# 引言

本次专业课程设计针对数据结构的内容给出了8个综合性练习,并对这些问题分析、设计、实现以及测试等方面进行了详细的论述，目的是让学生在上述内容学习的基础上,针对大型综合习题动手解决问题，进一步全面提高分析问题、解决问题的能力。

每道题给出了问题描述、输入输出格式要求、分工要求及主要算法提示等。学生可根据实际情况单独或分组加以设计和实现。

## 题目一： 二叉搜索树：各种搜索树效率比较

题目要求

本题目要求对普通的二叉搜索树、AVL树、伸展树( Splay Tree)分别实现指定操作，并分析比较这三种不同数据结构对应的一系列插入和删除操作的效率。要求测试对N个不同整数进行下列操作的效率：

●按递增顺序插入N个整数,并按同样顺序删除

●按递增顺序插人N个整数,并按相反顺序删除

●按随机顺序插入N个整数,并按随机顺序删除

要求N从1000到10000取值,并以数据规模N为横轴,运行时间为纵轴,画出3种不同数据结构对应的操作效率比较图。

分工要求

程序员:实现3种不同数据结构对应的插入删除操作;编写效率测试程序。注意源代码必须有充分注释。

测试员:提供测试输入,并根据测试结果填写运行时间记录表:画出3种不同数据结构对应的操作效率比较图；完成测试结果分析与探讨。

文档员：完成实验报告第一部分、第二部分,要求文档风格统一

简要提示

需要注意的是，比较测试的表格要设计得清晰，最好将插入和删除分开列表。

另外，当运行时间太短时，为避免得到0秒，应将同一操作重复充分多遍，得到1秒以上 的总时间，再将结果除以重复次数。若单重循环不能达到目的，可用多重嵌套循环解决。

绘图时须注意坐标轴上的单位长度要固定，经常出现的错误是0到1000的距离跟 1000到5000的距离长度相同，则绘出的曲线会对读者产生误导。

## 题目二：并 查 集 : 检 查 网 络

**题目要求**

给定一个计算机网络以及机器间的双向连线列表，每一条连线允许两端的计算机进行直接的文件传输,其他计算机间若存在一条连通路径，也可以进行间接的文件传输。请写程序判断:任意指定两台计算机,它们之间是否可以进行文件传输？

**输入要求：**

输入由若干组测试数据组成。对于每一组测试，第1行包含一个整数N(<=10000)，即网络中计算机的总台数，因而每台计算机可用1到N之间的一个正整数表示。接下来的几行输 入格式为I C1 C2或者C Cl C2或者S,其中C1和C2是两台计算机的序号,I表示在C1和C2

间输入一条连线 ,C表示检查C1和C2间是否可以传输文件，S表示该组测试结束。

当N为0时，表示全部测试结束，不要对该数据做任何处理。

**输出要求：**

对每一组C开头的测试，检查C1和C2间是否可以传输文件,若可以，则在一行中输出

“yes”,否则输出“no”。

当读到S时，检查整个网络。若网络中任意两机器间都可以传输文件，则在一行中输出“The network is connected .”，否则输出“There are k components.”，其中 k 是网络中连通集的个数。

两组测试数据之间迠输出一空行分隔。

**输入例子：**

3

C 1 2

I 1 2

C 1 2

S

3

I 3 1

I 2 3

C 1 2

S

0

**输出例子:**

no

yes

There are 2 components.

yes

The network is connected.

**分 工 要 求**

•程序员：实现程序。注意源代码必须有充分注释。

•测试员：提供测试输入，并根据测试结果填写测试报告;完成测试结果分析与探讨。

•文档员：完成实验报告第一部分、第二部分内容，要求文档风格统一。

**简 要 提 示**

该题目可应用并査集(union-find set )实现，将每台机器当作一个节点,令迮通的机器属于同一个集合。即每次读到I开头的输入时，检査C1和C2 是否属于同一个集合，若不属于，则合并它们分別属于的两个集合。每次读到C开头的输入时，检查C1和C2是否属于同一个集合，若是则表明两者之间有连通路径，可以传输文件，输出“yes”,否则输出“no”。最后检查连通集个数时，只要数一下集合中还有几个节点是根节点就可以了。由于对集合的操作主要是并(union)和查找(find〉，所以可以采用双亲表示法的树结构来表示集体。

