**1、数串**

**题目描述**

设有n个正整数，将他们连接成一排，组成一个最大的多位整数。  
如:n=3时，3个整数13,312,343,连成的最大整数为34331213。  
如:n=4时,4个整数7,13,4,246连接成的最大整数为7424613。

**输入描述:**

有多组测试样例，每组测试样例包含两行，第一行为一个整数N（N<=100），第二行包含N个数(每个数不超过1000，空格分开)。

**输出描述:**

每组数据输出一个表示最大的整数。

**示例1**

输入

[复制](javascript:void(0);)

2

12 123

4

7 13 4 246

输出

[复制](javascript:void(0);)

12312

7424613

import sys

for i, v in enumerate(sys.stdin.readlines()):

if i % 2 == 1:

v = v.strip().split()

v.sort(cmp=lambda x, y: cmp(x + y, y + x), reverse=True)

print('0' if v[0] == '0'else "".join(v))

if \_\_name\_\_ == '\_\_main\_\_':

n = raw\_input()

arr = raw\_input().split()

print ''.join(sorted(arr, lambda x, y : -cmp(x+y, y+x)))

import sys

from functools import cmp\_to\_key

def mycmp(x, y):

if x + y > y + x: return 1

elif x == y: return 0

else: return -1

for i, v in enumerate(sys.stdin.readlines()):

if i % 2 == 1:

v = v.strip().split()

v.sort(key=cmp\_to\_key(mycmp), reverse=True)

print("".join(v).lstrip("0") or "0")

#include <iostream>

#include <algorithm>

using namespace std;

bool campare(string i ,string j){

return (i+j)>(j+i);

}

int main(){

int n;

while(cin>>n){

vector<string>temp(n,"");

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

cin>>temp[i];

}

sort(temp.begin(),temp.end(),campare);

for(int i = 0 ; i < n;++i)

cout<<temp[i];

cout<<endl;

}

return 0;

}

#include <string>

#include <algorithm>

#include <vector>

#include <iostream>

using namespace std;

int main() {

int n;

cin >> n;

vector<string> sArray;

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

string temp;

cin >> temp;

sArray.push\_back(temp);

}

sort(sArray.begin(), sArray.end(), [](string s1, string s2){

return (s1+s2) > (s2+s1);

});

for (auto s:sArray) {

cout << s;

}

return 0;

}

**2、树的高度**

**题目描述**

现在有一棵合法的二叉树，树的节点都是用数字表示，现在给定这棵树上所有的父子关系，求这棵树的高度

**输入描述:**

输入的第一行表示节点的个数n（1 ≤ n ≤ 1000，节点的编号为0到n-1）组成，

下面是n-1行，每行有两个整数，第一个数表示父节点的编号，第二个数表示子节点的编号

**输出描述:**

输出树的高度，为一个整数

**示例1**

输入

[复制](javascript:void(0);)

5

0 1

0 2

1 3

1 4

输出

[复制](javascript:void(0);)

3

链接：<https://www.nowcoder.com/questionTerminal/4faa2d4849fa4627aa6d32a2e50b5b25>  
来源：牛客网  
  
不需要用到map啊，声明一个数组存储深度，child位置的值为parent位置的值加一，然后维护全局最大值即可。但是这样做通不过全部测试的原因是测试用例中有多叉树，引入第二个数组存储该节点的子节点数量，大于2不做深度判断即可。

**二叉树/多叉树/层次遍历**

import java.util.Scanner;

public class Main{

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int n = in.nextInt();

if(n<3){

System.out.println(n);

}

else{

int[] height = new int [n];

int[] binary = new int[n];

height[0] = 1;

int max = 0;

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

int parent = in.nextInt();

int child = in.nextInt();

binary[parent] += 1;

if(binary[parent] < 3){

height[child] = height[parent]+1;

}

max = Math.max(max, height[child]);

}

System.out.println(max);

