

《 算法设计与分析 》

实验报告本

|  |  |
| --- | --- |
| 班 级： | **计211** |
| 学 号： | **21013017** |
| 姓 名： | **蒋文强** |
| 指导教师： | **叶琪** |

信息科学与工程学院

2023年 12月

**实验一 递归与分治算法**

**一、实验目的**

**1、**熟悉数据结构和基本的排序和搜索算法，熟悉编程语言的集成开发环境，掌握程序设计与实现的能力，分析算法的复杂度。

2、通过上机实验，要求掌握递归与分治法算法的问题描述、算法设计思想、程序设计和算法复杂性分析等。

**二、实验装置**

**Win7, Visual C++/Java**

**三、实验内容**

**（**在第一次实验题目库中选择题目，填写题目内容及输入输出要求**）**

## Quoit Design

**Time Limit : 10000/5000ms (Java/Other)   Memory Limit : 65536/32768K (Java/Other)**

**Total Submission(s) : 0   Accepted Submission(s) : 0**

Font: Times New Roman | Verdana | Georgia

Font Size: ← →

**Problem Description**

Have you ever played quoit in a playground? Quoit is a game in which flat rings are pitched at some toys, with all the toys encircled awarded.

In the field of Cyberground, the position of each toy is fixed, and the ring is carefully designed so it can only encircle one toy at a time. On the other hand, to make the game look more attractive, the ring is designed to have the largest radius. Given a configuration of the field, you are supposed to find the radius of such a ring.

Assume that all the toys are points on a plane. A point is encircled by the ring if the distance between the point and the center of the ring is strictly less than the radius of the ring. If two toys are placed at the same point, the radius of the ring is considered to be 0.

**Input**

The input consists of several test cases. For each case, the first line contains an integer N (2 <= N <= 100,000), the total number of toys in the field. Then N lines follow, each contains a pair of (x, y) which are the coordinates of a toy. The input is terminated by N = 0.

**Output**

For each test case, print in one line the radius of the ring required by the Cyberground manager, accurate up to 2 decimal places.

## Elevator

**Time Limit : 2000/1000ms (Java/Other)   Memory Limit : 65536/32768K (Java/Other)**

**Total Submission(s) : 0   Accepted Submission(s) : 0**

Font: Times New Roman | Verdana | Georgia

Font Size: ← →

**Problem Description**

The highest building in our city has only one elevator. A request list is made up with N positive numbers. The numbers denote at which floors the elevator will stop, in specified order. It costs 6 seconds to move the elevator up one floor, and 4 seconds to move down one floor. The elevator will stay for 5 seconds at each stop.

For a given request list, you are to compute the total time spent to fulfill the requests on the list. The elevator is on the 0th floor at the beginning and does not have to return to the ground floor when the requests are fulfilled.

**Input**

There are multiple test cases. Each case contains a positive integer N, followed by N positive numbers. All the numbers in the input are less than 100. A test case with N = 0 denotes the end of input. This test case is not to be processed.

**Output**

Print the total time on a single line for each test case.

## Number Sequence

**Time Limit : 2000/1000ms (Java/Other)   Memory Limit : 65536/32768K (Java/Other)**

**Total Submission(s) : 0   Accepted Submission(s) : 0**

Font: Times New Roman | Verdana | Georgia

Font Size: ← →

**Problem Description**

A number sequence is defined as follows:

f(1) = 1, f(2) = 1, f(n) = (A \* f(n - 1) + B \* f(n - 2)) mod 7.

Given A, B, and n, you are to calculate the value of f(n).

**Input**

The input consists of multiple test cases. Each test case contains 3 integers A, B and n on a single line (1 <= A, B <= 1000, 1 <= n <= 100,000,000). Three zeros signal the end of input and this test case is not to be processed.

**Output**

For each test case, print the value of f(n) on a single line.

## 汉诺塔II

**Time Limit : 2000/1000ms (Java/Other)   Memory Limit : 65536/32768K (Java/Other)**

**Total Submission(s) : 0   Accepted Submission(s) : 0**

Font: Times New Roman | Verdana | Georgia

Font Size: ← →

**Problem Description**

经典的汉诺塔问题经常作为一个递归的经典例题存在。可能有人并不知道汉诺塔问题的典故。汉诺塔来源于印度传说的一个故事，上帝创造世界时作了三根金刚石柱子，在一根柱子上从下往上按大小顺序摞着64片黄金圆盘。上帝命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一回只能移动一个圆盘。有预言说，这件事完成时宇宙会在一瞬间闪电式毁灭。也有人相信婆罗门至今仍在一刻不停地搬动着圆盘。恩，当然这个传说并不可信，如今汉诺塔更多的是作为一个玩具存在。Gardon就收到了一个汉诺塔玩具作为生日礼物。

　　Gardon是个怕麻烦的人（恩，就是爱偷懒的人），很显然将64个圆盘逐一搬动直到所有的盘子都到达第三个柱子上很困难，所以Gardon决定作个小弊，他又找来了一根一模一样的柱子，通过这个柱子来更快的把所有的盘子移到第三个柱子上。下面的问题就是：当Gardon在一次游戏中使用了N个盘子时，他需要多少次移动才能把他们都移到第三个柱子上？很显然，在没有第四个柱子时，问题的解是2^N-1，但现在有了这个柱子的帮助，又该是多少呢？

**Input**

包含多组数据，每个数据一行，是盘子的数目N(1<=N<=64)。

**Output**

对于每组数据，输出一个数，到达目标需要的最少的移动数。

## 5.Uniform Generator

**Input**

Each line of input will contain a pair of integers for STEP and MOD in that order (1 <= STEP, MOD <= 100000).

**Output**

For each line of input, your program should print the STEP value right- justified in columns 1 through 10, the MOD value right-justified in columns 11 through 20 and either "Good Choice" or "Bad Choice" left-justified starting in column 25. The "Good Choice" message should be printed when the selection of STEP and MOD will generate all the numbers between and including 0 and MOD-1 when MOD numbers are generated. Otherwise, your program should print the message "Bad Choice". After each output test set, your program should print exactly one blank line.

