****最长广播效应**** ****【2021 H2, 2022 Q1,Q2 考试题】****

时间限制：C/C++ 1秒，其他语言 2秒

空间限制：C/C++262144K，其他语言524288K

64bit IO Format：%lld

****本题可使用本地IDE编码，不能使用本地已有代码，无跳出限制，****

****编码后请点击”保存并调试“按钮进行代码提交。****

****■ 题目描述****

* 某通信网络中有N个网络结点，用1到N进行标识。
* 网络中的结点互联互通，且结点之间的消息传递有时延，相连结点的时延均为一个时间单位。
* 现给定网络结点的连接关系link[i]={u，v}，其中u和v表示网络结点。
* 当指定一个结点向其他结点进行广播，所有被广播结点收到消息后都会在原路径上回复一条响应消息，请计算发送结点至少需要等待几个时间单位才能收到所有被广播结点的响应消息。

****注：****

1. N的取值范围为[1，100];
2. 连接关系link的长度不超过3000，且1 <= u,v <= N;
3. 网络中任意结点间均是可达的;

****输入描述：****

* 输入的第一行为两个正整数，分别表示网络结点的个数N，以及时延列表的长度T；
* 接下来的T行输入，表示结点间的连接关系列表；
* 最后一行的输入为一个正整数，表示指定的广播结点序号；

****输出描述：****

* 输出一个整数，表示发送结点接收到所有响应消息至少需要等待的时长。

import java.util.Scanner;

public class Main {

private static int[][] arr;

private static int[] dis;

private static boolean[] mark;

private static int from;

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String[] nms = sc.nextLine().split(" ");

int count = Integer.parseInt(nms[0]);

int line = Integer.parseInt(nms[1]);

arr = new int[count + 1][count + 1];

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

String[] dis = sc.nextLine().split(" ");

int x1 = Integer.parseInt(dis[0]);

int y1 = Integer.parseInt(dis[1]);

arr[x1][y1] = 1;

}

from = Integer.parseInt(sc.nextLine());

dis = new int[count + 1];

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

if (i == from) {

dis[i] = 0;

} else {

dis[i] = Integer.MAX\_VALUE;

}

}

mark = new boolean[count + 1];

calcTls(from, count);

}

private static void calcTls(int local, int count) {

mark[local] = true;

for (int i = 1; i <= count; i++) {

if (arr[local][i] != 0) {

dis[i] = arr[local][i];

}

}

int used = 0;

while (used < count) {

int tempLocal = 0;

int distMin = Integer.MAX\_VALUE;

for (int i = 1; i <= count; i++) {

if (!mark[i] && dis[i] < distMin) {

distMin = dis[i];

tempLocal = i;

}

}

local = tempLocal;

mark[local] = true;

for (int i = 1; i <= count; i++) {

if (arr[local][i] != 0) {

dis[i] = Math.min(dis[local] + arr[local][i], dis[i]);

}

}

used++;

}

int maxDis = Integer.MIN\_VALUE;

for (int i = 1; i <= count; i++) {

maxDis = Math.max(maxDis, dis[i]);

}

System.out.println(maxDis << 1);

}

}

****考古问题**** ****【2021 H2, 2022 Q1,Q2 考试题】****

时间限制：C/C++ 1秒，其他语言 2秒

空间限制：C/C++262144K，其他语言524288K

64bit IO Format：%lld

****本题可使用本地IDE编码，不能使用本地已有代码，无跳出限制，****

****编码后请点击”保存并调试“按钮进行代码提交。****

****■ 题目描述****

考古问题，假设以前的石碑被打碎成了很多块，每块上面都有一个或若干个字符，请你写个程序来把之前石碑上文字可能的组合全部写出来，按升序进行排列。

****示例1****   输入输出示例仅供调试，后台判题数据一般不包含示例

****输入****

3

a b c

****输出****

abc

acb

bac

bca

cab

cba

****示例2****   输入输出示例仅供调试，后台判题数据一般不包含示例

****输入****

3

a b a

****输出****

aab

aba

baa

import java.util.HashSet;

import java.util.Scanner;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = Integer.parseInt(scanner.nextLine());

String str = scanner.nextLine();

String[] strings = str.split(" ");

ArrayList<String> items = new ArrayList<>();

Collections.addAll(items, strings);

HashSet<String> hashSet = new HashSet<>();

dynamicCombine(items, new StringBuilder(), hashSet);