}

}

}

n = input()

tree = [1] \* n

childnum = [0] \* n

for i in xrange(n-1):

parent, this = map(int, raw\_input().split(" "))

if childnum[parent] >= 2:

tree[this] = 0

childnum[this] = 2

continue

tree[this] += tree[parent]

childnum[parent] += 1

print max(tree)

**3、句子反转**

**题目描述**

给定一个句子（只包含字母和空格）， 将句子中的单词位置反转，单词用空格分割, 单词之间只有一个空格，前后没有空格。 比如： （1） “hello xiao mi”-> “mi xiao hello”

**输入描述:**

输入数据有多组，每组占一行，包含一个句子(句子长度小于1000个字符)

**输出描述:**

对于每个测试示例，要求输出句子中单词反转后形成的句子

**示例1**

输入

[复制](javascript:void(0);)

hello xiao mi

输出

[复制](javascript:void(0);)

mi xiao hello

import sys

print(" ".join(sys.stdin.readline().strip().split(" ")[::-1]))

#include<iostream>

#include<stack>

#include<string>

using namespace std;

int main(){

string s;

stack<string> sstack;

while(cin>>s){

sstack.push(s);

}

while(sstack.size()!=1){

cout<<sstack.top()<<" ";

sstack.pop();

}

cout<<sstack.top()<<endl;

return 0;

}

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

int main() {

string words;

while (getline(cin, words)) {

reverse(words.begin(), words.end());

unsigned i = 0, j = i;

while (i < words.size()) {

while (i < words.size() && words[i] != ' ')

++i;

reverse(words.begin() + j, words.begin() + i);

j = ++i;

}

cout << words << endl;

}

return 0;

}

**4、电话号码分身**

**题目描述**

继MIUI8推出手机分身功能之后，MIUI9计划推出一个电话号码分身的功能：首先将电话号码中的每个数字加上8取个位，然后使用对应的大写字母代替 （"ZERO", "ONE", "TWO", "THREE", "FOUR", "FIVE", "SIX", "SEVEN", "EIGHT", "NINE"）， 然后随机打乱这些字母，所生成的字符串即为电话号码对应的分身。

**输入描述:**

第一行是一个整数T（1 ≤ T ≤ 100)表示测试样例数；接下来T行，每行给定一个分身后的电话号码的分身（长度在3到10000之间）。

**输出描述:**

输出T行，分别对应输入中每行字符串对应的分身前的最小电话号码（允许前导0）。

**示例1**

输入

[复制](javascript:void(0);)

4

EIGHT

ZEROTWOONE

OHWETENRTEO

OHEWTIEGTHENRTEO

输出

[复制](javascript:void(0);)

0

234

345

0345

#include <iostream>

#include <string.h>

#include <vector>

#include <algorithm>

using namespace std;

int main(){

int i = 0, j = 0;

int n;

while (cin >> n){

string s;

for (i = 0; i < n; i++){

cin >> s;

vector<int> iimap(256);

for (j = 0; j < s.size(); j++){

iimap[s[j]]++;

}

vector<int> res(10);

int count = iimap['U'];

res[6] = count;

iimap['F'] -= count;

iimap['O'] -= count;

iimap['U'] -= count;

iimap['R'] -= count;

count = iimap['X'];

res[8] = count;

iimap['S'] -= count;

iimap['I'] -= count;

iimap['X'] -= count;

count = iimap['W'];

res[4] = count;

iimap['T'] -= count;

iimap['W'] -= count;

iimap['O'] -= count;

count = iimap['G'];

res[0] = count;

iimap['E'] -= count;

iimap['I'] -= count;

iimap['G'] -= count;

iimap['H'] -= count;

iimap['T'] -= count;

count = iimap['Z'];

res[2] = count;

iimap['Z'] -= count;

iimap['E'] -= count;

iimap['R'] -= count;

iimap['O'] -= count;

count = iimap['F'];

res[7] = count;

iimap['F'] -= count;

iimap['I'] -= count;

iimap['V'] -= count;

