**1.组合总和**

【问题描述】

给你一个 无重复元素 的整数数组 candidates 和一个目标整数 target ，找出 candidates 中可以使数字和为目标数 target 的 所有 不同组合 ，并以列表形式返回。你可以按 任意顺序 返回这些组合。

candidates 中的 同一个 数字可以 无限制重复被选取 。如果至少一个数字的被选数量不同，则两种组合是不同的。

对于给定的输入，保证和为 target 的不同组合数少于 150 个。

【示例1】

输入：candidates = [2,3,6,7], target = 7

输出：[[2,2,3],[7]]

解释：

2 和 3 可以形成一组候选，2 + 2 + 3 = 7 。注意 2 可以使用多次。

7 也是一个候选， 7 = 7 。

仅有这两种组合。

【示例1】

输入: candidates = [2,3,5], target = 8

输出: [[2,2,2,2],[2,3,3],[3,5]]

示例 3：

输入: candidates = [2], target = 1

输出: []

提示：

1 <= candidates.length <= 30

2 <= candidates[i] <= 40

candidates 的所有元素 互不相同

1 <= target <= 40

package com.my.algocode;

import java.util.\*;

public class Main01 {

    private static int[] candidates;

    private static List<List<Integer>> result = new ArrayList<>();

    private static List<Integer> list = new ArrayList<>();

    public static void main(String[] args) {

        Scanner input1 = new Scanner(System.in);

        String s = input1.nextLine();

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

        candidates = new int[numStr.length];

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

            candidates[i]= Integer.parseInt(numStr[i]);

        }

        Scanner input2 = new Scanner(System.in);

        int target = input2.nextInt();

        dfs(0,target,0);

        System.out.println(result);

    }

    private static void dfs(int t,int target,int sum){

        if(sum==target){

            for(int i=0;i<list.size();i++){

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

                for(Integer num:list){

                    temp.add(num);

                }

                if(result!=null){

                    for(List<Integer> l:result){

                        if(l.containsAll(temp)){

                           return;

                        }

                    }

                }

                result.add(temp);

            }

        }

        else {

            if(sum>target){

                return;

            }

            else {

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

                    list.add(candidates[i]);

                    sum+=candidates[i];

                    dfs(t+1,target,sum);

                    sum-=candidates[i];

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

                }

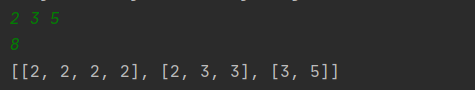
            }

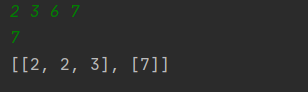
        }

    }

}

运行结果：







**2.变形课**

【问题描述】

呃……变形课上Harry碰到了一点小麻烦,因为他并不像Hermione那样能够记住所有的咒语而随意的将一个棒球变成刺猬什么的,但是他发现了变形咒语的一个统一规律:如果咒语是以a开头b结尾的一个单词,那么它的作用就恰好是使A物体变成B物体.

Harry已经将他所会的所有咒语都列成了一个表,他想让你帮忙计算一下他是否能完成老师的作业,将一个B(ball)变成一个M(Mouse),你知道,如果他自己不能完成的话,他就只好向Hermione请教,并且被迫听一大堆好好学习的道理。

【输入形式】

测试数据有多组。每组有多行，每行一个单词,仅包括小写字母,是Harry所会的所有咒语.数字0表示一组输入结束.

【输出形式】

如果Harry可以完成他的作业,就输出”Yes.”,否则就输出”No.”(不要忽略了句号)

【输入样例】

so  
soon  
river  
goes  
them  
got  
moon  
begin  
big  
0

【输出样例】

Yes.

package com.my.algocode;

import java.util.Arrays;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class Mian02 {

    private static char[][] map = new char[1000][2];

    private static int flag = 0;

    private static int[][] log = new int[1000][2];

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        String s = input.next();

        int k=0;

        while (!s.equals("0")){

            map[k][0]=s.charAt(0);

            map[k][1]=s.charAt(s.length()-1);

            k++;

            s=input.next();

        }

        dfs(0,'m','b',k);

        if(flag==1){

            System.out.println("YES");

        }

        else {

            System.out.println("NO");

        }

    }

    private static void dfs(int t,char target,char now,int k){

        if(flag==1){

            return;

        }

        if(now==target){

            flag=1;