## 题目一

给定一个大小为n的数组nums，返回其中的多数元素。多数元素是指在数组中出现次数大于IMG_256的元素。假设数组是非空的，并且给定的数组总是存在多数元素。

a)          示例1。输入：[3, 2, 3]；输出：3

b)         示例2。输入：[2, 2, 1, 1, 1, 2, 2]；输出：2

## 题目二

连续数列。给定一个整数数组，找出总和最大的连续数列，并返回总和。

a)示例：输入：[-2, 1, -3, 4, -1, 2, 1, -5, 4]；输出：6；解释：[4, -1, 3, 1]和最大为6

## 题目三

至少有K个重复字符的最长子串。给定一个字符串S和一个整数k，找出S中的最长子串，要求该子串中的每一个字符出现次数都不少于k。返回这一子串的长度。如果不存在这样的子字符串，则返回0。（S仅由小写字母组成）

## 题目四

最长的美好子字符串。当一个字符串s包含的每一种字母的大写和小写形式同时出现在s中，就称这个字符串s是美好字符串。比如，“abABB”是美好字符串，因为“A”和“a”同时出现了，且“B”和“b”也同时出现了。然而，“abA”不是美好字符串因为‘b’出现了，而‘B’没有出现。给定一个字符串s，请返回s最长的美好子字符串。如果有多个答案，请返回最早出现的一个。如果不存在美好子字符串，请返回一个空字符串。

## 题目四

库存管理。仓库管理员以数组stock形式记录商品库存表，其中stock[i]表示对应商品库存余量。请返回库存余量最少的cnt个商品余量，返回顺序不限。

示例1。输入：stock = [2, 5, 7, 4], cnt = 1；输出：[2]

示例2。输入：stock = [0, 2, 3, 6], cnt = 2；输出：[0, 2]或[2, 0]

**四、程序运行结果**

**（说明设计思路，解释使用的数据结构，计算时间复杂度）**

Quoit Design（玩具设计）

设计思路： 针对每对玩具，通过计算以其中点为圆心时的最大半径，遍历所有可能的圆心，找到最小的最大半径，即为所求。

数据结构： 使用结构体 Point 表示平面上的点，包含 x 和 y 坐标。通过动态数组 toys 存储所有的玩具坐标。

时间复杂度计算： 外层循环遍历所有可能的圆心，导致时间复杂度为 O(N^2)。 内层循环计算每个圆心的最大半径，时间复杂度为 O(N)。 综合得到总体时间复杂度为 O(N^3)。

Elevator（电梯运行）

设计思路： 遍历请求列表，计算每一层楼之间的时间消耗，对于上升和下降，分别计算时间并累加。每一层停留时间为固定值，累加到总时间中。

数据结构： 使用基本数据类型，如整数表示楼层和时间。通过循环结构遍历请求列表。

时间复杂度计算： 遍历请求列表导致时间复杂度为 O(N)。每次迭代中执行基本数学运算，时间复杂度为 O(1)。 综合得到总体时间复杂度为 O(N)。

Number Sequence（数列计算）

设计思路： 使用迭代的方式计算数列中的每一项，通过循环计算从第3项到第n项的值，根据给定的递推关系计算当前项的值。

数据结构： 使用基本数据类型，如整数，来表示系数和数列项。通过循环结构进行迭代计算。

时间复杂度计算： 循环迭代计算数列的每一项，导致时间复杂度为 O(n)。每次迭代执行基本数学运算，时间复杂度为 O(1)。

汉诺塔II

设计思路： 使用递归求解汉诺塔问题，但在每个递归步骤中，除了考虑经典汉诺塔的移动方式外，还要考虑使用第四根柱子的情况。递归公式为 F(N) = 2 \* F(N-1) + 1。

数据结构： 使用基本数据类型，如整数，表示盘子的数量和移动次数。通过递归结构解决问题。

时间复杂度计算： 递归深度为 N，每层递归执行 O(1) 的操作。总体时间复杂度为 O(N)。

Uniform Generator（均匀随机数生成器）

设计思路： 对于给定的 STEP 和 MOD，使用循环遍历检查是否能生成均匀分布的伪随机数序列。使用一个标记数组记录生成的数字，若所有数字都生成，则是良好选择；否则，是糟糕选择。

数据结构： 使用基本数据类型，如整数，表示步长 STEP 和模数 MOD。使用 vector 作为标记数组，记录生成的数字是否被访问过。

时间复杂度计算： 遍历 MOD 次导致时间复杂度为 O(MOD)。每次循环执行基本的数学运算和数组访问，时间复杂度为 O(1)。

## 题目一

**设计思路：**

一个数组中有一个数字出现次数大于 n/2 ，从第 0 个字符开始，假设它就是最多的那个数字，遇到相同的数字则计数 +1 ， 遇到不同的则计数 -1

其实就是互相消耗，等到计数为 0 的时候，表示本次互拼完毕，从下一个字符重新开始互拼，但是归根结底出现次数大于 n/2 的这个数字数量更多，因此也是最后保留的字符。

**数据结构：**数组

**时间复杂度：**o（n）

## 题目二

**设计思路 :**

同样利用分治和递归的思想，将原先的数组划分为一个个小数组，先找出小数组的总和，再合并数组并算出合并后数的总和，与原来数组比较，直到数组合并比较完，最后输出总和最大的那一个结果