本题应有大数据量的测试。为保证程序的效率,可采用按树的大小加权合并、查找时减小树高的改进方法。

测试员应注意边界测试，例如将N取为1的用例和N为10000的用例。

## 题目三 网络流:宇宙旅行

**题目要求**

在走遍了地球上的所有景点以后，旅游狂人开始计划他的宇宙旅行项目。经过谨慎调查，他目前掌握了一张各卫星空间站可以临时容纳的旅客人数的列表。当旅客从一个星球飞往另一个星球时，需要在若干卫星空间站临时停靠中转，而这些空间站不能接待任何旅客驻留，旅客必须立刻转乘另一艘飞船离开，所以空间站不能接待超过自己最大容量的旅客流。为了估计预算，现在旅游狂人需要知道终点星球的接待站应该设计多大容量，才能使得每艘飞船在到达时都可以保证让全部旅客下船。

**输入要求：**

输人由若干组测试数据组成。

每组测试数据的第1行包含旅行的起点星球和终点星球的名称和一个不超过500的正整数N(N为0标志全部测试结束，不要对该数据做任何处理)。

接下来的N 行里，数据格式为： sourcei destinationi capacityi，其中 sourcei 和destinationi是卫星空间站的名称或起点、终点星球的名称，正整数capacityi是飞船从sourcei到desti­

nationi—次能运载的最大旅客流量。每个名称是由 A 〜Z 之间 3 个大写字母组成的字符串，例如ZJU。

测试数据中不包含任何到达起点星球的信息以及任何从终点星球出发的信息。

**输出要求：**

对每一组测试，在一行里输出终点星球接待站应具有的最小容量，使得每艘飞船在到达时都可以保证让全部旅客下船。

**输入例子：**

EAR MAR 8

EAR AAA 300

EAR BBB 200

AAA BBB 100

AAA CCC 300

BBB DDD 200

AAA DDD 400

CCC MAR 200

DDD MAR 300

EAR MAR 8

EAR AAA 300

AAA BBB 400

EAR DDD 200

AAA DDD 100

AAA CCC 300

DDD BBB 200

CCC MAR 200

BBB MAR 300

ABC DEF 0

**输出例子：**

500

500

**分 工 要 求**

•程序员：实现程序。注意源代码必须有充分注释。

•测试员：提供测试输入，并根据测试结果填写测试报告;完成测试结果分析与探讨。

•文档员：完成实验报告第一部分、第二部分内容，要求文档风格统一。

**简要提示**

该题目中给定了起点和终点(星球），其他节点(空间站)的流入量必须等于流出量，建议用网络流(Network Flow)方法解决。每次可从剩余图（ Residual Graph)中找一条从起点到终点的最短路径，再按照修正算法进行。

