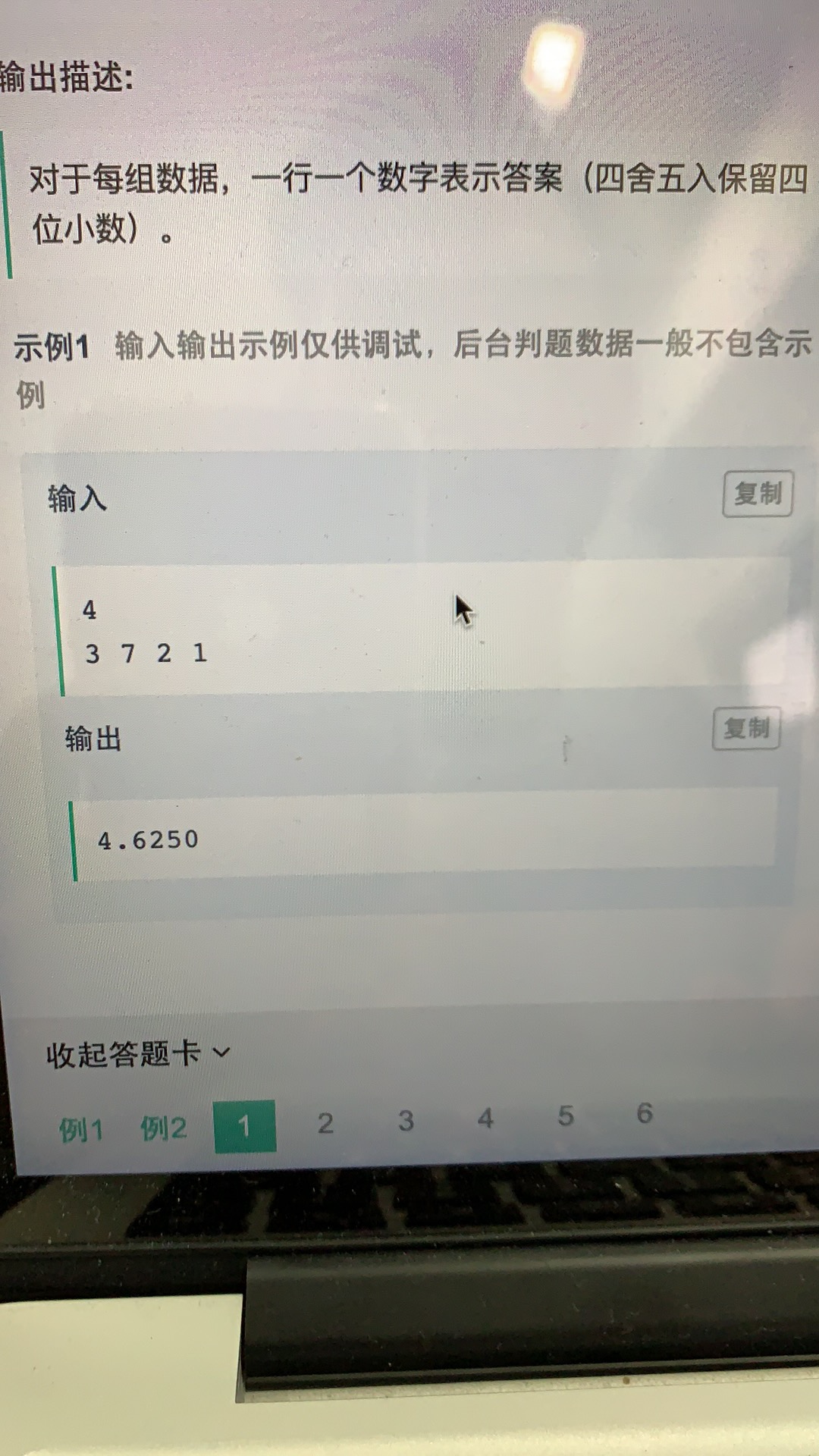
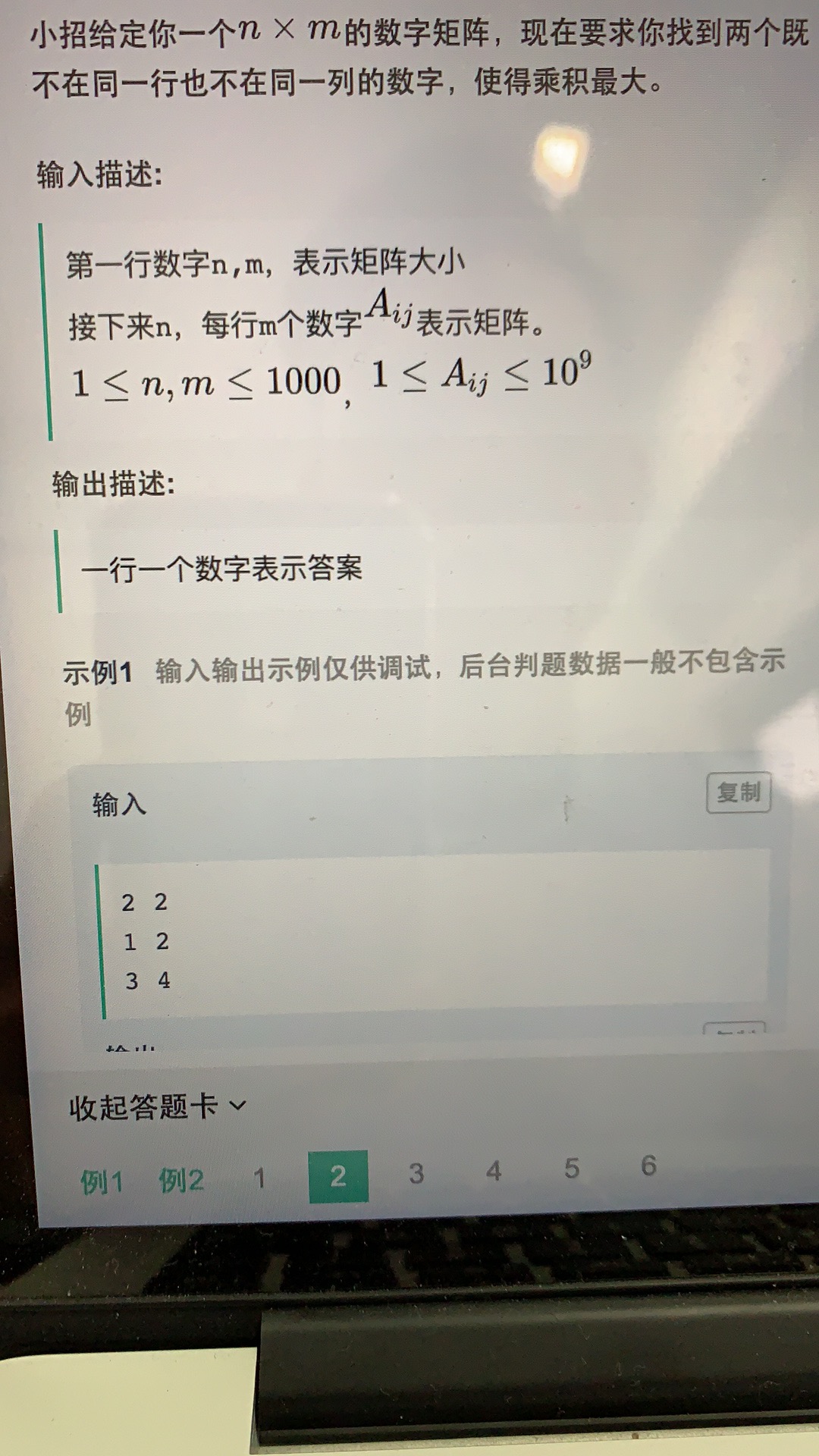
# 招商银行M-Geeker技术竞赛