List<String> list = new ArrayList(strings.length);

list.addAll(hashSet);

list.sort((Comparator.comparing(o -> o)));

for (String s : list) {

System.out.println(s);

}

}

public static void dynamicCombine(ArrayList<String> items, StringBuilder builder, HashSet<String> ret)

{

if (items.size() != 0) {

for (int i = 0, len = items.size(); i < len; i++) {

StringBuilder newBuilder = new StringBuilder(builder);

newBuilder.append(items.get(i));

ArrayList<String> newItems = new ArrayList<>();

for (int j = 0; j < len; j++) {

newItems.add(j, items.get(j));

}

newItems.remove(i);

dynamicCombine(newItems, newBuilder, ret);

}

} else {

if (builder.length() > 0) {

ret.add(builder.toString());

}

}

}

}

****贪吃蛇****【2021, 2022H1 考试题】

时间限制：C/C++ 1秒，其他语言 2秒

空间限制：C/C++262144K，其他语言524288K

64bit IO Format：%lld

本题可使用本地IDE编码，不能使用本地已有代码，无跳出限制，

编码后请点击”保存并调试”按钮进行代码提交。

**SnakeGame【贪吃蛇】**

****■ 题目描述****

贪吃蛇是一个经典游戏，蛇的身体由若干方格连接而成，身体随蛇头移动。蛇头触碰到食物时，蛇的长度会增加一格。

蛇头和身体的任一方格或者游戏版图边界碰撞时，游戏结束。

下面让我们来完成贪吃蛇游戏的模拟。

给定一个N\*M的数组arr，代表N\*M个方格组成的版图，贪吃蛇每次移动一个方格。

若arr[i][j] == ‘H’，表示该方格为贪吃蛇的起始位置；

若arr[i][j] == ‘F’，表示该方格为食物，

若arr[i][j] == ‘E’，表示该方格为空格。

贪吃蛇初始长度为1，初始移动方向为向左。

为给定一系列贪吃蛇的移动操作，返回操作后蛇的长度，如果在操作执行完之前已经游戏结束，返回游戏结束时蛇的长度。

贪吃蛇移动、吃食物和碰撞处理的细节见下面图示：

图1：截取了贪吃蛇移动的一个中间状态，H表示蛇头，F表示食物，数字为蛇身体各节的编号，蛇为向左移动，此时蛇头和食物已经相邻

图2：蛇头向左移动一格，蛇头和食物重叠，注意此时食物的格子成为了新的蛇头，第1节身体移动到蛇头位置，第2节身体移动到第1节身体位置，以此类推，

最后添加第4节身体到原来第3节身体的位置。

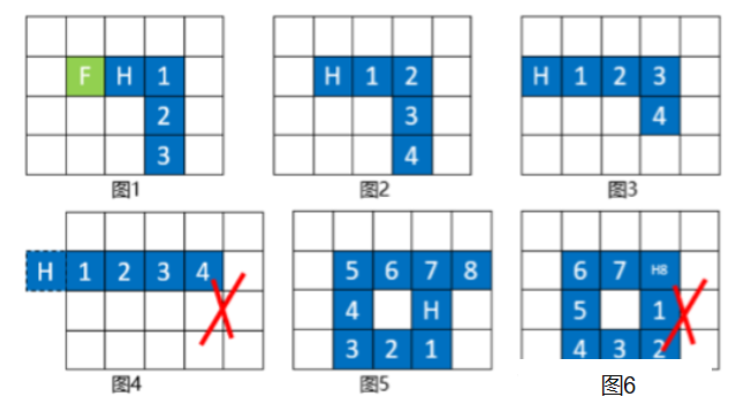
图3：蛇头继续向左移动一格，身体的各节按上述规则移动，此时蛇头已经和边界相邻，但还未碰撞。

图4：蛇头继续向左移动一格，此时蛇头已经超过边界，发生碰撞，游戏结束。

图5和图6给出一个蛇头和身体碰撞的例子，蛇为向上移动。

图5时蛇头和第7节身体相邻，但还未碰撞；

图6蛇头向上移动一格，此时蛇头和第8节身体都移动到了原来第7节身体的位置，发生碰撞，游戏结束。



****输入描述****

* 输入第一行为空格分隔的字母，代表贪吃蛇的移动操作。
* 字母取值为U、D、L、R和G，
* U、D、L、R分别表示贪吃蛇往上、下、左、右和转向，转向时贪吃蛇不移动 ，G表示贪吃蛇按当前的方向移动一格。
* 用例保证输入的操作正确。
* 第二行为空格分隔的两个数，指定N和M，为数组的行和列数。
* 余下N行每行是空格分隔的M个字母。字母取值为H、F和E，H表示贪吃蛇的起始位置，F表示食物，E表示该方格为空。
* 用例保证有且只有一个H，而F和E会有多个。