**数据结构：**数组

**时间复杂度：**o（n）

## 题目三

**设计思路：**利用分治和递归的思想，将输入的字符串划分开来，分别算出是否符合条件，并返回相应情况，若符合条件，则合并起来继续计算合并的字符串的情况，直到计算完所有情况。最后输出符合条件的最长字符串。

**数据结构：**数组

**时间复杂度：**o（n）

## 题目四

**设计思路：**利用分治和递归的思想，将输入的数组划分开，并逐个挑选，选出符合数量要求及条件的数并输出。

**数据结构：**数组和指针

**时间复杂度：**o（n）

## 题目五

**设计思路：**同样是利用分治和递归的思想，将输入的数组字符划分开，并逐个组合并挑选，选出符合条件的字符串，直到求出最长的字符串

**数据结构：**数组

**时间复杂度：**O（n）

**实验一成绩：\_\_\_\_\_\_\_\_\_\_\_\_ 教师签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**实验二 动态规划和贪心算法**

**一、实验目的**

1、理解动态规划法的设计思想，分析是否满足最优子结构性质，刻画其结构特征，递归地定义最优值（动态规划方程），以自底向上的方式计算出最优值，构造最优解。掌握动态规划的算法框架和设计策略。

2、理解贪心算法的设计思想，掌握贪心算法的算法框架和设计策略，选取度量标准，逐步构造最优解。

**二、实验装置**

**Win7, Visual C++/Java**

**三、实验内容**

**（**在第一次实验题目库中选择题目，填写题目内容及输入输出要求**）**

## 1. Max Sum

Problem Description

Given a sequence a[1],a[2],a[3]......a[n], your job is to calculate the max sum of a sub-sequence. For example, given (6,-1,5,4,-7), the max sum in this sequence is 6 + (-1) + 5 + 4 = 14.

Input

The first line of the input contains an integer T(1<=T<=20) which means the number of test cases. Then T lines follow, each line starts with a number N(1<=N<=100000), then N integers followed(all the integers are between -1000 and 1000).

Output

For each test case, you should output two lines. The first line is "Case #:", # means the number of the test case. The second line contains three integers, the Max Sum in the sequence, the start position of the sub-sequence, the end position of the sub-sequence. If there are more than one result, output the first one. Output a blank line between two cases.

## 2. Robberies

Input

The first line of input gives T, the number of cases. For each scenario, the first line of input gives a floating point number P, the probability Roy needs to be below, and an integer N, the number of banks he has plans for. Then follow N lines, where line j gives an integer Mj and a floating point number Pj .

Bank j contains Mj millions, and the probability of getting caught from robbing it is Pj .

Output

For each test case, output a line with the maximum number of millions he can expect to get while the probability of getting caught is less than the limit set.

Notes and Constraints

0 < T <= 100

0.0 <= P <= 1.0

0 < N <= 100

0 < Mj <= 100

0.0 <= Pj <= 1.0

A bank goes bankrupt if it is robbed, and you may assume that all probabilities are independent as the police have very low funds.

## 3.Employment Planning

Problem Description

A project manager wants to determine the number of the workers needed in every month. He does know the minimal number of the workers needed in each month. When he hires or fires a worker, there will be some extra cost. Once a worker is hired, he will get the salary even if he is not working. The manager knows the costs of hiring a worker, firing a worker, and the salary of a worker. Then the manager will confront such a problem: how many workers he will hire or fire each month in order to keep the lowest total cost of the project.

Input

The input may contain several data sets. Each data set contains three lines. First line contains the months of the project planed to use which is no more than 12. The second line contains the cost of hiring a worker, the amount of the salary, the cost of firing a worker. The third line contains several numbers, which represent the minimal number of the workers needed each month. The input is terminated by line containing a single '0'.

Output

The output contains one line. The minimal total cost of the project.

## 4. The King’s Ups and Downs

**Input**

The first line of input contains a single integer P, (1 <= P <= 1000), which is the number of data sets that follow. Each data set consists of single line of input containing two integers. The first integer, D is the data set number. The second integer, n (1 <= n <= 20), is the number of guards of differing heights.

**Output**

For each data set there is one line of output. It contains the data set number (D) followed by a single space, followed by the number of up and down orders for the n guards.

## 5. Attack on Titans

**Input**

There are multiple test cases. For each case, there is a line containing 3 integers N (0 < N < 1000000), M (0 < M < 10000) and K (0 < K < 10000), separated by spaces.

**Output**

 One line for each case, you should output the number of ways mod 1000000007.

## 6、题目一

用2台处理机A和B处理n个作业。

## 7、题目二

在一个操场上一排地摆放着N堆石子。现要将石子有次序地合并成一堆。规定每次只能选相邻的2堆石子合并成新的一堆，并将新的一堆石子数记为该次合并的得分。请设计一个程序，计算出将N堆石子合并成一堆的最小得分和最大得分。

## 题目三

给定一个由n行数字组成的数字三角形，试设计一个算法，计算从三角形的顶至底的一条路径，使该路径经过的数字总和最大。由文件input.txt提供输入数据。文件的第1行是数字三角形的行数n，1 ≤ n ≤ 100 。接下来n行是数字三角形各行中的数字，所有的数字在0~99之间。将计算结果输出到文件output.txt。文件的第1行中的数字是计算出的最大值

## 题目四

长江游艇俱乐部在长江上设置了n个游艇出租站 1 , 2 , … … , n 1,2,……,n1,2,……,n。游客可在这些游艇出租站租用游艇，并在下游的任何一个游艇出租站归还游艇。游艇出租站i到游艇出租站 j 之间的租金为 r ( i , j ) ， 1 < = i < j < = n 。试设计一个算法，计算出从游艇出租站 1 到游艇出租站 n 所需的最少租金。

## 题目五

关于整数的2元圈乘运算定义为

(X圈乘Y)=10 进制整数X的各位数字之和\*10 进制整数 Y的最大数字+Y 的最小数字例如，(9圈乘30)=9\*3+0=27。

对于给定的 10 进制整数X和K，由X和#运算可以组成各种不同的表达式。试设计一个算法，计算出由 X 和圈乘运算组成的值为 K 的表达式最少需用多少个圈乘运算。

**四、程序运行结果**

**（说明设计思路，解释使用的数据结构，计算时间复杂度）**

### Max Sum

这里，找到一个数组中的最大子序列和，可以采用一个双变量的方法，一个是全局最大值，一个是当前局部最大值。遍历数组时，不断更新这两个变量的值，确定最终的最大和。数据结构主要使用整数变量来存储当前局部最大和、全局最大和以及最大子序列的起始和结束位置。

### Robberies

针对抢劫问题，考虑了每个银行的抢或不抢两种情况，使用二维数组来存储每个银行在不同概率下的最大金额。对于每个银行，遍历其概率范围，更新动态规划数组。这种方法的时间复杂度主要受到银行数量和概率范围的影响。

### Employment Planning

在雇佣问题中，使用一个二维数组来记录前i个月雇佣j个工人的最小总成本。遍历每个月和雇佣的工人数量，考虑雇佣和解雇两种情况，选择其中成本更小的方案。这里的时间复杂度受到月份数和最大工人数量的影响。

4.

