（1）

class Result {

/\*

\* Complete the 'maxLCS' function below.

\*

\* The function is expected to return an INTEGER.

\* The function accepts STRING s as parameter.

\*/

public static int maxLCS(String s) {

// Write your code here

Map<Character, Integer> rightMap = new HashMap<>();//char, freq

for(char c : s.toCharArray()) {

rightMap.put(c, rightMap.getOrDefault(c, 0)+1);

}

int leftend = 0;

int common = 0;

int maxcom = 0;

Map<Character, Integer> leftMap = new HashMap<>();//char, freq

while(leftend < s.length()) {

char c = s.charAt(leftend);

if(leftMap.containsKey(c)) common-=Math.min(leftMap.get(c), rightMap.get(c));

leftMap.put(c, leftMap.getOrDefault(c,0)+1);

if(rightMap.get(c) == 1)

rightMap.remove(c);

else

rightMap.put(c,rightMap.get(c)-1);

if(rightMap.containsKey(c) && leftMap.containsKey(c))

common+=Math.min(leftMap.get(c), rightMap.get(c));

if(common > maxcom) {

maxcom = common;

}

leftend++;

}

return maxcom;

}

}

（2）

class Result {

/\*

\* Complete the 'strangeSort' function below.

\*

\* The function is expected to return a STRING\_ARRAY.

\* The function accepts following parameters:

\* 1. INTEGER\_ARRAY mapping

\* 2. STRING\_ARRAY nums

\*/

public class sortObj{

String ori;

String real;

int pos;

sortObj(String s, int p){

this.ori = s;

this.pos = p;

}

}

public String[] help(int[] mapping, String nums[]) {

if(nums == null || nums.length == 0) return new String[0];

List<sortObj> ls = new ArrayList<>();

HashMap<Character, Character> map = new HashMap<>(); //wrong -> real

for(int i : mapping) {

map.put( (char)('0'+mapping[i]), (char)('0'+i) );

}

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

sortObj obj = new sortObj(nums[i],i);

StringBuilder sb = new StringBuilder();

boolean zerostart = true;

for(char c : nums[i].toCharArray()) {

char correctc = map.get(c);

if(zerostart && Character.getNumericValue(correctc) != 0)

zerostart = false;

if(!zerostart)

sb.append(correctc);

}

obj.real = sb.toString();

ls.add(obj);

}

Collections.sort(ls, new Comparator<sortObj>() {

public int compare(sortObj a, sortObj b) {

if(a.real.length() == b.real.length())

return a.real.compareTo(b.real);

return a.real.length() - b.real.length();

}

});

String[] res = new String[ls.size()];

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

res[i] = ls.get(i).ori;

}

return res;

}

public List<String> ListToString(String[] str){

return Arrays.asList(str);

}

public String[] StringToList(List<String> list){

return list.toArray(new String[0]);

}

public int[] IntToList(List<Integer> list){

return list.stream().mapToInt(i -> i).toArray();

}

public static List<String> strangeSort(List<Integer> mapping, List<String> nums) {

// Write your code here

Result res = new Result();

String[] help = res.help(res.IntToList(mapping),res.StringToList(nums));

return res.ListToString(help);

}

}