iimap['E'] -= count;

count = iimap['V'];

res[9] = count;

iimap['S'] -= count;

iimap['E'] -= count;

iimap['V'] -= count;

iimap['E'] -= count;

iimap['N'] -= count;

count = iimap['T'];

res[5] = count;

iimap['T'] -= count;

iimap['H'] -= count;

iimap['R'] -= count;

iimap['E'] -= count;

iimap['E'] -= count;

count = iimap['O'];

res[3] = count;

iimap['O'] -= count;

iimap['N'] -= count;

iimap['E'] -= count;

count = iimap['I'];

res[1] = count;

iimap['N'] -= count;

iimap['I'] -= count;

iimap['N'] -= count;

iimap['E'] -= count;

for (int k = 0; k < res.size(); k++){

for (int l = 0; l < res[k]; l++)

cout << k;

}

cout << endl;

}

}

return 0;

}

/\*

Z:0; W:2;

U:4; X:6;

G:8; O:0, 1, 2, 4;

R:0, 3, 4; F:4, 5;

S:6, 7; I:5, 6, 8, 9

字母-代表数字-加8前表示数字

G-8-0, I-9-1, Z-0-2, O-1-3, W-2-4, R-3-5, U-4-6, F-5-7, X-6-8, S-7-9

\*/

#include <iostream>

using namespace std;

int main(){

int T;

char mas[10] = {'G', 'I', 'Z', 'O', 'W', 'R', 'U', 'F', 'X', 'S'};

while(cin >> T){

for(int i = 0; i < T; i ++){

string tel;

cin >> tel;

int count[10] ={0};

for(int j = 0; j < tel.length(); j++){

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

if(tel[j] == mas[z]) count[z]++;

}

}

count[3] = count[3] - count[2] - count[4] - count[6];

count[5] = count[5] - count[2] - count[6];

count[7] = count[7] - count[6];

count[9] = count[9] - count[8];

count[1] = count[1] - count[7] - count[8] - count[0];

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

for(int j = 0; j < count[i]; j++){

cout << i;

}

}

cout << endl;

}

}

return 0;

}

import sys

def originalDigits(s):

result = [0] \* 10

result[0] = s.count("Z")

result[2] = s.count("W")

result[4] = s.count("U")

result[6] = s.count("X")

result[7] = s.count("S") - result[6]

result[5] = s.count("V") - result[7]

result[1] = s.count("O") - result[0] - result[4] - result[2]

result[9] = (s.count("N") - result[1] - result[7]) // 2

result[8] = s.count("I") - result[5] - result[6] - result[9]

result[3] = s.count("H") - result[8]

t, resStr = "", ""

for i, r in enumerate(result):

t += r \* str(i)

for i in t:

if int(i) >= 8:

resStr += str(int(i) - 8)

else:

resStr += str(int(i) + 10 - 8)

return "".join(sorted(resStr))

for i in sys.stdin.readlines():

if i[0].isalpha():

print(originalDigits(i.strip()))

**5、水仙花数**

**题目描述**

春天是鲜花的季节，水仙花就是其中最迷人的代表，数学上有个水仙花数，他是这样定义的： “水仙花数”是指一个三位数，它的各位数字的立方和等于其本身，比如：153=1^3+5^3+3^3。 现在要求输出所有在m和n范围内的水仙花数。

**输入描述:**

输入数据有多组，每组占一行，包括两个整数m和n（100 ≤ m ≤ n ≤ 999）。

**输出描述:**

对于每个测试实例，要求输出所有在给定范围内的水仙花数，就是说，输出的水仙花数必须大于等于m,并且小于等于n，如果有多个，则要求从小到大排列在一行内输出，之间用一个空格隔开;

如果给定的范围内不存在水仙花数，则输出no;

每个测试实例的输出占一行。

**示例1**

输入

[复制](javascript:void(0);)

100 120

300 380

输出

[复制](javascript:void(0);)

no

370 371