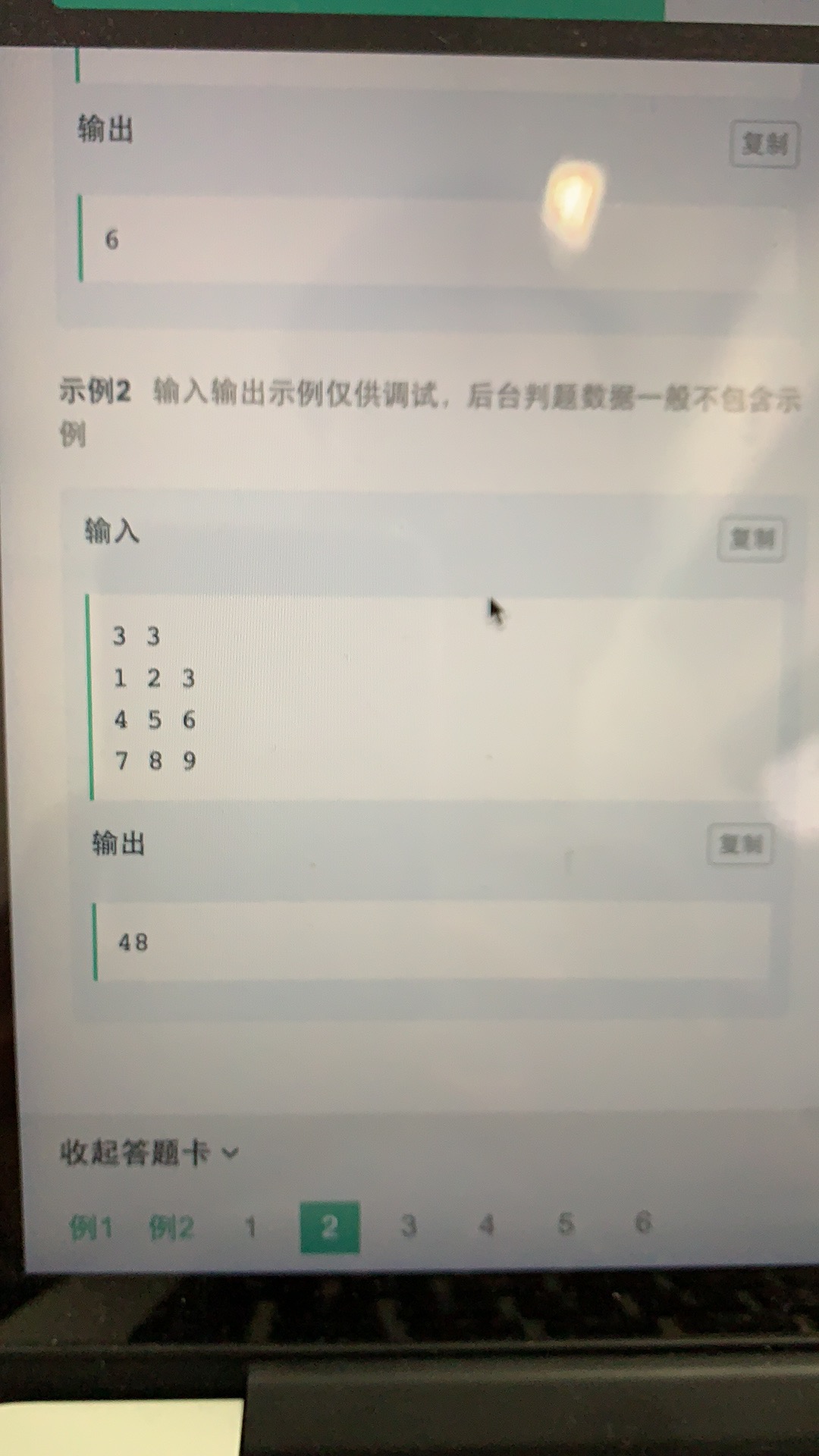
1、



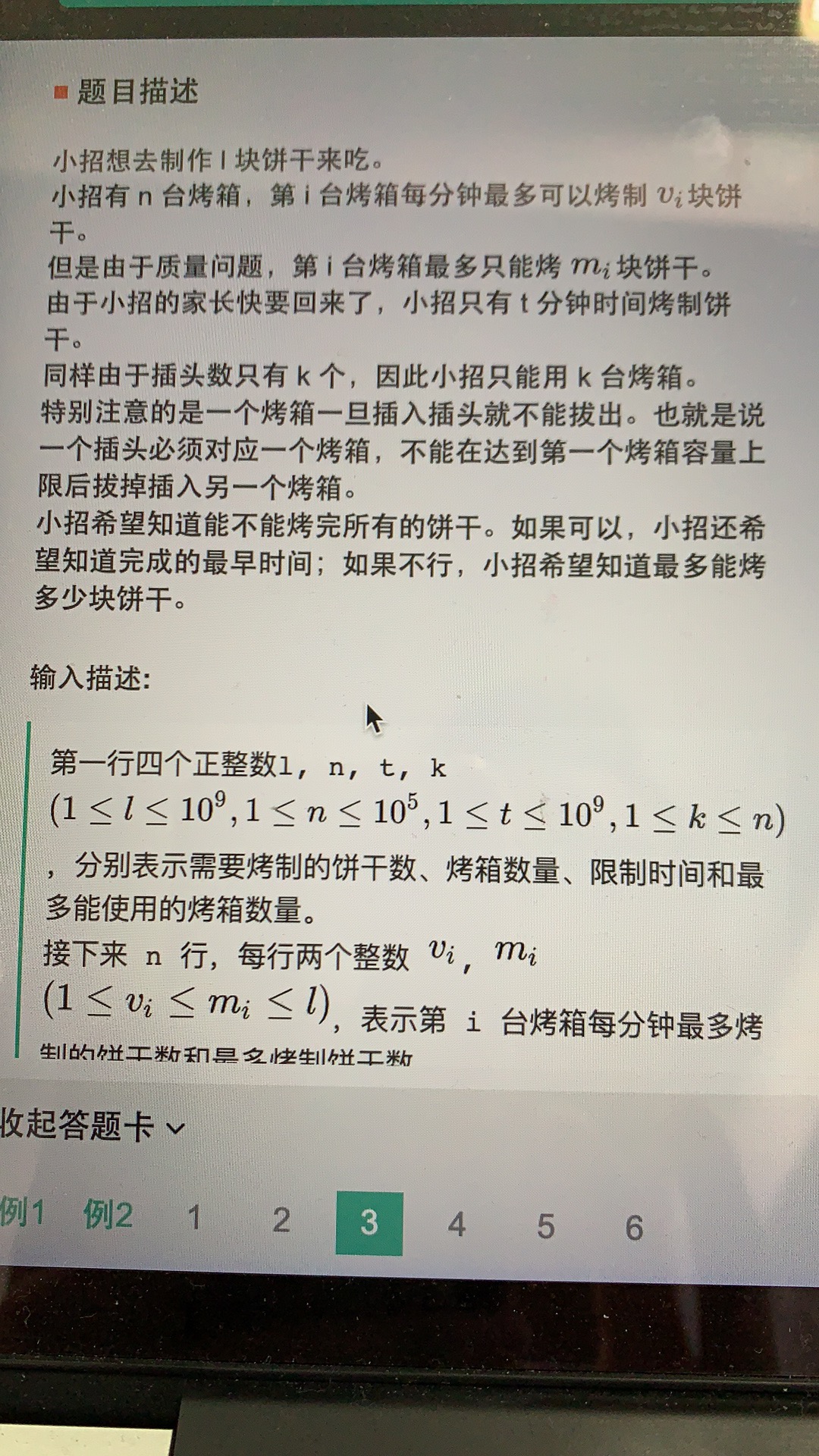


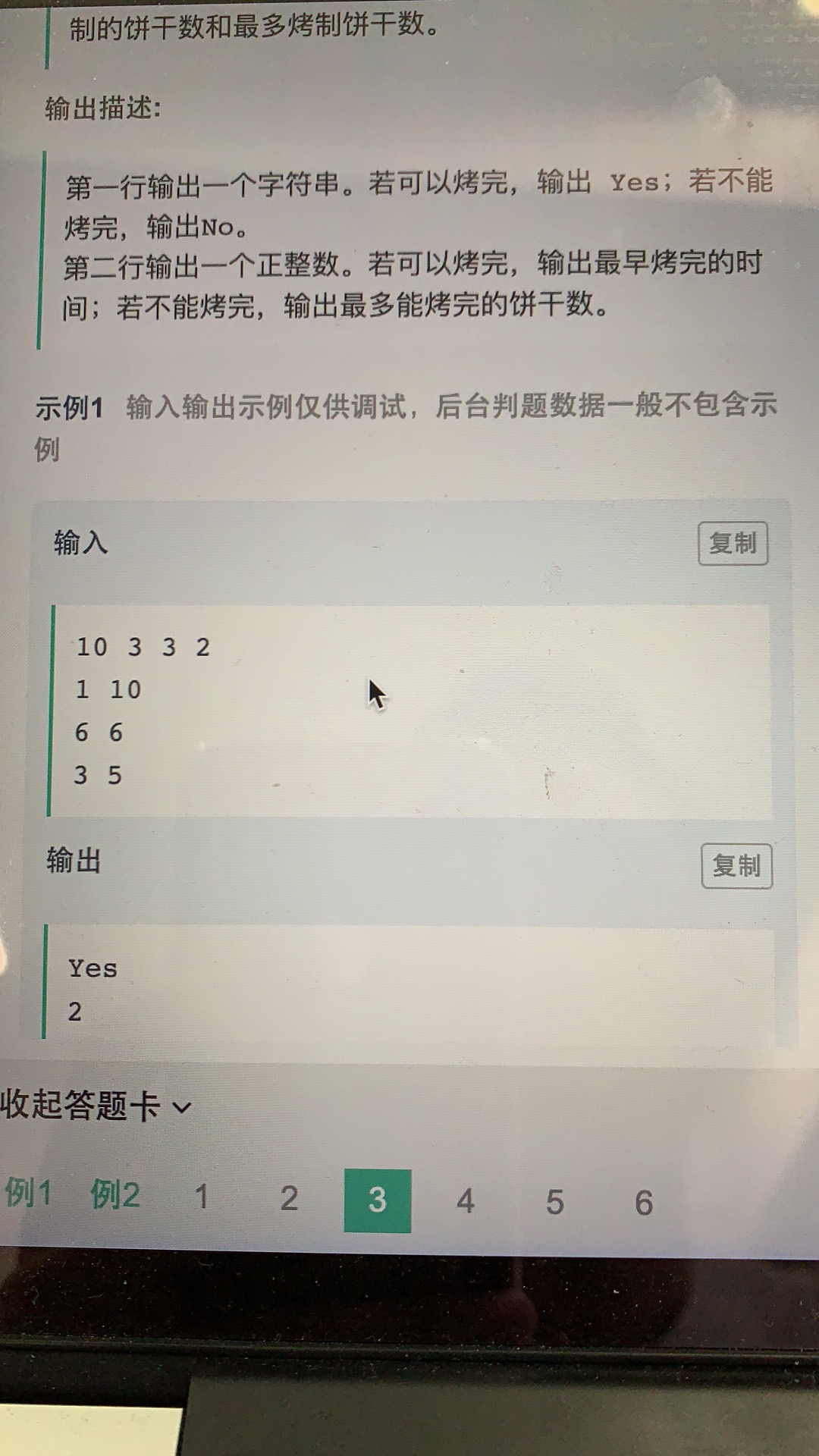
2、

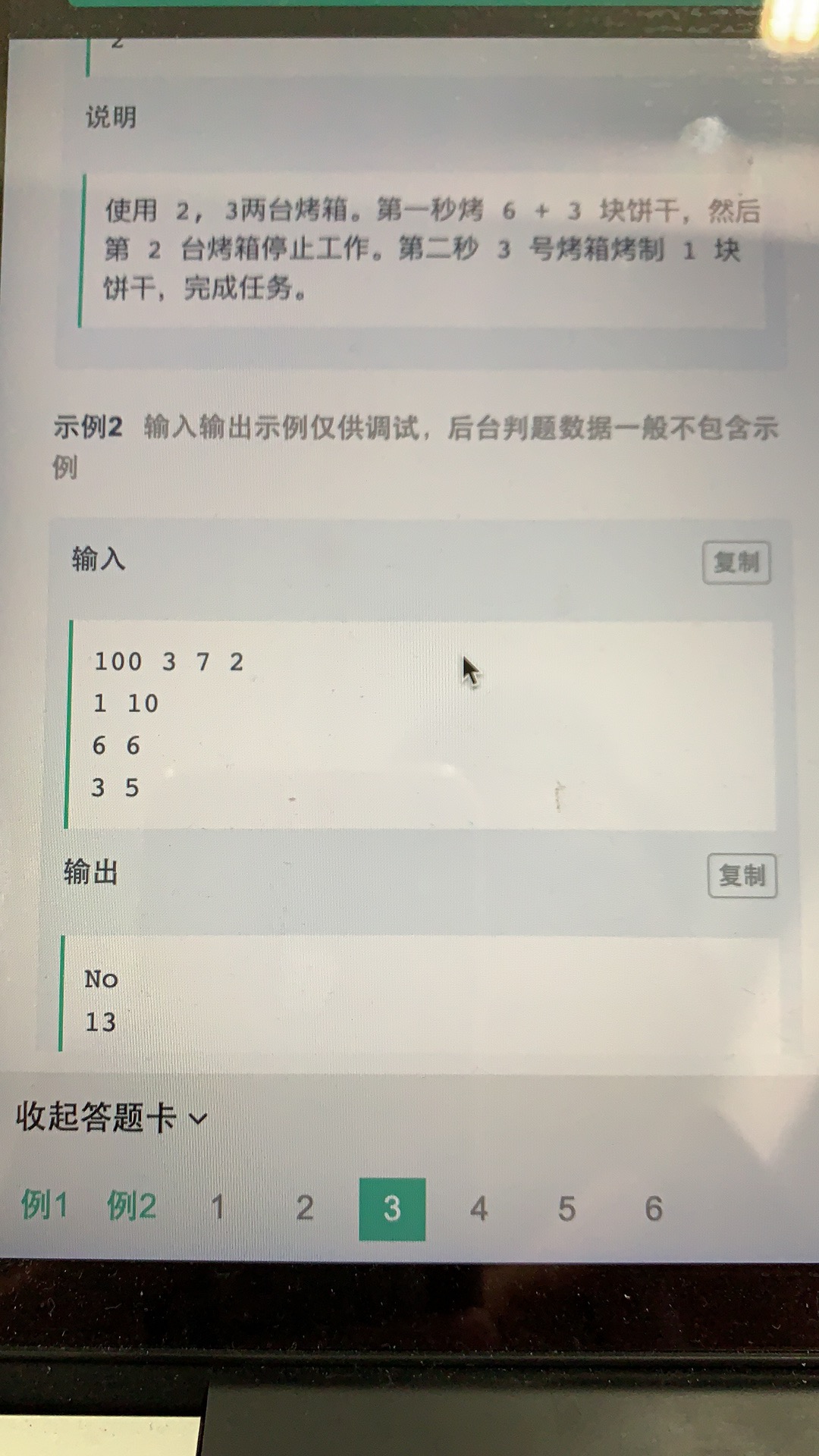




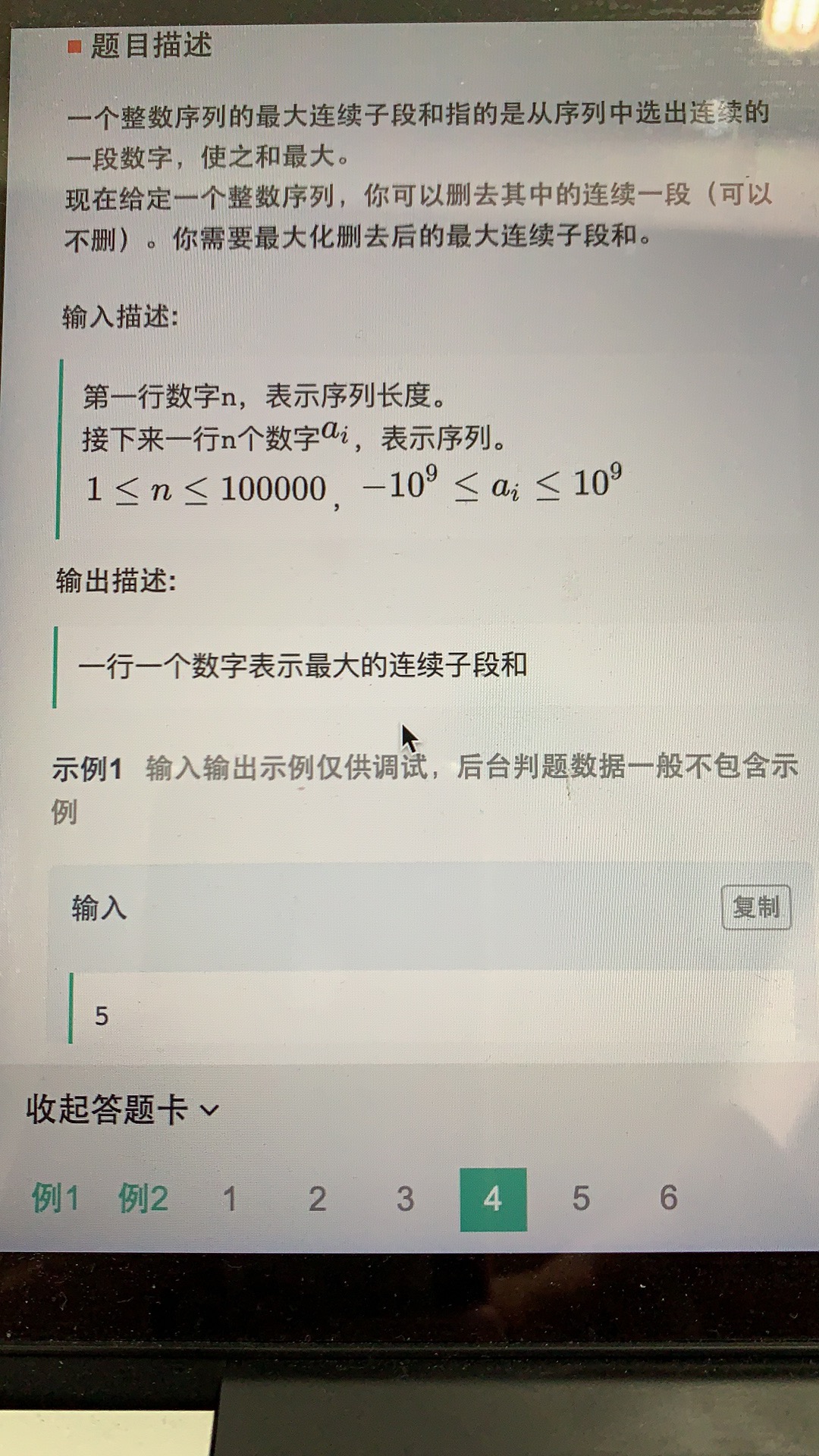
3、

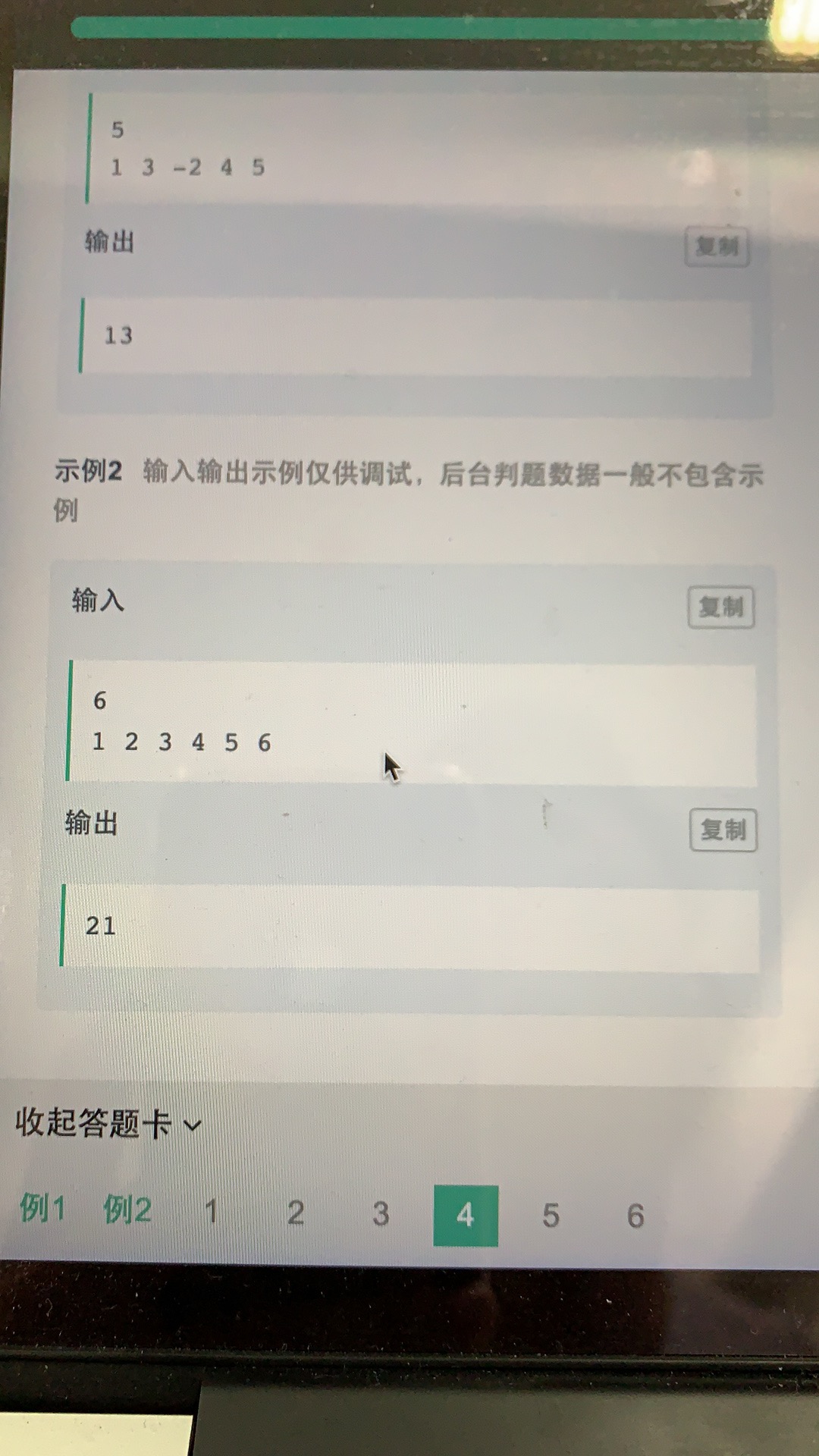




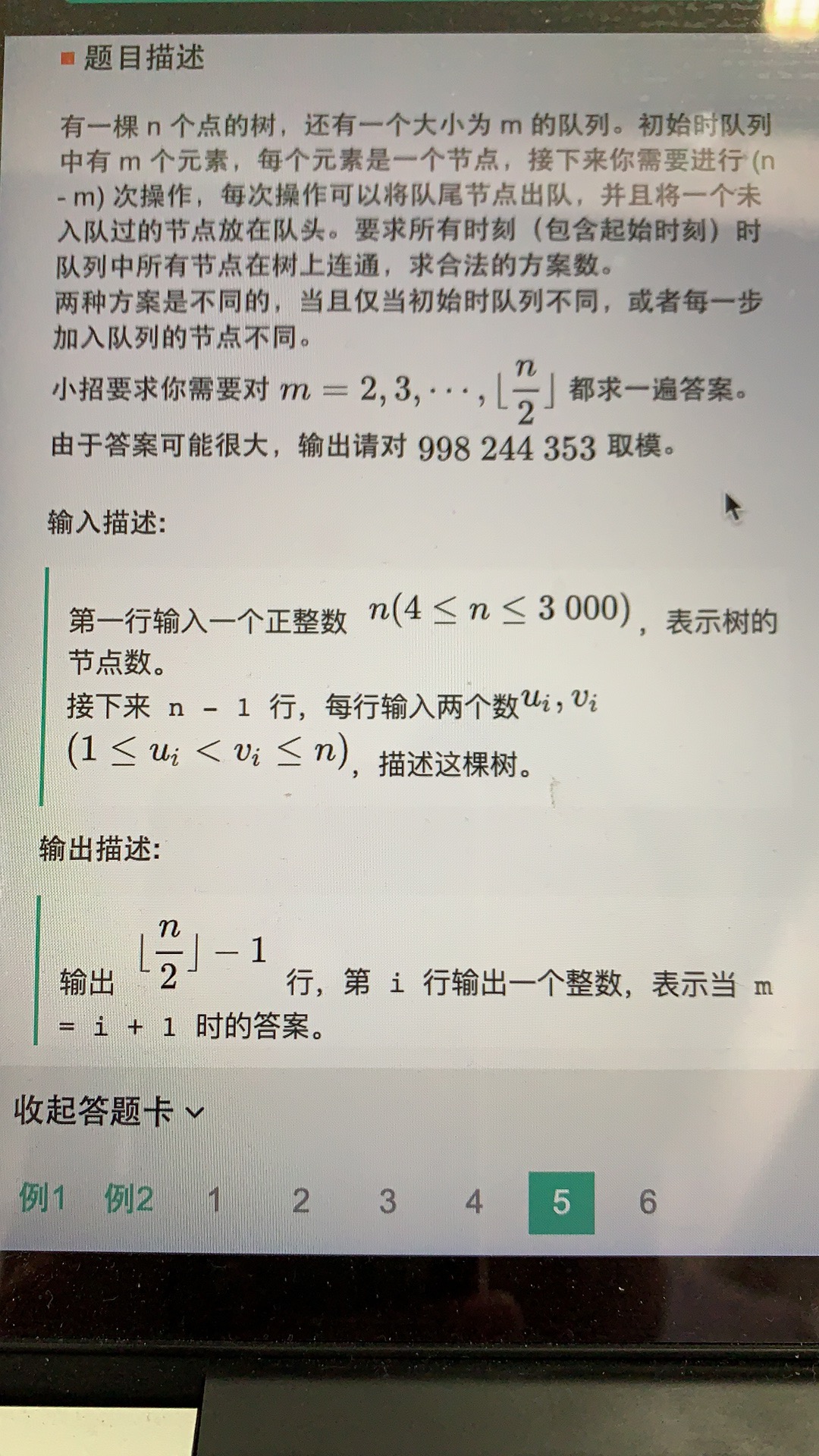


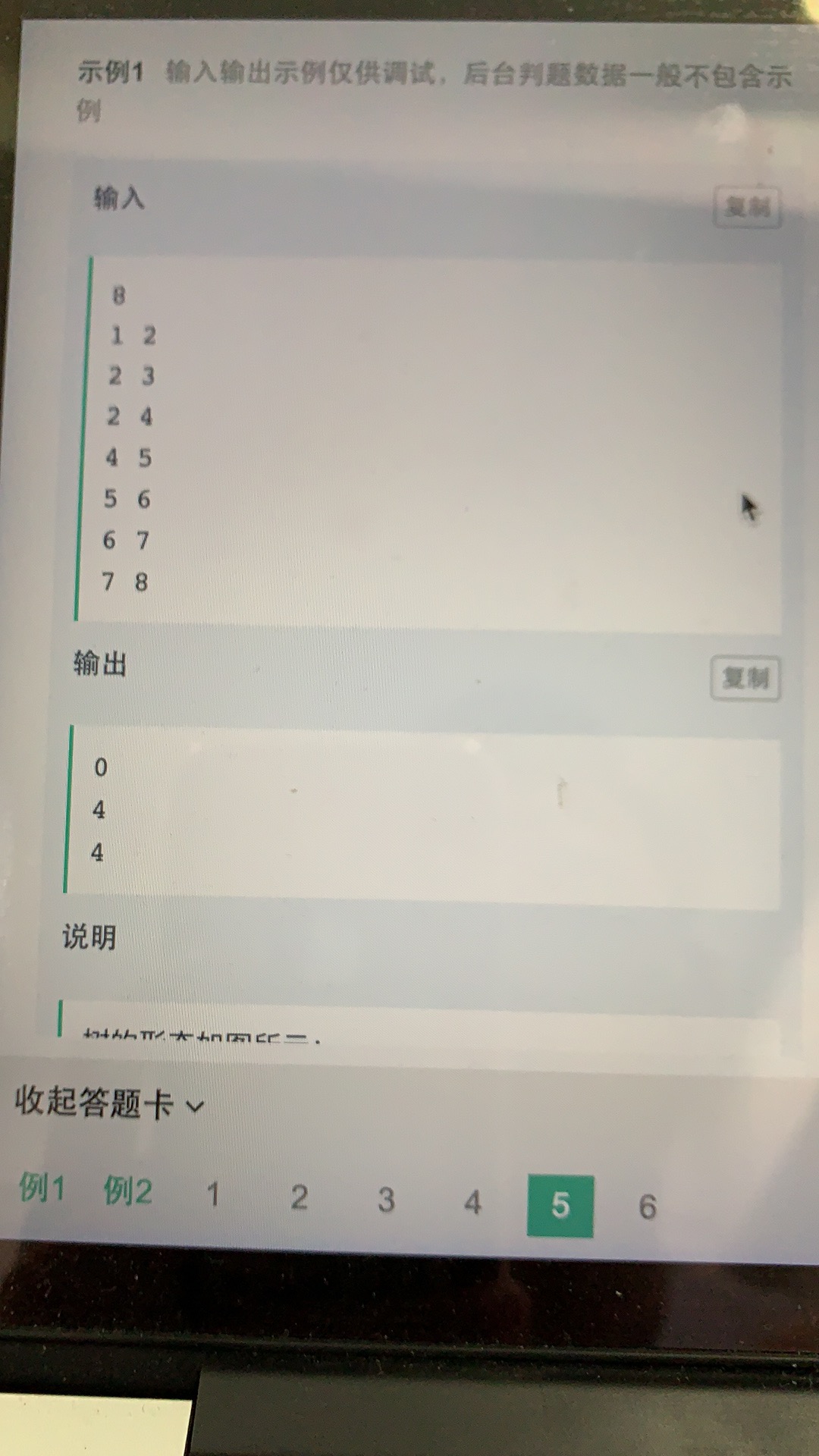
4、

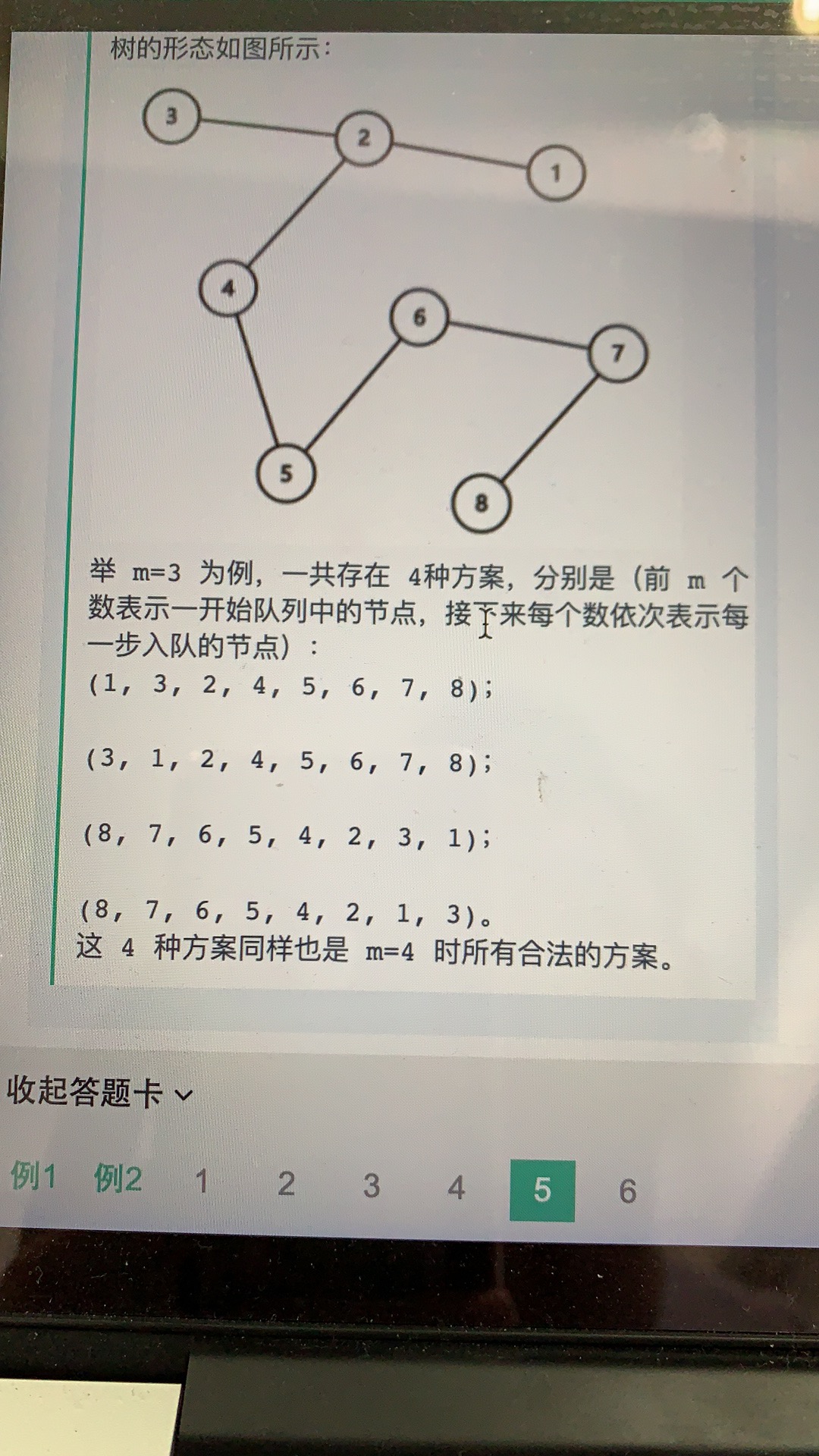




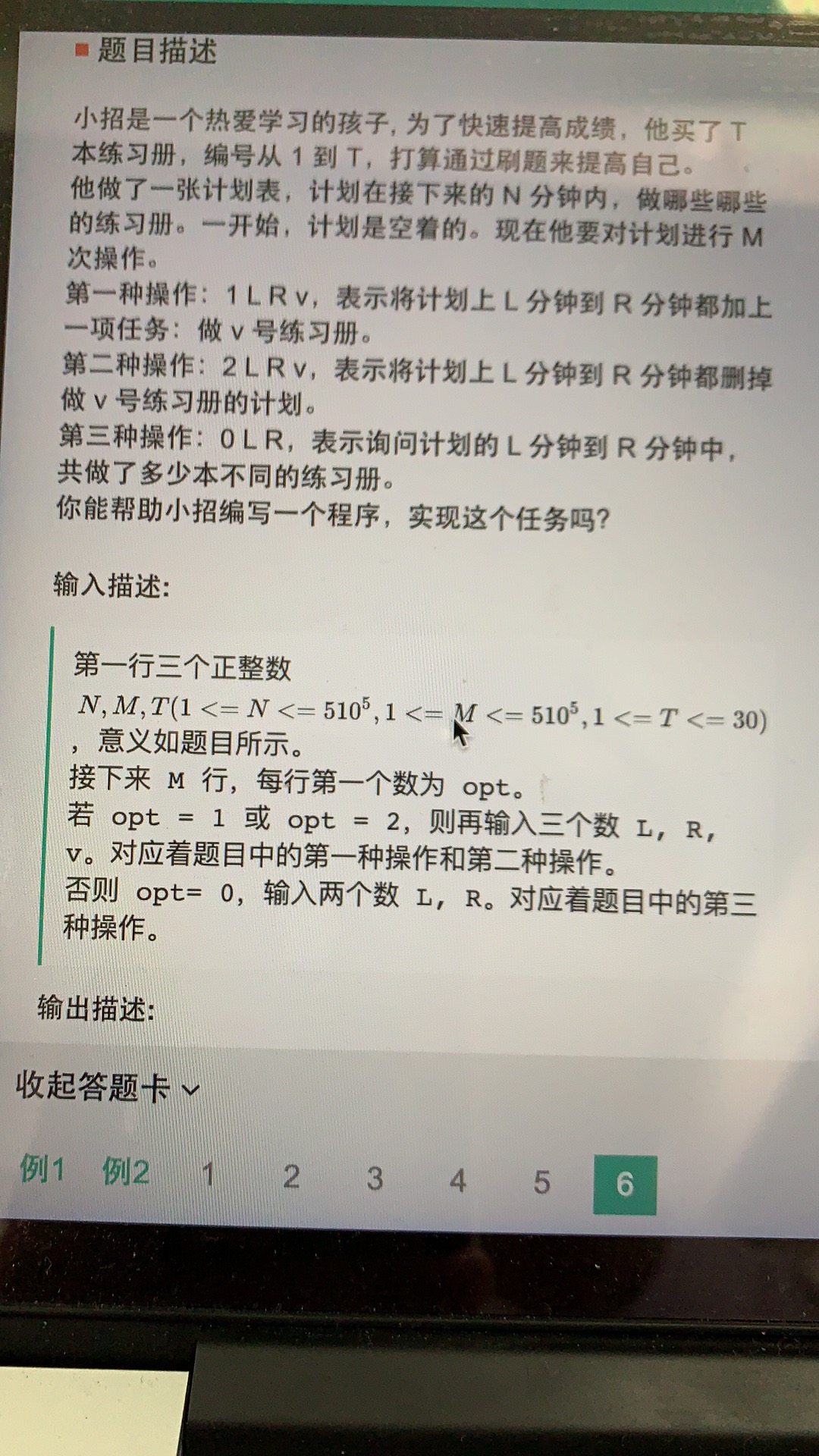
5、

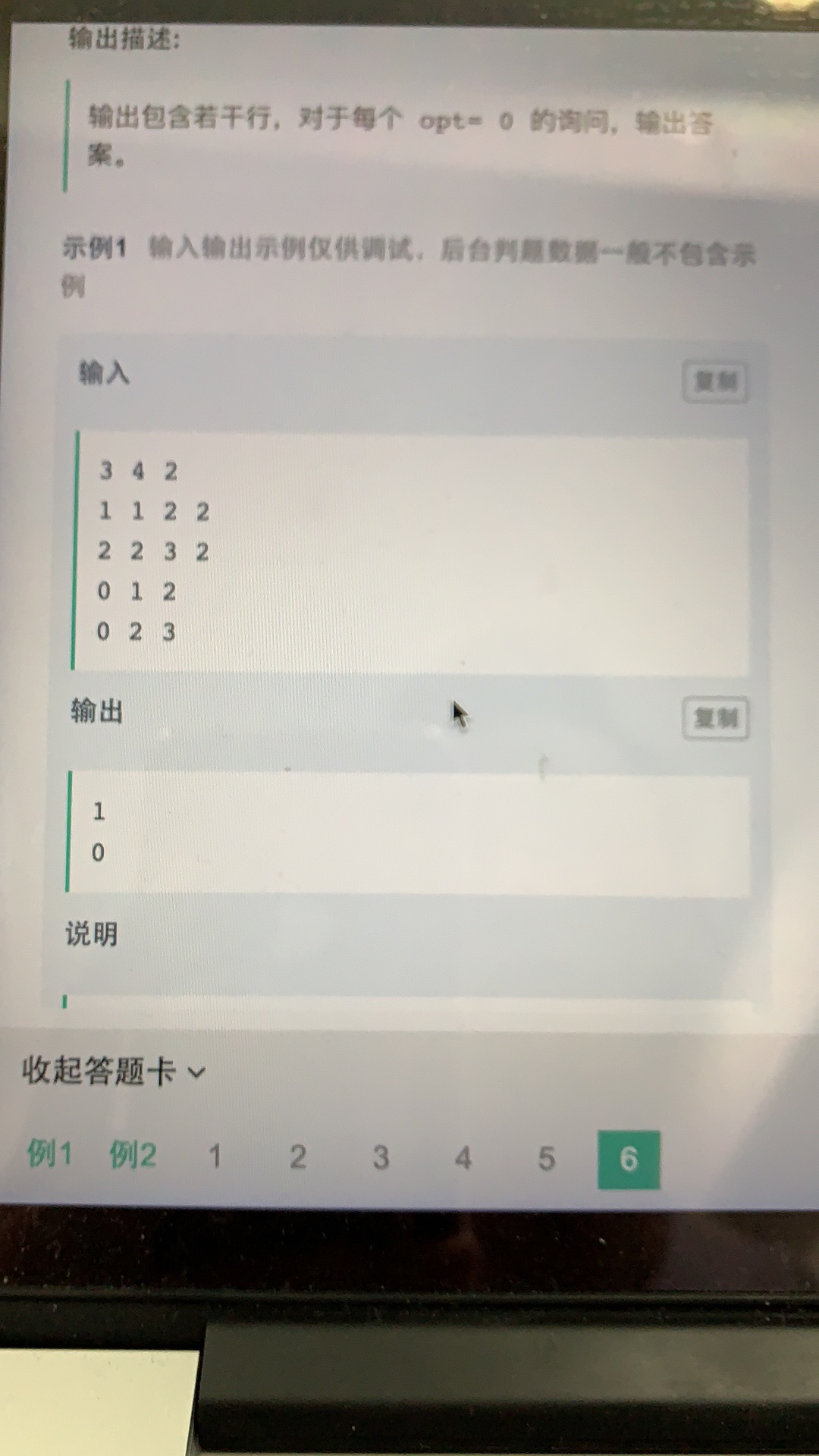


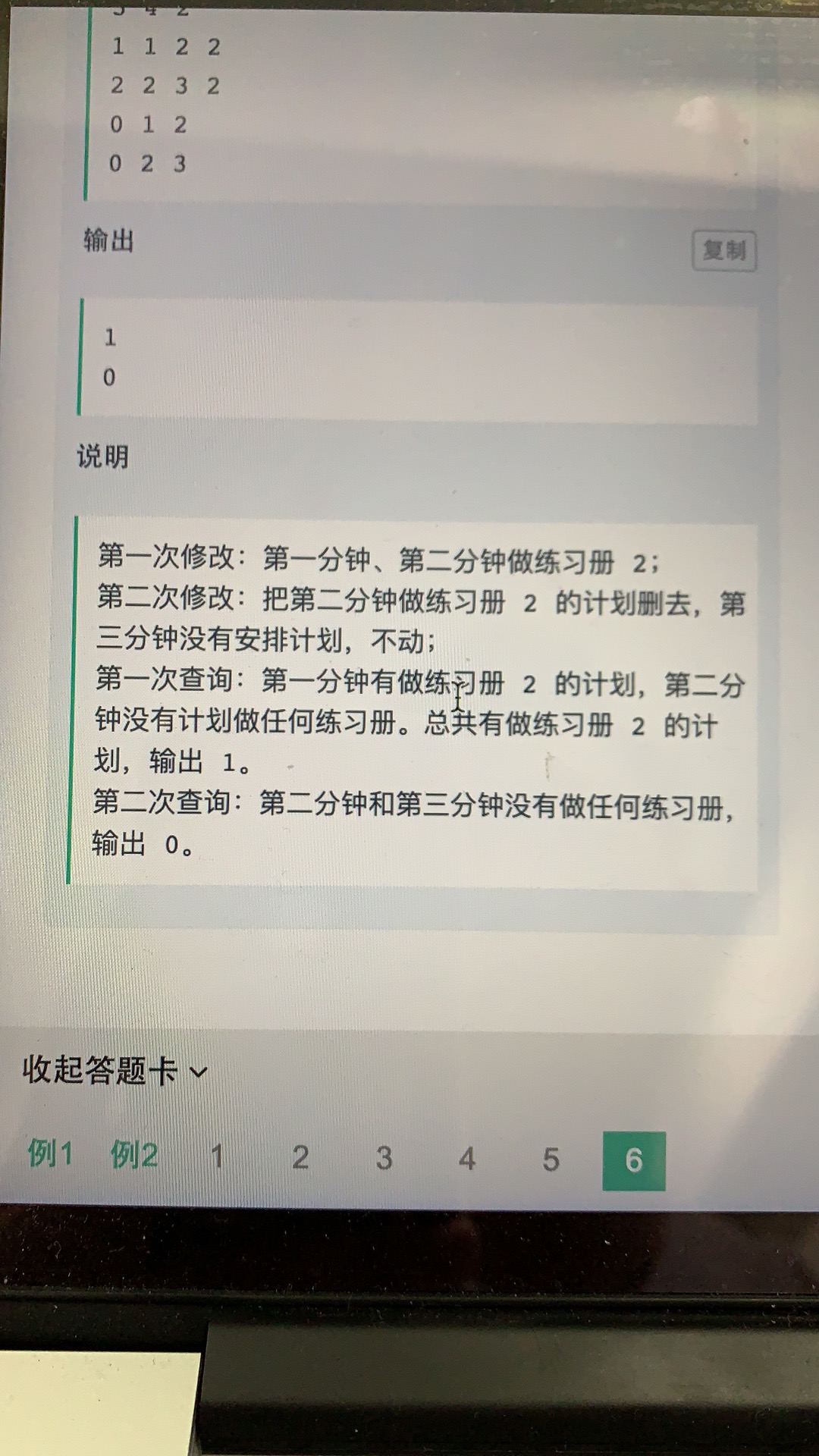




6、







提交答案

|  |
| --- |
| */\*\*  \* Copyright (c) 2017 XiaoMi Inc. All Rights Reserved.  \*  \** ***@author:*** *linzebin <linzebin@xiaomi.com>  \* Created on 2019/7/21  \*/* // 本题为考试多行输入输出规范示例，无需提交，不计分。  