实现时注意，程序应能在2秒内处理10组最大规模(N=500)的测试用例。为提高程序效率，需要为节点名建立散列表，避免在查找节点名上浪费时间。

所以这个题目的实现涉及3种算法:最短路径、散列映射、网络流。

测试时应注意边界测试，需要写个程序生成N为500的输入。

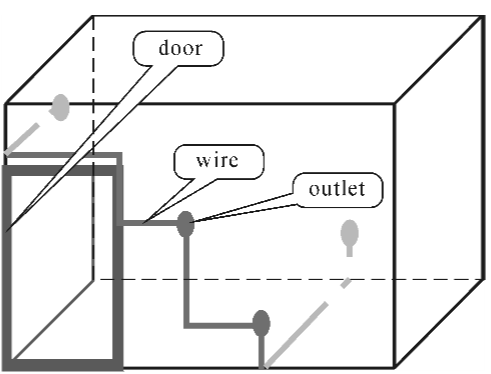
## 题目四 最 小 生 成 树 : 室 内 布 线

**题目要 求**

装修新房子是一项颇为复杂的工程,现在需要写个程序帮助房主设计室内电线的布局。

首先，墙壁上插座的位置是固定的。插座间需要有电线相连，而且要布置得整齐美观,即要求每条线都与至少一条墙边平行，且嵌入四壁或者地板(不能走屋顶)。

房主要求知道，要将所有插座连通，自己需要买的电线的最短长度。



室内布线示意图

另外，别忘了每个房间都有门，电线不可以穿门而过。上图给出了一个有4插座的房间的电线布局。

**输入要求：**

输人由若干组测试数据组成。

每组数据的第1行包含房间的长、宽、髙和插座的个数N（N为一个不超过20的正整数)。

接下去的N行中，第i行给出第i个插座的位置坐标（xi,yi,zi）;最后一行包含4个3 元组（x1,y2,z3）•••（x4,y4,z4）， 分别是长方形门框的4个角的三维坐标。4个数字全部为0表示全部测试结束,不要对该数据做任何处理。

注意:这里假设长方体形状的房间完全位于三维直角坐标系的第一象限内，并且有一个角落 在原点上。地板位于x-y平面。题目数据保证，每个插座仅属于四面墙中的一面，门上没有插座。要求每段电线的两端必须仅与插座连接，电线之间不能互相交叉焊接。

**输出要求：**

对每一组测试，在一行里输出要将所有插座连通需要买的电线的最短整数长度。

输入例子：

10 10 10 4

0 1 3.3

2.5 0 2

5 0 0.8

5 10 1

0 0 0 0 0 3 1.5 0 0 1.5 0 3

0 0 0 0

**输出例子：**

21

**分工要求**

•程序员：实现程序。注意源代码必须有充分注释。

•测试员：提供测试输入，并根据测试结果填写测试报告;完成测试结果分析与探讨。

•文档员：完成实验报告第一部分、第二部分内容，要求文档风格统一。

**简要提示**

该题目中给定了插座的位置,可以将每个插座看成图中的一个节点，计算出任息两插座间的最短距离，作为两节点间边的权重。要求的布线结果是一个保证连通的子图，其中包含的边的权重和最小，这实际上是一个最小生成树，可以用各种经典的求最小生成树的算法(如Kruskal算法)解决。

实现时注意，因为图中任意两点间都有边,所以采用邻接矩阵表示图比较好。

构建图的过程比较复杂。为了方便地计算两插座间的最短距离，每个插座除了要在数据结构体中记录坐标外，还需要判断它属于哪一面墙。然后根据墙的编号，判断两插座属于下列哪3种情况之一:在同一面墙上、在相邻两面墙上、在对面的墙上。对于任何一种情况，还需要判断它们所在的墙上是否有门。没有门的情况相对简单些，否则还需要考虑绕过门框的布线方法。建议将有门的情况单独抽取出来，写成一个函数，可以被有效地重用。

输出时应注意，题目要求输出整数长度，但不能四舍五人，而必须向上取整，因为电线短一点就不能保证连通了。

## 题目五分 治 法 : 最 小 套 圈 设 计

**題目要求**

套圈游戏是游乐场中常见的游戏之一，其规则为:游戏者将手中的圆环套圈投向场中的玩具，被套中的玩具就作为奖品奖给游戏者。

现给定一个套圈游戏场的布局，固定每个玩具的位置，请你设计圆环套圈的半径尺寸，使得它每次最多只能套中一个玩具。但同时为了让游戏看起来更具有吸引力，这个套圈的半径又需要尽可能大。

把问题进一步简化，假设每个玩具都是平面上的一个没有面积的点，套圈是简单的圆。一个玩具被套住，是指这个点到圆心的距离严格小于圆半径。如果有两个玩具被放在同一个位置，那么输出的圆半径就是0。

**输入要求：**

输入由若干组测试数据组成。

每组数据的第1行包含一正整数N(2<=N<=100000),代表场地中玩具的个数。接下来有N行输入，每行包含一个玩具的x和y坐标。当N为0时，表示全部测试结束,不要对该数据做任何处理。