**设计思路：**

使用二维数组 f 来存储动态规划的状态，其中 f[i][j] 表示在第 i 轮操作后，第 j 个状态的数量。

数组初始化：f[0][0]=1; f[0][1]=f[0][2]=0;

状态转移方程：

当 i <= u 时：f[i][0] = (f[i-1][0] + f[i-1][1] + f[i-1][2]) % 1000000007;

当 i == u + 1 时：f[i][0] = (f[i-1][0] + f[i-1][1] + f[i-1][2] - 1) % 1000000007;

当 i > u + 1 时：f[i][0] = (f[i-1][0] + f[i-1][1] + f[i-1][2] - f[i-u-1][1] - f[i-u-1][2]) % 1000000007；

当 i <= v 时：f[i][1] = (f[i-1][0] + f[i-1][1] + f[i-1][2]) % 1000000007;

当 i == v + 1 时：f[i][1] = (f[i-1][0] + f[i-1][1] + f[i-1][2] - 1) % 1000000007;

当 i > v + 1 时：f[i][1] = (f[i-1][0] + f[i-1][1] + f[i-1][2] - f[i-v-1][0] - f[i-v-1][2]) % 1000000007;

对于 f[i][2]：f[i][2] = (f[i-1][0] + f[i-1][1] + f[i-1][2]) % 1000000007;

**时间复杂度：**

外层循环是 for i in range(1, n+1)，内层循环是 for j in range(0, 3)

所以，总的时间复杂度为 O(n \* 3) = O(n)。

5.

**设计思路：**

状态定义：

dp[i][j]: 表示在长度为 i 的序列中，有 j 个 0 和 1 相邻的排列的总数。

状态初值：

对于长度为1的序列，dp[1][0] = dp[1][1] = 1，表示长度为1的序列中，不存在相邻的0和1（只有一个元素）。

状态转移方程：

dp[i][0] 和 dp[i][1] 的计算使用了之前计算的 dp 数组和组合数 c 数组的值。

对于 dp[i][0]，通过以下的状态转移方程计算： [ dp[i][0] = sum{j=0-(i-1)} dp[j][0] \* dp[i-j-1][1] \* c[i-1][j] ]

对于 dp[i][1]，也使用了相似的转移方程： [ dp[i][1] = dp[i][0] ]

**时间复杂度：**

组合数计算：O(n^2)

状态转移：O(n^2)

总：(O(n^2))

循Max Sum

使用的数据结构：

基本数据类型：整数用于表示测试用例数、数组长度、数组元素等。

整数变量：记录当前局部最大和、全局最大和、以及最大子序列的起始和结束位置。

时间复杂度计算：

遍历数组一次，时间复杂度为 O(N)。

Robberies

使用的数据结构：

结构体 Bank：表示每个银行，包含金额和概率。

二维数组 dp：存储动态规划的中间结果。

时间复杂度计算：

遍历每个银行，时间复杂度为 O(N)。

对于每个银行，遍历概率范围，时间复杂度为 O(P \* 100)。

总体时间复杂度为 O(N \* P \* 100)。

Employment Planning

使用的数据结构：

整数数组 dp：存储动态规划的中间结果。

整数数组 a：存储每个月所需的最小工人数量。

时间复杂度计算：

遍历每个月，时间复杂度为 O(t)。

对于每个月，遍历工人数量，时间复杂度为 O(maxx)。

总体时间复杂度为 O(t \* maxx)。

Attack on Titans

使用的数据结构：

二维整数数组 f：存储动态规划的中间结果。

时间复杂度计算：

遍历每个人，时间复杂度为 O(n).

The King’s Ups and Downs

使用的数据结构：

二维长整型数组 dp：存储动态规划的中间结果。

长整型数组 ans：存储最终结果。

时间复杂度计算：

外层循环遍历数据集，时间复杂度为 O(P)。

内层循环遍历1到20，计算动态规划的值，时间复杂度为 O(20)。环遍历数字范围进行动态规划计算。

## 6、题目一

**设计思路：**根据题意使用数组，由f[i][j] = min(f[i-1][j-a[i]],f[i-1][j]+b[i])来决定作业分配给A做或给B做，之后输出结果中最小的。

**数据结构：**数组

**时间复杂度：**o（n^2）

## 7、题目二

**设计思路：**对n的取值逐步分析：当n=1时，没有进行合并，得分为0；当n=2时，只有一种合并的方案，得分为两堆石子之和；当n=3时，有两种得分方案，先合并左边两个，先合并右边两个，这两种方案的区别我们可以看成是起点不同，最后一次合并的得分都是这三堆石子的总数；当n=4时，有三种得分方式，分别是先合并后面三个，先合并后面两个，先合并后面一个（也就是先合并前面三个），这三种方案对应的起点是1、2、3，最后一次合并的得分都是这四堆石子的总数。

由此可知，我们所求的最大或者最小的得分就是这几种方案中得分最大或者最小的分数加上最n堆石子的总石子数

**数据结构：**数组

**时间复杂度：**o(n^3)

## 8、题目三

**设计思路：**数字三角形的最后一行上的数字到底的数字和最大的路径为该行上的数字本身；数字三角形倒数第二行上的数字到底的数字和最大的路径为该行上的数字加上该行左下或右下的数字之和……以此类推，可以从下往上推出数字三角形的顶至底的路径经过的数字和的最大值。因此运用递归的方法可以解决