import java.util.Arrays; import java.util.Comparator; import java.util.Scanner;  public class zhaoshangyinhang {  public static void main(String[] args) {  *fun1*(); // fun2(); // fun3(); // fun4(); // fun5(); // fun6();  }    */\*\*\*  \* ac  \* 时间复杂度:排序 -> O(n\*logn)  \* 输入:  4  3 7 2 1  \*/* public static void fun1() {  Scanner sc = new Scanner(System.*in*);  int size = sc.nextInt();  if (size <= 0) {  *print*(0);  return;  }   int[] nums = new int[size];  for(int i = 0 ;i < size;i++) {  nums[i] = sc.nextInt();  }  if (size == 1) {  *print*(nums[0]);  return;  }   Arrays.*sort*(nums);   double result = nums[0];  for(int i = 1 ;i < size;i++) {  result =(nums[i] + result) /2.0;  }  *print*(result);  }  public static void print(double result){  System.*out*.println(String.*format*("%.4f", result));;  }   //60%  public static void fun2() {  Scanner sc = new Scanner(System.*in*);  int n=sc.nextInt();  int m=sc.nextInt();  long [] a= new long[n\*m+1];  for(int i=0;i<n\*m;i++) {  a[i]=sc.nextLong();  }  long sum=0;  for(int i=0;i<n\*m;i++) {  for(int j=i+1;j<n\*m;j++) {  if(((long)(i/m)!=(long)(j/m))&&(Math.*abs*(i-j)%m!=0)&&i!=j) {  sum=Math.*max*(sum, a[i]\*a[j]);  }  }  }  System.*out*.println(sum);   }   public static class Kaoxiang{  long v;  long m;  long mr;   public Kaoxiang(long v, long m, long mr) {  this.v = v;  this.m = m;  this.mr = mr;  }  }   public static class KaoxiangComparator implements