**输出要求：**

对每一组测试，在一行里输出符合设计要求的套圈的半径，精确到小数点后两位。

**输入例子：**

**2**

**0 0**

**1 1**

**2**

**1 1**

**1 1**

**3**

-1.5 0

0 0

0 1.5

0

**输出例子:**

0.71

0.00

0.75

**分工要求**

•程序员：实现程序。注意源代码必须有充分注释。

•测试员：提供测试输入，并根据测试结果填写测试报告;完成测试结果分析与探讨。

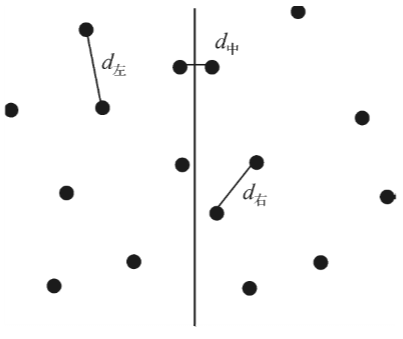
•文档员：完成实验报告第一部分、第二部分内容，要求文档风格统一。

**简要提示**

首先必须认识到，找到最小距离点对之后，将套圈半径取为这个最小距离的一半，就一定满足题目的要求。所以,问题实质上是经典的求最近点对的问题。

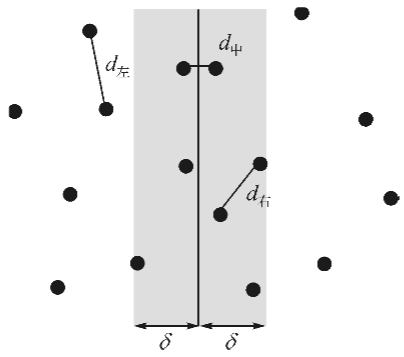
一个最简单的做法是用二重循环把所有可能的点对遍历，计算全部（N-1)N /2个距离，从中找出最小值。这个算法的时间复杂度显然是O(N2) ，当测试数据规模比较大时，会导致超时，必须设计时间复杂度为O(log2N) 的算法。

—种解法是“分而治之”,如图所示。将所有点按其x坐标排序，从中间将场地一分为二，递归地解决两边场地的子问题，分別得到两个子问题的最小半径d左和d右--这个过程是“分”;在所有横跨分界线的点对中找出距离最近的点对，并将这个距离作为d中与两个子问题的解d左和d右进行比较 ， 其中最小的值即为问题的最终解--这个过程称为“治”。这个问题的难点在于“治”,即求出所有横跨分界线的点对中距离最近的点对。如果这个过程的算法复杂度不能减少到O(N)，则整体复杂度将仍然是O(N2)，不能满足要求。



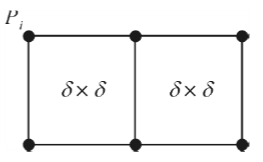
“分而治之”示意图

注意到如果d中真正能起作用，则应有d中<δ=min{d左,d右} ， 只须考虑距中分线δ范围以内的点，如下图中阴影所示的中间带状区域(δ=d右）。同理，中间带内若两点的y坐标之差大于δ则这样的点对也不必考虑，这就将搜索的范围大大缩小了。可将中间带闪的点按其y坐标排序。对任一点Pi，只需向下比较Pj（j>i)，一旦|Pi•y-Pj•y|>δ 则可结束对Pi的考察而转向考察Pi+1。



计算d中所涉及的中间带

由δ的定义可知，对任一Pi，最坏情况下也只须比较5个点(不考虑有重合点的情况），如下图所示。这是因为在左右两个δXδ的正方形区域内，若Pi占据了某个正方形的一角，则该正方形内不可能有点，否则该点与Pi的距离必然小于δ，与δ的定义矛盾。所以最坏情况只有6个点刚好落在两个正方形的角点上，而其中一个点是Pi。