        }

        else {

            for(int i=0;i<k;i++){

                if(map[i][0]==now&&log[i][0]!=1){

                    now=map[i][1];

                    log[i][0]=1;

                    dfs(t+1,target,now,k);

                    log[i][0]=0;

                    now=map[i][0];

                }

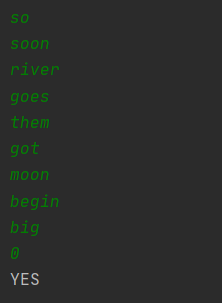
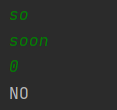
            }

        }

    }

}

结果：

**3.Tempter of the Bone**

【问题描述】

The doggie found a bone in an ancient maze, which fascinated him a lot. However, when he picked it up, the maze began to shake, and the doggie could feel the ground sinking. He realized that the bone was a trap, and he tried desperately to get out of this maze.  
  
The maze was a rectangle with sizes N by M. There was a door in the maze. At the beginning, the door was closed and it would open at the T-th second for a short period of time (less than 1 second). Therefore the doggie had to arrive at the door on exactly the T-th second. In every second, he could move one block to one of the upper, lower, left and right neighboring blocks. Once he entered a block, the ground of this block would start to sink and disappear in the next second. He could not stay at one block for more than one second, nor could he move into a visited block. Can the poor doggie survive? Please help him.

【输入形式】

The input consists of multiple test cases. The first line of each test case contains three integers N, M, and T (1 < N, M < 7; 0 < T < 50), which denote the sizes of the maze and the time at which the door will open, respectively. The next N lines give the maze layout, with each line containing M characters. A character is one of the following:  
  
'X': a block of wall, which the doggie cannot enter;  
'S': the start point of the doggie;  
'D': the Door; or  
'.': an empty block.  
  
The input is terminated with three 0's. This test case is not to be processed.

【输出形式】

For each test case, print in one line "YES" if the doggie can survive, or "NO" otherwise.

【输入样例】

4 4 5

S.X.

..X.

..XD

....

3 4 5

S.X.

..X.

...D

0 0 0

【输出样例】

NO

YES

package com.my.algocode;

import java.util.Arrays;

import java.util.Scanner;

public class MaxSum {

    private static char[][] map;

    private static int[][] log;

    private static int minTime = Integer.MAX\_VALUE;

    private static int[] goX = {0,0,-1,1};

    private static int[] goY = {1,-1,0,0};

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int n = input.nextInt();

        int m = input.nextInt();

        int t = input.nextInt();

        map = new char[n][m];

        log = new int[n][m];

        int x0=0;

        int y0=0;

        String s = "";

        for(int i=0;i<n;i++){

            s=input.next();

            for(int j=0;j<map[i].length;j++){

                map[i][j]=s.charAt(j);

                if(map[i][j]=='S'){

                    x0=i;

                    y0=j;

                }

            }

        }

        dfs(0,x0,y0,n,m);

        if(minTime<=t){

            System.out.println("YES");

        }

        else {

            System.out.println("NO");

        }

    }

    private static void dfs(int t,int x,int y,int n,int m){

        if(map[x][y]=='D'){

            if(t<minTime){

                minTime=t;

            }

        }

        else {

            if(t>minTime){

                return;

            }

            for(int i=0;i<4;i++){

                int xx = x+goX[i];

                int yy = y+goY[i];

                if(xx>=0 && yy>=0 && xx<n && yy<m){

                    if(map[xx][yy]!='X' && log[xx][yy]==0){

                        log[xx][yy]=1;

                        dfs(t+1,xx,yy,n,m);

                        log[xx][yy]=0;

                    }

                }

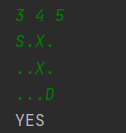
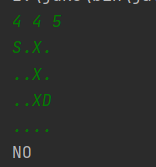
            }

        }

    }

}

结果：



# 4. N皇后问题

在N\*N的方格棋盘放置了N个皇后，使得它们不相互攻击（即任意2个皇后不允许处在同一排，同一列，也不允许处在与棋盘边框成45角的斜线上。你的任务是，对于给定的N，求出有多少种合法的放置方法。

Input

共有若干行，每行一个正整数N≤10，表示棋盘和皇后的数量；如果N=0，表示结束。

Output

共有若干行，每行一个正整数，表示对应输入行的皇后的不同放置数量。

Sample Input

1

8

5

0

Sample Output

1

92

10

package com.my.algocode;

import java.util.Scanner;

public class Queen {

    private static int[][] map;

    private static int[] logX;//行记录

    private static int[] logY;//列记录

    private static int count = 0;