****输出描述****

* 输出一个数字，为蛇的长度。

****示例 1****输入输出示例仅供调试，后台判题数据一般不包含示例

****输入****

D G G

3 3

F F F

F F H

E F E

****输出****

1

**地图表示为：**

蛇头 H(Head)

食物 F(Food)

E表示该方格为空

**四个方向分别表示为：**

向上 U(up)

向下 D(down)

向左 L(Left)

向右 R(Right)

最大社交距离

****【编程题目 |200分】最大社交距离**** ****【2021 H2,2022 Q1,Q2 考试题】****

时间限制：C/C++ 1秒，其他语言 2秒

空间限制：C/C++262144K，其他语言524288K

64bit IO Format：%lld

本题可使用本地IDE编码，不能使用本地已有代码，无跳出限制，

编码后请点击”保存并调试“按钮进行代码提交。

****■ 题目描述****

疫情期间需要大家保证一定的社交距离，公司组织开交流会议。座位一排共 N 个座位，编号分别为[0,N-1],

要求员工一个接着一个进入会议室，并且可以在任何时候离开会议室。

满足：

每当一个员工进入时，需要坐到最大社交距离（最大化自己和其他人的距离的座位）；

如果有多个这样的座位，则坐到索引最小的那个座位。

****输入描述****

会议室座位总数seatNum。(1 <= seatNum <= 500)

员工的进出顺序 seatOrLeave 数组，元素值为 1，表示进场；元素值为负数，表示出场（特殊：位置 0 的员工不会离开）。

例如 -4 表示坐在位置 4 的员工离开（保证有员工坐在该座位上）

****输出描述****

最后进来员工，他会坐在第几个位置，如果位置已满，则输出-1。

****示例1****

****输入****

10

[1,1,1,1,-4,1]

****输出****

5

****说明****

* seat -> 0,空在任何位置都行，但是要给他安排索引最小的位置，也就是座位 0
* seat -> 9,要和旁边的人距离最远，也就是座位 9
* seat -> 4,要和旁边的人距离最远，应该坐到中间，也就是座位 4
* seat -> 2,员工最后坐在 2 号座位上
* leave[4], 4 号座位的员工离开
* seat -> 5,员工最后坐在 5 号座位上

import java.util.Scanner;

import java.util.TreeSet;

public class MaxSocialDistance {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int seatNum = in.nextInt();

in.nextLine();

String seat = in.nextLine();

String[] c = seat.substring(1, seat.length() - 1).split(",");

int[] seatOrLeave = new int[c.length];

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

seatOrLeave[i] = Integer.parseInt(c[i]);

}

int ans = seatDistance(seatNum, seatOrLeave);

System.out.print(ans);

}

public static int seatDistance(int seatNum, int[] seatOrLeave) {

TreeSet<Integer> seatedNums = new TreeSet<>();

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

int op = seatOrLeave[i];

if (op > 0) {

if (seatedNums.size() == 0) {

if (i == seatOrLeave.length - 1) {

return 0;

}

seatedNums.add(0);

} else if (seatedNums.size() == 1) {

seatedNums.add(seatNum - 1);

if (i == seatOrLeave.length - 1) {

return seatNum - 1;

}

} else if (seatedNums.size() < seatNum) {

int[] ints = new int[seatedNums.size()];

int count = 0;

for (Integer seatedNum : seatedNums) {

ints[count++] = seatedNum;

}

int maxLen = 0;

int start = 0;

for (int j = 0; j < ints.length - 1; j++) {

int len = ints[j + 1] - ints[j];

if (len / 2 > maxLen) {

maxLen = len / 2;

start = ints[j];

}

}

seatedNums.add(start + maxLen);

if (i == seatOrLeave.length - 1) {

return start + maxLen;

}

} else {

return -1;

}

} else {

seatedNums.remove(-op);

}

}

return 0;

}

}

【编程题目 | 200分】叠积木 [ 2022 Q1 考试题 ]