Comparator<Kaoxiang> {   @Override  public int compare(Kaoxiang o1, Kaoxiang o2) {  long temp = o2.mr - o1.mr;  if(temp > 0)  return 1;  else if( temp == 0){  return 0;  }else{  return -1;  }  }   }   //18.8%   */\*\*  \*  \* 第三题: 小明要在t分钟之内做l张饼，有n个锅，但只能选其中k个锅，每个锅每分钟能做vi个饼，最多能做mi个饼，  \* 问能不能做完l张饼，如果能，输出最少需要多少分钟；如果不能，输出最多能做几张饼。  \* 解法：   \* 先讨论能不能做完：每个锅在t分钟内能做的饼数为min(mi,vi\*t), 降序排列，前k个锅能做出来的饼>l就能；   \* 如果不能做完：直接输出前k个锅能做饼的和；  \* 如果能：二分最短时间，然后判断在mid分钟内能不能做完饼，判断方法同t分钟的情况  \*/* public static void fun3() {   Scanner in =new Scanner(System.*in*);  long l=in.nextInt();  int n=in.nextInt();  long t=in.nextInt();  int k=in.nextInt();   Kaoxiang[] kaoxiangs = new Kaoxiang[n];  long v =0, m =0;  for(int i=0;i<n;i++){  v=in.nextLong();  m=in.nextLong();  long mr = Math.