矩形域内点的稀疏性

这样就可以在O(N)时间内解决d中的计算。

还有一点必须注意，不可以在每次递归中都对点的y坐标重新排序，否则整体复杂度还是不能降低。解决方法是在执行分治法之前对点集进行两次预排序，保存两个点列：一是按x坐标排序后的点列，另一为按y坐标排序后的点列。完成预排序后，递归时就可以顺序扫描按y坐标排序后的点列，根据x坐标所在的左右子集划分，将点按其y坐标有序地分别放进左右两个子集中。该过程的时间复杂度是O(N)。

## 题目六 动态规划：商店购物

**题目要求**

某商店促销中为顾客提供各种优惠，把若干种商品分成一组降价销售。例如一朵花原价2元，一只花瓶原价5元，而用优惠券可以用5元买3朵花，用10元买2只花瓶加1朵花。这时顾客买3朵花和2只花瓶只须付14元一一一用第2种优惠组合买2只花瓶加1朵花，再用原价买2朵花，所付费用最少。

请编写程序帮助收银员计算顾客所购商品应付的最少费用。假定顾客手中优惠券都有充分多张，可将同一种券反复用任意多次。例如，给定优惠券上写明3朵花售5元，则顾客买6朵花时可将此券用2次，只需付10元。

注意：收银员不能改变顾客的购物种类和数量，即使增加某些商品会使付款总数减少，也不允许收银员做任何更改。

**输入要求：**

输人由2部分组成。

第1部分包含10000种商品的名称和价格，每一行以下列格式给出：

[商品名称]价格

其中商品名称由不超过60个字符且不包括“[”和“]”的字符串组成，价格是非负实数。接下来的输人由若干组测试数据组成。每组数据包含一张优惠券的N种组合优惠的描述和顾客购物清单。其中优惠券的描述按如下格式给出：

N(<=20）是优惠券中的商品组合方法的个数。下面有N行，每行给出

m [商品2]\*n1 [商品2]\*n2 … [商品m]\*nm 总价格

表示该组合包含m种商品其中商品i必须买ni（<=9）件，总的优惠价格是非负实数总价格。任何一张优惠券中涉及的不同商品数(N种组合涉及的所有商品数）不超过6件。例如，样例输人中的

2

1 [flower]\*3 5.00

2 [flower]\*1 [vase]\*2 10.00

表示该优惠券包含2种组合方式，即：可以用5元买3朵花，用10元买2只花瓶加1朵花。

顾客购物清单的描述按如下格式给出：

L（<=30）是不同商品的个数。下面有L行，每行给出

[商品名称] 购买数量（<=9）

例如输人例子中的

2

[flower] 3

[vase] 2

表示该顾客要买3朵花和2只花瓶。

当N为负数时表示全部测试结束，不要对该数据做任何处理。

**输出要求：**

对每一组测试，在一行里输出顾客应付的最少费用精确到分。

**输入例子：**

[the 1st item] 15.20

[the 2nd item] 80.00

[the 3rd item] 120.00

[vase] 5.00

[flower] 2.00

……

[the 10000th item] 1.00

2

1 [flower]\*3 5.00

2 [flower]\*1 [vase]\*2 10.00

2

[flower] 3

[vase] 2

1

1 [flower]\*3 5.00

1

[flower] 6

4

1 [flower]\*3 6.00

2 [flower]\*2 [vase]\*1 7.00

2 [flower]\*4 [vase]\*3 11.00

2 [flower]\*5 [vase]\*2 9.00

2

[flower] 9

[vase] 5

-1

**输出例子：**

14.00

10.00

20.00

**分工要求**

•程序员：实现程序。注意源代码必须有充分注释。

•测试员：提供测试输入，并根据测试结果填写测试报告;完成测试结果分析与探讨。

•文档员：完成实验报告第一部分、第二部分内容，要求文档风格统一。

**简要提示**

直接对各种优惠组合做深度优先搜索是可以解的，但是效率非常低。有兴趣的同学可以尝试画一下输人例子中的第3组数据的深度优先搜索树，就会发现搜索中存在大量重复。

事实上，这题可以用动态规划法高效地解决。可用一个6维数组存放各种优惠商品的最低价格，即用m[n1][n2]…[n6]记录购买第1种商品n1个、第2种n2个、．．．、第6种n6个的最少费用，则m具有最优子结构性质，可以用动态规划法求解。具体递推公式为：