时间限制:C/C++ 1000ms | 其他语言:2000ms

空间限制:C/C++ 262144K | 其他语言:524288K

64bit IO Format:%lld

本题可使用本地IDE编码，不能使用本地已有代码。无跳出限制，

编码后请点击”保存并提交“按钮进行代码提交。

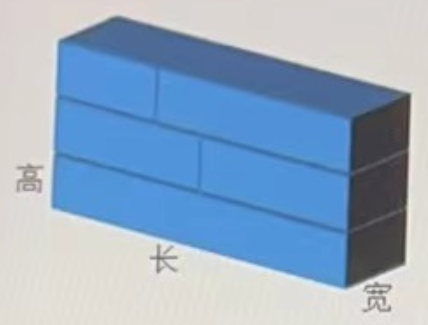
****■ 题目描述****

有一堆长方体积木，它们的长度和宽度都相同，但长度不一。

小橙想把这堆积木叠成一面墙，墙的每层可以放一个积木，也可以将两个积木拼接起来，要求每层的长度相同。

若必须用完这些积木，叠成的墙最多为多少层？

如下是叠成的一面墙的图示，积木仅按宽和高所在的面进行拼接。



****输入描述：****

输入为一行，为各个积木的长度，数字为正整数，并由空格分隔。积木的数量和长度都不超过5000。

****输出描述：****

输出一个数字，为墙的最大层数，如果无法按要求叠成每层长度一致的墙，则输出-1。

****输入****

给定积木的长度，以空格分隔，例如:3 6 6 3。

****输出****

如果可以搭建，返回最大层数，如果不可以返回-1。

****示例1****输入输出示例仅供调试，后台判题数据一般不包含示例

****输入****

3 6 3 3 3

****输出****

3

****解释：****以 6 为底的墙，第一层为 6 ，第二层为 3 + 3，第三层 3 + 3。

****示例2****输入输出示例仅供调试，后台判题数据一般不包含示例

****输入****

9 9 9 5 3 2 2 2 2 2

****输出****

5

****解释：****

5+2+2=9

3+2+2+2=9

9,9,9

共五层

import java.util.Arrays;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String str = scanner.nextLine();

String[] s = str.split(" ");

int[] nums = new int[s.length];

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

nums[i] = Integer.parseInt(s[i]);

}

int n = nums.length;

if (n == 1) {

System.out.println(1);

}

if (n == 0) {

System.out.println(-1);

}

Arrays.sort(nums);

int maxVal = nums[n - 1];

int sum = 0;

for (int num : nums) {

sum += num;

}

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

if (sum % maxVal == 0) {

break;

} else {

maxVal += nums[i];

}

}

int cnt = sum / maxVal;

sum /= cnt;

boolean[][] dp = new boolean[n + 1][sum + 1];

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

dp[i][0] = true;

}

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

for (int j = 1; j <= sum; j++) {

if (j - nums[i - 1] < 0) {

dp[i][j] = dp[i - 1][j];

} else {

dp[i][j] = dp[i - 1][j] || dp[i - 1][j - nums[i - 1]];

}

}

}

if (dp[n][sum] && cnt != 1) {

System.out.println(cnt);

} else {

System.out.println(-1);

}

}

}

****统计文本数量 【编程 | 200分】****

****题目描述****

有一个文件，包含以一定规则写作的文本，请统计文件中包含的文本数量。

规则如下：

1. 文本以”;”分隔，最后一条可以没有”;”，但空文本不能算语句，比如”COMMAND A; ;”只能算一条语句

注意，无字符/空白字符/制表符都算作”空”文本；

2. 文本可以跨行，比如下面，是一条文本，而不是三条；

COMMAND A

AND

COMMAND B;

3. 文本支持字符串，字符串为成对的单引号(‘)或者成对的双引号(“)，字符串可能出现用转义字符(\)处理的单双引号(“your input is\””)和转义字符本身，比如

COMMAND A “Say \”hello\””;

4. 支持注释，可以出现在字符串之外的任意位置注释以”–“开头，到换行结束，比如

**隐藏内容**

COMMAND A; –this is comment

COMMAND –comment

A AND COMMAND B;

注意字符串内的”–“，不是注释。

****输入描述：****

文本文件

****输出描述：****

包含的文本数量

****示例1****

****输入****

COMMAND TABLE IF EXISTS "UNITED STATE";

COMMAND A GREAT (

ID ADSAB,

download\_length INTE-GER, -- test

file\_name TEXT,

guid TEXT,

mime\_type TEXT,

notifica-tionid INTEGER,

original\_file\_name TEXT,

pause\_reason\_type INTERGER,

resumable\_flag INTERGER,

start\_time INTERGER,

state INTERGER,

folder TEXT,

path TEXT,

total\_length INTE-GER,

url TEXT

);