*min*(v \* t,m);  Kaoxiang kaoxiang = new Kaoxiang(v, m,mr);  kaoxiangs[i] = kaoxiang;  }   Arrays.*sort*(kaoxiangs, new KaoxiangComparator());    long sum=0;  for (long e = 0; e < t; e++) {  for (int j = 0; j < kaoxiangs.length && j < k; j++) {  if((kaoxiangs[j].m - (e \* kaoxiangs[j].v)) >= kaoxiangs[j].v)  sum += kaoxiangs[j].v;  else if( (kaoxiangs[j].m - (e \* kaoxiangs[j].v)) >= 0)  sum += (kaoxiangs[j].m - (e \* kaoxiangs[j].v));  }  if(sum >= l) {  System.*out*.println("Yes");  System.*out*.println(e+1);  return;  }  }   System.*out*.println("No");  System.*out*.println(sum);   } // public static void fun4() { // Scanner in =new Scanner(System.in); // int n=in.nextInt(); // int[] array=new int[n]; // int sum=0; // for(int i=0;i<array.length;i++){ // array[i]=in.nextInt(); // sum+=array[i]; // } // System.out.println(sum-core(array)); // }   // 20%  public static void fun4() {  Scanner sc=new Scanner(System.*in*);  while(sc.hasNext()){  int n=sc.nextInt();  if(n<=0)  System.*out*.println(0);  else{  int []nums=new int[n];  for(int i=0;i<n;i++){  nums[i]=sc.nextInt();  }  System.*out*.println(*Soulustion*(nums));  }  }  }   public static int Soulustion(int []nums){  int []sum=new int[nums.length];  int s=0;  int flag=0;  int count0=0;  for(int i=0;i<nums.length;i++){  s+=nums[i];  sum[i]=s;  if(nums[i]<0){  flag=1;  count0++;  }  }  if(flag==0)return sum[nums.length-1];  if(count0==nums.length){  Arrays.*sort*(nums);  return nums[nums.length-1];  }  int max=Integer.*MIN\_VALUE*;  for(int i=1;i<nums.length;i++){  for(int j=i-1;j>=0;j--){  int temp=sum[i]-sum[j];  int count=0;  for(int k=j+1;k<=i;k++){  if(nums[k]<0)count+=nums[k];  }  if(temp-count>max)max=temp-count;  }  }  for(int i=1;i<nums.length;i++){  int temp=sum[i];  for(int k=0;k<i;k++){  if(nums[k]<0)temp-=nums[k];  }  if(max<temp)max=temp;  }  return max;  }   public static int core(int[] array){  int res=0;  int min=Integer.*MAX\_VALUE*;  for(int i=0;i<array.length;i++){  if(res+array[i]<0){  res+=array[i];  }else  res=0;  if(res<min)  min=res;  }  return min;  }   public static void fun5() {   }  public static void fun6() {   } } |