其中，是不用优惠券直接购买的总费用(pi是第i种商品的价格），可以作为m的初始值。是指在使用第k种优惠组合之前的最优解，为第k种优惠组合的总价格。因而，当前最优解m[n1][n2]…[n6]就在使用或不使用下一个优惠券组合的所有可能中取最小值。

另外必须注意到，解题过程中需要反复找到某商品的原定价格，所以在开始进人动态规划之前，必须将全部商品按其名称排序，这样每次要查找该商品信息时，使用二分查找就可以了。

## 题目七：熊猫烧香

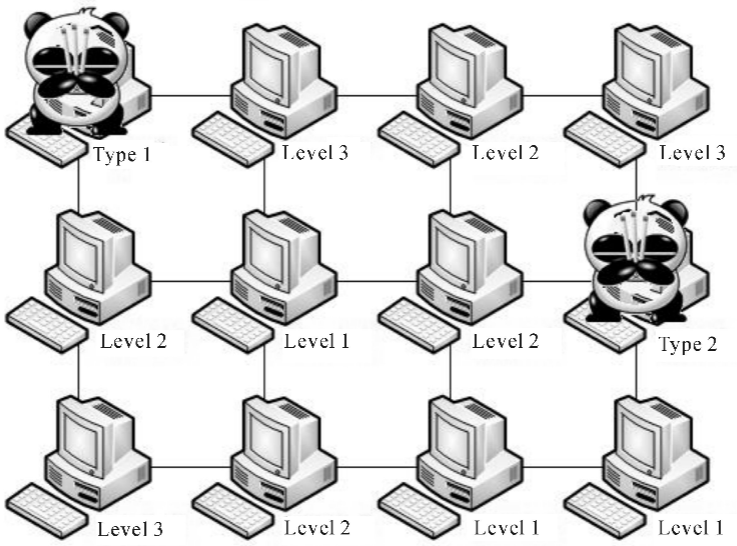
**题目要求**

“熊猫烧香”是在网络中传播的一种著名病毒，因为图标是一只可爱的熊猫而得名。该病毒比较难以处理的一个原因是它有很多变种。

现在某实验室的网络就不幸感染了这种病毒。从下图中可以看到，实验室的机器排列为一个M行N列的矩阵，每台机器只和它相邻的机器直接相连。开始时有T台机器被感染，每台遭遇的熊猫变种类型都不同，分别记为Type1，Type2，… Typer，每台机器都具有一定级别的防御能力，将防御级别记为L(0<L<1000）。“熊猫烧香”按照下列规则迅速在网络中传播：

* 病毒只能从一台被感染的机器传到另一台没有被感染的机器。
* 如果一台机器已经被某个变种的病毒感染过，就不能再被其他变种感染。
* 病毒的传播能力每天都在增强。第1天，病毒只能感染它可以到达的、防御级别为1的机器，而防御级别大于1的机器可以阻止它从自己处继续传播。第D天，病毒可以感染它可以到达的、防御级别不超过D的机器，而只有防御级别大于的机器可以阻止它从自己处继续传播。
* 在同一天之内，Type1变种的病毒先开始传播，感染所有它可能感染的机器，然后是Type2变种、Type3变种……依次进行传播。

以下图为例说明传染的过程。



“熊猫烧香”传播示意图

上图中显示的3×4的网络中，开始阶段只有2台机器被感染。用一个矩阵表示网络中机器的状态，用负整数-L表示未被感染的、防御级别为L的机器，正整数Typei表示该机器被Typei类型的病毒变种感染，则初始状态有矩阵：。

病毒传播1天后，1号变种无法传播，2号变种攻下了第3行中2台防御级别为1的机器，矩阵变为。第2天，1号变种攻下了所有未被感染的、防御级别为1或2的机器，2号变种则无事可做，因为它唯一可以接触到的未被感染的机器，其防御级别是3。这时矩阵变为：。第3天,1号病毒继续发威，攻下了剩下的3台防御级别为3的机器，则整个网络全被感染，矩阵变为。