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int n = input.nextInt();

        while (n!=0){

            map = new int[n][n];

            logX = new int[n];

            logY = new int[n];

            for(int i=0;i<n;i++){

                logX[i]=-12;

                logY[i]=-12;

            }

            dfs(0,n);

            System.out.println(count);

            count=0;

            n=input.nextInt();

        }

    }

    public static void dfs(int t,int n){

        if(t==n){

            count++;

        }

        else {

            for(int i=0;i<n;i++){

                if(check(t,i)){

                    logX[t]=t;

                    logY[t]=i;

                    dfs(t+1,n);

                    logX[t]=-12;

                    logY[t]=-12;

                }

            }

        }

    }

    public static boolean check(int x,int y){

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

            if(logY[i]!=-12 &&logX[i]!=-12){

                if (logX[i]==x){

                    return false;

                }

                if(logY[i] == y){

                    return false;

                }

                if(logX[i]-logY[i]==x-y){

                    return false;

                }

                if(logX[i]+logY[i]==x+y){

                    return false;

                }

            }

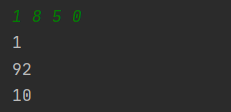
        }

        return true;

    }

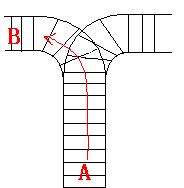
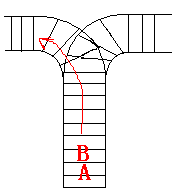
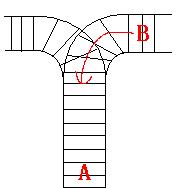
}

结果：



**5. Train Problem I**

Problem Description

As the new term comes, the Ignatius Train Station is very busy nowadays. A lot of student want to get back to school by train(because the trains in the Ignatius Train Station is the fastest all over the world ^v^). But here comes a problem, there is only one railway where all the trains stop. So all the trains come in from one side and get out from the other side. For this problem, if train A gets into the railway first, and then train B gets into the railway before train A leaves, train A can't leave until train B leaves. The pictures below figure out the problem. Now the problem for you is, there are at most 9 trains in the station, all the trains has an ID(numbered from 1 to n), the trains get into the railway in an order O1, your task is to determine whether the trains can get out in an order O2.  


Input

The input contains several test cases. Each test case consists of an integer, the number of trains, and two strings, the order of the trains come in:O1, and the order of the trains leave:O2. The input is terminated by the end of file. More details in the Sample Input.

Output

The output contains a string "No." if you can't exchange O2 to O1, or you should output a line contains "Yes.", and then output your way in exchanging the order(you should output "in" for a train getting into the railway, and "out" for a train getting out of the railway). Print a line contains "FINISH" after each test case. More details in the Sample Output.

Sample Input

3 123 321

3 123 312

Sample Output

Yes.

in

in

in

out

out

out

FINISH

No.

FINISH

代码  
package com.my.algo.DFS01;

import java.util.ArrayDeque;

import java.util.Deque;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner inputInt = new Scanner(System.in);

        int n = inputInt.nextInt();

        while (n!=0){

            Scanner inputStr = new Scanner(System.in);

            String str1 = inputStr.next();

            String str2 = inputStr.next();

            char[] carr1 = str1.toCharArray();

            char[] carr2 = str2.toCharArray();

            int[] log = new int[1000];

            int flag = 0;

            Deque<Character> deque = new ArrayDeque<>();

            int j = 0;

            int k = 0;

            for(int i=0;i< carr1.length;i++){

                deque.push(carr1[i]);

                log[k] = 1;

                k++;

                if(carr1[i]==carr2[j]){

                    deque.pop();

                    log[k]=-1;

                    k++;

                    j++;

                }

            }

            while (!deque.isEmpty()){

                char x = deque.pop();

                if(x!=carr2[j]){

                    System.out.println("No");

                    flag = 1;

                    break;

                }

                log[k] = -1;

                k++;

                j++;

            }

            if(flag==0){

                System.out.println("Yes");

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

                    if(log[i]==1){

                        System.out.println("in");

                    }

                    else if(log[i]==-1){

                        System.out.println("out");

                    }

                }

            }

            System.out.println("Finish");

            n=inputInt.nextInt();

        }

    }

}

结果：