**数据结构：**数组

**时间复杂度：**O(n^2)

## 9、题目四

**设计思路:** 运用了动态规划的思路，划分出中间位置，状态转移方程为：dp[i][j] = min(dp[i][k] + dp[k][j]), k = [i+1 , j-1]。

**数据结构:**数组

**时间复杂度：**O（n^3）

## 10、题目五

**设计思路:** 运用动态规划算法求解，且计算得知若X,Y<=81\*N+9(N>=2),则原等式为X圈乘Y<=81\*N+9。设m=X的位数，则有X、@和括号组成的表达式的值小于等于81\*max{2,m}+9设K=Mi@Nj,f(K)表示表达式值为K的最小运算次数，则f(K)=min{f(Mi)+f(Nj)},i=1..m，j=1..n

**数据结构：**数组

**时间复杂度：**o(n^2)

**实验二成绩：\_\_\_\_\_\_\_\_\_\_\_\_ 教师签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**实验三 回溯算法**

**一、实验目的**

**1、**理解回溯法的设计思想，回溯法是一个既带有系统性又带有跳跃性的搜索算法。掌握从包含问题的所有解的解空间树中，按照深度优先的策略，从根结点出发搜索解空间树的过程。掌握回溯法的算法框架和设计策略。

**二、实验装置**

**Win7, Visual C++/Java**

**三、实验内容**

**（在第三次实验题目库中选择题目，填写题目内容及输入输出要求）**

## 小希的数表

【问题描述】

Gardon 昨天给小希布置了一道作业，即根据一张由不超过 5000 的 N(3<=N<=100)个正整数组成的数表两两相加得到 N\*(N-1)/2 个和，然后再将它们排序。例如，如果数表里含有四个数 1，3，4，9，那么正确答案是 4，5，7，10，12，13。小希做完作业以后出去玩了一阵，可是下午回家时发现原来的那张数表不见了，好在她做出的答案还在，你能帮助她根据她的答案计算出原来的数表么？

【输入形式】

包含多组数据，每组数据以一个 N 开头，接下来的一行有按照大小顺序排列的 N\*(N-1)/2 个数，是小希完成的答案。文件最后以一个 0 结束。

假设输入保证解的存在性和唯一性。

【输出形式】

对于每组数据，输出原来的数表。它们也应当是按照顺序排列的。

## 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)

## 3．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。

## 4.八球胜负问题

【问题描述】

8球是一种台球竞赛的规则。台面上有7个红球、7个黄球以及一个黑球，当然还有一个白球。对于本题，我们使用如下的简化规则：红、黄两名选手轮流用白球击打各自颜色的球，如果将该颜色的7个球全部打进，则这名选手可以打黑球，如果打进则算他胜。如果在打进自己颜色的所有球之前就把黑球打进，则算输。如果选手不慎打进了对手的球，入球依然有效。

现在给出打进的球（白球除外）的顺序，以及黑球由哪方打进，你的任务是判定哪方是胜者。

假设不会有一杆同时打进一颗黑球和其他彩球。

【输入形式】

输入包含多组数据。每组数据第一行是一个整数NN(1≤N≤15)，表示打进的球的个数，N=0表示结束。随后有一行，包含N个字符，依序表示打进的是何种球。如果是B，表示是红方打进的黑球，如果是L，表示是黄方打进的黑球。如果是Y则表示是黄球，R表示红球。字符间没有空格。

所有输入都满足如下条件：最后一颗球打进时这局比赛正好结束，而且打进的红球和黑球都不超过7个。

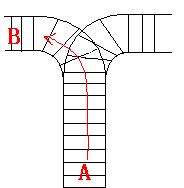
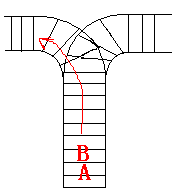
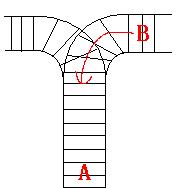
【输出形式】

对每组数据，输出一行。如果红方胜，输出Red；黄方胜，输出Yellow。

## 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.

## 题目一

子集和问题的一个实例为<S,c>。其中，S={x1,x2,…,xn}是一个正整数的集合，c是一个正整数。子集和问题判定是否存在S的一个子集S1，使得子集S1和等于c。

对于给定的正整数的集合S={x1,x2,…,xn}和正整数c，编程计算S 的一个子集S1，使得子集S1和等于c。

第1行有2个正整数n和c，n表示S的个数，c是子集和的目标值。接下来的1 行中，有n个正整数，表示集合S中的元素。

输出一个解即可，如果没有解，输出“No Solution!”。

## 题目二

最小长度电路板排列问题是大规模电子系统设计中提出的实际问题。该问题的提法是，将块电路板以最佳排列方案插入带有n个插槽的机箱中。n 块电路板的不同的排列方式对应于不同的电路板插入方案。设B ={1，2....，n}是n块电路板的集合。集合L ={N1，N2，...，Nm}是n块电路板的 m 连接块其中每个连接块 Ni是B 的一个子集，且 Ni中的电路板用同一根导线连接在一起。在最小长度电路板排列问题中，连接块的长度是指该连接块中第 1 块电路板到最后 1 块电路板之间的距离。试设计一个回溯法找出所给 n 个电路板的最佳排列，使得 m 个连接块中最大长度达到最小。

## 题目三

羽毛球队有男女运动员各n人。给定2个n×n矩阵P和Q。

P[i][j]是男运动员i的女运动员j配对组成混合双打的男运动员竞赛优势；Q[i][j]是女运动员i和男运动员j配对的女运动员竞赛优势。由于技术配合和心理状态等各种因素影响，P[i][j]不一定等于Q[i][j]。男运动员i和女运动员j配对组成混合双打的男女双方竞赛优势为P[i][j]×Q[i][j]。