****输出****

2

代码实现

**隐藏内容**

#include <bits/stdc++.h>

using namespace std;

int main(int argc, const char \*argv[])

{

string str;

bool f1 = **false**, f2 = **false**;

bool f = **false**;

int ans = 0;

**while** (getline(cin, str)) {

int l = str.size();

**for** (int i = 0; i < l; i++) {

**if** (str[i] != ' ' && str[i] != ';' && str[i] != '\t') f = **true**;

**if** (str[i] == '-') {

**if** (!f1 && !f2 && i != l - 1 && str[i + 1] == '-')

break;

} **else** **if** (!f1 && !f2 && str[i] == ';' && f) {

ans++;

f = **false**;

}

**else** **if** (str[i] == '\"' && i > 0 && str[i - 1] != '\\') {

f1 = f1 ^ **true**;

} **else** **if** (str[i] == '\'' && i > 0 && str[i - 1] != '\\' && !f1) {

f2 = f2 ^ **true**;

}

}

cout << ans << endl;

}

**return** 0;

}

【编程题目 |200分】报文解压缩【2022 Q2考试题】

时间限制：C/C++ 1秒，其他语言 2秒

空间限制：C/C++262144K，其他语言524288K

64bit IO Format：%lld

本题可使用本地IDE编码，不能使用本地已有代码，无跳出限制，

编码后请点击”保存并调试“按钮进行代码提交。

■****题目描述****

* 为了提升数据传输的效率，会对传输的报文进行压缩处理。
* 输入一个压缩后的报文，请返回它解压后的原始报文。
* 压缩规则：n[str]，表示方括号内部的 str 正好重复 n 次。
* 注意 n 为正整数（0 < n <= 100），str只包含小写英文字母，不考虑异常情况。

****输入描述:****

****输入压缩后的报文：****

1）不考虑无效的输入，报文没有额外的空格，方括号总是符合格式要求的；

2）原始报文不包含数字，所有的数字只表示重复的次数 n ，例如不会出现像 5b 或 3[8] 的输入；

****输出描述:****

* 解压后的原始报文

****注：****

1）原始报文长度不会超过1000，不考虑异常的情况

****示例 1****输入输出示例仅供调试，后台判题数据一般不包含示例

****输入****

3[k]2[mn]

****输出****

kkkmnmn

****说明****

* k 重复3次，mn 重复2次，最终得到 kkkmnmn

****示例2****输入输出示例仅供调试，后台判题数据一般不包含示例

****输入****

3[m2[c]]

****输出****

mccmccmcc

****说明****

* m2[c] 解压缩后为 mcc，重复三次为 mccmccmcc

import java.util.Deque;

import java.util.LinkedList;

import java.util.Objects;

import java.util.Scanner;

public class DecodeString {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String input = sc.nextLine();

System.out.println(decodeString(input));

}

public static String decodeString(String s) {

if (s == null || s.equals("")) return "";

StringBuilder temp = new StringBuilder();

StringBuilder now = new StringBuilder();

Deque<String> deque = new LinkedList<>();

int i = 0;

do {

now = new StringBuilder();

if (s.charAt(i) - '0' >= 0 && s.charAt(i) - '0' <= 9) {

while (s.charAt(i) - '0' >= 0 && s.charAt(i) - '0' <= 9) now.append(s.charAt(i++));

deque.push(now.toString());

deque.push("[");

i = i + 1;

while (true) {

while (s.charAt(i) != ']') {

now = new StringBuilder();

if (s.charAt(i) - '0' >= 0 && s.charAt(i) - '0' <= 9) {

while (s.charAt(i) - '0' >= 0 && s.charAt(i) - '0' <= 9) now.append(s.charAt(i++));

deque.push(now.toString());

} else {

deque.push(s.charAt(i++) + "");

}

}

i++;

now = new StringBuilder();

while (!Objects.equals(deque.peek(), "[")) {

now.insert(0, deque.pop());

}

deque.pop();

int count = Integer.parseInt(deque.pop());

StringBuilder s0 = new StringBuilder();

for (int j = 0; j < Math.max(0, count); j++) {

s0.append(now);

}

//s0.append(String.valueOf(now).repeat(Math.max(0, count)));

if (deque.isEmpty()) {

temp.append(s0);

break;

} else deque.push(s0.toString());

}

} else {

temp.append(s.charAt(i++));

}

} while (i < s.length());

return temp.toString();

}

}