设计一个算法，计算男女运动员最佳配对法，使各组男女双方竞赛优势的总和达到最大。

## 题目四

设有 n 个立方体，每个立方体的每一面用红、黄、蓝、绿等 n 种颜色之一染色。要把这n 个立方体叠成一个方形柱体，使得柱体的 4 个侧面的每一侧均有 n 种不同的颜色。试设计一个回溯算法，计算出 n 个立方体的一种满足要求的叠置方案。

对于给定的 n 个立方体以及每个立方体各面的颜色，计算出 n 个立方体的一种叠置方案，使得柱体的 4 个侧面的每一侧均有 n 种不同的颜色。

第一行有 1 个正整数 n，0< n< 27，表示给定的立方体个 数和颜色数均为 n。第 2 行是 n 个大写英文字母组成的字符串。该字符串的第 k(0≤ k< n) 个字符代表第 k 种颜色。接下来的 n 行中，每行有 6 个数，表示立方体各面的颜色。

## 题目五

整数变换问题。关于整数 i 的变换 f 和 g 定义如下: f(i)=3i，g(i)=[i/2]

试设计一个算法，对于给定的 2 个整数 n 和 m，用最少的 f 和 g 变换次数将 n 变换为 m。

例如，可以将整数 15 用 4 次变换将它变换为整数 4: 4=gfgg(15)。当整数 n 不可能变换为整数 m 时，算法应如何处理?

对任意给定的整数 n 和 m，编程计算将整数 n 变换为整数 m 所需要的最少变换次数。

## 小希的数表

**设计思路：**

使用回溯算法穷举可能的数表，验证是否符合条件。

遍历找到第一个未访问的和，假设它是 a2+a3，然后通过数学推导找到 a1，a4，a5... 直至找到所有数字。

输出符合条件的数表。

**使用的数据结构：**

使用整型数组 sum 存储和。

使用整型数组 vis 标记和是否被访问。

使用整型数组 num 存储找到的数字。

**时间复杂度计算：**

外层循环遍历多组数据，时间复杂度为 O(P)。

内层循环遍历 sum 数组，最坏情况下需要遍历 n\*(n-1)/2 次，时间复杂度为 O(N^2)。

## 变形课

**设计思路：**

使用深度优先搜索（DFS）遍历咒语表，尝试将球（B）变成老鼠（M）。

若成功变成老鼠，则输出"Yes."，否则输出"No."。

**使用的数据结构：**

二维字符数组 c 存储咒语表。

一维布尔数组 flag 标记咒语是否被使用过。

整数 cnt 记录咒语的数量。

布尔变量 ans 记录是否成功变成老鼠。

**时间复杂度计算：**

每次 DFS 的时间复杂度为 O(N)，其中 N 为咒语的数量。

遍历所有咒语的时间复杂度为 O(M)，其中 M 为咒语的总长度。

## N皇后问题

**设计思路：**

在N\*N的棋盘上放置N个皇后，使得它们不相互攻击，即满足不在同一行、同一列、不在45度斜线上。使用深度优先搜索（DFS）算法来遍历所有可能的皇后放置方案，通过剪枝策略来排除不符合条件的情况。

**使用的数据结构：**

一维数组 d[10010] 用于存储每行皇后所在的列数。

二维字符数组 result[10010][10010] 用于表示每个位置是否放置皇后，其中 . 表示空格，Q 表示皇后。

布尔数组 flag[10010] 用于标记某列是否已经被占用。

**时间复杂度计算：**

对于每一行的每一列，都需要判断是否符合条件，时间复杂度为O(N^2)。

对于每一行的深度优先搜索，最坏情况下需要尝试N列，总体时间复杂度为O(N!)。

综合来看，时间复杂度为O(N^2 \* N!)

## 4.八球胜负问题

**设计思路：**

使用两个变量 r 和 y 分别表示红球和黄球的数量，遍历输入的字符串，统计红球和黄球的数量。

根据规则判断比赛结果：

如果当前球是黄球且黄球数量等于7，或者当前球是红方打进的黑球且红球数量小于7，则黄方胜利，输出"Yellow"。

如果当前球是红球且红球数量等于7，或者当前球是黄方打进的黑球且黄球数量小于7，则红方胜利，输出"Red"。

**使用的数据结构：**

两个整数 r 和 y 分别表示红球和黄球的数量。

一个字符串 s 存储打进的球的顺序。

**时间复杂度计算：**

遍历字符串，时间复杂度为O(N)，其中N为打进的球的数量

## 5. Train Problem I

**设计思路：**

使用一个栈 s 来模拟火车站的铁轨，栈底元素为 #。

遍历输入的 o1，将元素依次压入栈中，并根据 o2 的顺序弹出栈顶元素。

如果最后栈为空（栈顶元素为 #），则说明可以按照 o2 的顺序出站，输出 "Yes."，并按照栈操作顺序输出进站和出站的操作。

如果栈不为空，则说明无法按照 o2 的顺序出站，输出 "No."。

**使用的数据结构：**

一个字符栈 s，用于模拟铁轨。

两个字符数组 o1 和 o2，分别表示火车进站和出站的顺序。

一个整型数组 op，用于记录每一步的操作（1 表示进站，0 表示出站）。

**时间复杂度计算：**

遍历字符数组，时间复杂度为O(N)，其中N为火车的数量

## 6、题目一

**设计思路：**使用深度优先解决。设立一个flag数组用来标记第i个数字是否已经被引用，设立一个cnt来记录是否都找到解，若找到则结束搜索。写dfs函数时间传入两个值，一个来表示是否已经遍历所有的项，一个来记录数字和。分左右两枝来进行搜索。

**数据结构：**数组

**时间复杂度：**O(N)

## 7、题目二

**设计思路：**运用回溯法，对所有的电路板进行全排列，每次找到连接块的最左和最右电路板，相减得到连接块的长度，在全排列的过程中找到最小长度。

**数据结构：**栈回溯法

**时间复杂度：**O（n^2）

## 8、题目三

**设计思路：**固定男运动员选女运动员，构成一颗排列数。树的第i行表示第i个男运动员，树结点序号表示与当前行男运动员匹配的女运动员序号。

若贪心的加上其余运动员匹配最大优势后，仍不能超过已经搜索出的最优解，那么向下搜完整颗子树也不会得到最优解，则要进行剪枝操作。因为再向下搜索的结果只可能是小于等于最大优势的。

**数据结构：**栈和数组

**时间复杂度：**O(N^2)

## 9、题目四

**设计思路:** 首先要先开一个数组，因为要使柱子的每一侧都有n种颜色，只有n个正方体，所以颜色不能重复，所以用bool型。

接着用深搜递归函数，设参数x，表示现在正在考虑第x个正方体

如果x>n了，那么跳出函数，答案加一。

如果没有，枚举这个正方体的六种摆放方法，如果其中某一种不会重复，则dfs这个状态即可。

**数据结构：**栈数组

**时间复杂度：**O（n^2）

## 10、题目五

**设计思路:** 本题的解空间树是子集树，左子树为 f 操作右子树为 g 操作。

利用回溯算法的解空间树是在深度优先搜索的过程中建立的这一性质，将树的深度等价为操作次数。

**数据结构：**二叉平衡树和数组

**时间复杂度：**O（n）

**实验三成绩：\_\_\_\_\_\_\_\_\_\_\_\_ 教师签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**实验四 分支限界算法**

**一、实验目的**

**1、**理解分支限界法的设计思想，掌握分支限界法的算法框架和设计策略。

**2、**通过上机实验，要求掌握算法设计思想、程序设计和算法复杂性分析等。

**二、实验装置**

**Win7, Visual C++/Java**

**三、实验内容**

**（在第四次实验题目库中选择题目，填写题目内容及输入输出要求）**

## Rescue

**Input**

First line contains two integers stand for N and M.

Then N lines follows, every line has M characters. "." stands for road, "a" stands for Angel, and "r" stands for each of Angel's friend.

Process to the end of the file.

**Output**

For each test case, your program should output a single integer, standing for the minimal time needed. If such a number does no exist, you should output a line containing "Poor ANGEL has to stay in the prison all his life."

2．I NEED A OFFER!

**Input**

输入有若干组数据，每组数据的第一行有两个正整数n,m(0<=n<=10000,0<=m<=10000)

后面的m行，每行都有两个数据ai(整型),bi(实型)分别表示第i个学校的申请费用和可能拿到offer的概率。

输入的最后有两个0。

**Output**

每组数据都对应一个输出，表示Speakless可能得到至少一份offer的最大概率。用百分数表示，精确到小数点后一位。

## 3． 非常可乐

**Problem Description**

大家一定觉的运动以后喝可乐是一件很惬意的事情，但是seeyou却不这么认为。因为每次当seeyou买了可乐以后，阿牛就要求和seeyou一起分享这一瓶可乐，而且一定要喝的和seeyou一样多。但seeyou的手中只有两个杯子，它们的容量分别是N 毫升和M 毫升 可乐的体积为S （S<101）毫升　(正好装满一瓶) ，它们三个之间可以相互倒可乐 (都是没有刻度的，且 S==N+M，101＞S＞0，N＞0，M＞0) 。聪明的ACMER你们说他们能平分吗？如果能请输出倒可乐的最少的次数，如果不能输出"NO"。

**Input**

三个整数 : S 可乐的体积 , N 和 M是两个杯子的容量，以"0 0 0"结束。

**Output**

如果能平分的话请输出最少要倒的次数，否则输出"NO"。

## A计划

**Problem Description**

可怜的公主在一次次被魔王掳走一次次被骑士们救回来之后，而今，不幸的她再一次面临生命的考验。魔王已经发出消息说将在T时刻吃掉公主，因为他听信谣言说吃公主的肉也能长生不老。年迈的国王正是心急如焚，告招天下勇士来拯救公主。不过公主早已习以为常，她深信智勇的骑士LJ肯定能将她救出。

现据密探所报，公主被关在一个两层的迷宫里，迷宫的入口是S（0，0，0），公主的位置用P表示，时空传输机用#表示，墙用\*表示，平地用.表示。骑士们一进入时空传输机就会被转到另一层的相对位置，但如果被转到的位置是墙的话，那骑士们就会被撞死。骑士们在一层中只能前后左右移动，每移动一格花1时刻。层间的移动只能通过时空传输机，且不需要任何时间。

**Input**

输入的第一行C表示共有C个测试数据，每个测试数据的前一行有三个整数N，M，T。 N，M迷宫的大小N\*M（1 <= N,M <=10)。T如上所意。接下去的前N\*M表示迷宫的第一层的布置情况，后N\*M表示迷宫第二层的布置情况。

**Output**

如果骑士们能够在T时刻能找到公主就输出“YES”，否则输出“NO”。

## 5.哈密顿绕行世界问题

**Problem Description**

一个规则的实心十二面体，它的 20个顶点标出世界著名的20个城市，你从一个城市出发经过每个城市刚好一次后回到出发的城市。

**Input**

前20行的第i行有3个数,表示与第i个城市相邻的3个城市.第20行以后每行有1个数m,m<=20,m>=1.m=0退出.

**Output**

输出从第m个城市出发经过每个城市1次又回到m的所有路线,如有多条路线,按字典序输出,每行1条路线.每行首先输出是第几条路线.然后个一个: 后列出经过的城市.参看Sample output

## 题目一

给定一个赋权无向图 G=(V,E)，每个顶点 v∈V 都有一个权值 w(v)。如果 U⊆VU⊆V，且对任意(u,v)∈E 有 u∈U 或 v∈U，就称 U 为图 G 的一个顶点覆盖。G 的最小权顶点覆盖是指 G 中所含顶点权之和最小的顶点覆盖。

对于给定的无向图 G，设计一个优先队列式分支限界法，计算 G 的最小权顶点覆盖。

## 题目二

给定一个无向图G=(V, E),设U⊆V是G的顶点集。对任意(u, v)∈E,若u∈U，且v∈V-U,就称(u, 1)为关于顶点集U的一条割边。顶点集U的所有割边构成图G的一个割。G的最大割是指G中所含边数最多的割。

对于给定的无向图G,设计一个优先队列式分支限界法，计算G的最大割。

## 题目三

设某一机器由n个部件组成，每种部件都可以从m个不同的供应商处购得。设wij是从供应商j处购得的部件i的重置，cij是相应的价格。设计一个优先队列式分支限界法，给出总价格不超过d的最小重量机器设计。

## 题目四

印刷电路板将布线区域划分成n\*m个方格阵列，精确的电路布线问题要求确定连接方格a的中点到方格b的中点的最短布线问题。在布线时，电路只能沿着直线或直角布线。为了避免线路相交，已布了线的方格做了封锁标记，其他线路不允许穿过被封锁的方格。

## 题目五

假设有 n 个任务由 k 个可并行工作的机器来完成。完成任务 i 需要时间为ti ，设计完成这 n 个任务的最佳调度算法，使得完成全部任务的时间最早。对任意给定的整数n和k，以及完成任务i需要的时间为ti，i=1,2,…,n。编程计算完成这n个任务的最佳调度。

**四、程序运行结果**

**（说明设计思路，解释使用的数据结构，计算时间复杂度）**

1. Rescue

Design:

使用深度优先搜索（DFS）算法进行状态搜索，找到长度为20的路径，并确保路径首尾相接。

Data Structures:

二维数组 map 表示状态之间的连接关系。

一维数组 a 记录路径。

一维数组 b 标记访问状态。

Time Complexity:

对每一个状态进行深度优先搜索，最坏情况下搜索所有可能状态，时间复杂度为 O(3^20)。

2. I NEED A OFFER!

Design:

使用动态规划算法，其中 dp[i] 表示花费 i 万美元至少获得一份 offer 的最大概率。

对每个学校遍历，更新 dp 数组。

Data Structures:

一维数组 dp 用于动态规划状态转移。

两个一维数组 v 和 f 表示学校的申请费用和获得 offer 的概率。

Time Complexity:

对每个学校和每个花费进行动态规划，时间复杂度为 O(m \* n)。

3. 非常可乐

Design:

使用广度优先搜索（BFS）表示每种状态，结构体包含三个杯子的可乐量和步数。

通过队列进行BFS遍历，每次分配可乐到其他杯子，得到新状态。

Data Structures:

结构体 node 表示每种状态，包含可乐量和步数。

三维数组 vis 标记每种状态是否出现过。

Time Complexity:

使用BFS遍历所有可能状态，最坏情况下遍历整个状态空间，时间复杂度为 O(2^S)。

4. A计划

Design:

使用广度优先搜索（BFS）表示每种状态，结构体包含三个坐标和步数。

通过队列进行BFS遍历，每次更新当前状态的坐标和步数。

Data Structures:

结构体 node 表示每种状态，包含坐标和步数。

三维数组 a 表示两层迷宫布置情况。

三维数组 book 标记每种状态是否出现过。

Time Complexity:

使用BFS遍历所有可能状态，时间复杂度为 O(N \* M \* T)。

5. 哈密顿绕行世界问题

Design:

使用深度优先搜索（DFS）求解哈密顿回路问题。

通过递归函数 dfs，从给定城市出发，探索所有可能路径。

Data Structures:

二维数组 map 表示城市之间的连接关系。

一维数组 a 存储当前路径。

一维数组 b 标记城市是否被访问过。

Time Complexity:

深度优先搜索的时间复杂度取决于搜索的状态空间，在此问题中为 O(N!)，其中 N 是城市数量。

## 5.哈密顿绕行世界问题

**设计思路：**

使用深度优先搜索（DFS）求解哈密顿回路问题。

使用递归函数 dfs，从给定城市出发，探索所有可能的路径。

使用数组 map 表示城市之间的连接关系。

使用数组 a 存储当前路径，数组 b 标记城市是否被访问过。

如果找到符合条件的路径，输出该路径。

**使用的数据结构：**

二维数组 map 表示城市之间的连接关系。

一维数组 a 用于存储当前路径。

一维数组 b 用于标记城市是否被访问过。

**时间复杂度计算：**

深度优先搜索的时间复杂度取决于搜索的状态空间。在这个问题中，最坏情况下可能需要搜索所有可能的路径，时间复杂度为 O(N!)，其中 N 是城市的数量。

题目一：

设计思路：每个顶点有两种选择，进入左子树或右子树，优先选择权值小的路径。通过优先队列存储试探的解向量，按顶点权值小的优先级取出。判断是否满足顶点覆盖，如果满足即为最小权覆盖的解；否则继续判断。

数据结构：优先队列（堆）

时间复杂度：O(n^2)

题目二：

设计思路：采用回溯法，无剪枝函数设计。比较到达叶节点后的当前解和当前已知最优解，若更新更优，则更新最优解和结点是否加入U顶点集。通过两层循环判断相连节点是否属于不同的顶点集来计算割边数。

数据结构：队列

时间复杂度：O(N \* 2^N)

题目三：

设计思路：最小机器重量设计问题采用优先队列式分支限界法，用优先队列存储活结点表。优先队列中节点优先级定义为子树最小重量。遍历节点的子节点，对符合条件的子节点插入优先队列。使用剪枝函数、限界函数和约束函数优化搜索过程。

数据结构：数组、队列

时间复杂度：O(n^2)

题目四：

设计思路：利用队列式分支限界法解决电路布线问题。创建表示节点的结构体，二维数组表示电路板。使用队列扩展活结点表，从起始节点开始扩展可行解。通过位移矩阵确定节点的上下左右扩展方向。

数据结构：数组和队列

时间复杂度：O(n)

题目五：

设计思路：通过递归实现搜索，层层返回。搜索七层，每层有三个孩子，更新最大时间。利用三个机器中花费时间最长的机器的时间作为当前分配的所用时间。

数据结构：数组和递归

时间复杂度：O(n)

**实验四成绩：\_\_\_\_\_\_\_\_\_\_\_\_ 教师签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**