本题的任务是：当整个网络被感染后，计算有多少台机器被某个特定变种所感染。

**输入要求：**

输入由若干组测试数据组成。

每组数据的第1行包含2个整数M和N(1<=M，N<=500），接下来是一个M\*N的矩阵表示网络的初始感染状态，其中的正、负整数的意义如题目描述中所定义。

下面一行给出一个正整数Q，是将要查询的变种的个数。接下去的Q行里，每行给出一个变种的类型。

当M或N为0时，表示全部测试结束，不要对该数据做任何处理。

**输出要求：**

对每一组测试，在一行里输出被某个特定变种所感染的机器数量。

**输入例子：**

3 4

1 -3 -2 -3

-2 -1 -2 2

-3 -2 -1 -1

2

1

2

0 0

**输出例子：**

9

3

**分工要求**

•程序员：实现程序。注意源代码必须有充分注释。

•测试员：提供测试输入，并根据测试结果填写测试报告;完成测试结果分析与探讨。

•文档员：完成实验报告第一部分、第二部分内容，要求文档风格统一。

**简要提示**

这道题有多种解法，不同解法之间的计算复杂度可以差大约L倍，而L可以取到999。

解法一：广度优先搜索

这种想法比较直接，每天依次按照病毒变种类型的顺序，以已经被这种变种感染了的区域为起点，在未感染的区域进行一次广度优先搜索，被搜索到的节点如果可被感染，则记录为感染了该种病毒。这样L天之后，所有的计算机都被感染上了病毒，统计一下各种病毒的数目就是答案。这种解法的最坏时间复杂度O(M\*N\*L).

解法二：修改Dijkstra算法

Dijkstra算法是求单源最短路径的一种贪心算法。在这里将每台机器作为一个节点V，将每个节点的路径长度distance[V]的定义改造一下，它不再是一个简单的数字，而是一个二元组(day，vid）其中day表示被感染的日子，vid表示被哪个病毒感染，比较路径长度时，按day为第一关键字，vid为第二关键字排序。

初始时，有第i个病毒变种的机器的路径长定义为（0，i)，表示它第0天被第i个变种感染。其他节点的值初始化为(∞, ∞),然后利用Dijkstra算法求解所有其他机器的路径长，每次选取当前路径长最小的那个节点V，然后更新与它相临的节点U。如果U路径长的day值较大，则将day值更新为max{U的防御级别，V点的day值}。如果day值相等，则将vid更新成较小的。

由于是平面点阵图，所以这个算法的复杂度就是O(MNlog2(MN))，较解法一有了明显改进。

解法三：并查集（union-find set）

将初始被感染上病毒的节点作为根节点。如果一个新的节点被某种病毒变种感染了，则利用并查集算法将该节点的指针指向相应的根节点。

如果出现一块防御等级均不超过i的区域，区域的周围被防御等级大于i的节点包围，而这个区域之内没有初始感染病毒的基本根节点，那么，可以在这个区域之内任意取一个节点作为虚拟根节点，将区域内的所有节点指向这个虚拟的根节点。这样一旦某种病毒变种感染了这个区域的任何一个节点，则会立即感染这个区域的每一个节点。于是，只需将这个虚拟根节点指向初始感染病毒的基本根节点即可。

这样，第i天时，所有防御等级小于等于i的节点均能通过并查集，指向相应的根节点。

并查集的union函数部分要特别注意，虚拟根节点和基本根节点之间的指向很容易搞错。

这种解法的时间复杂度是O(MNlog\*(MN))。注意log\*N是逆Ackermann函数，即对N反复取对数直到结果不超过1时，取对数的次数就是log\*N,例如log\*265536=5，因为log2(log2 (log2 (log2 (log2265536))))=1。log\*N函数增长非常慢，所以可以近似认为是一个常数。

## 题目八：神秘国度的爱情故事

**题目要求**

某个太空神秘国度中有很多美丽的小村，从太空中可以望见，小村间有路相连，更精确一点说，任意两村之间有且仅有一条路径。

小村A中有位年轻人爱上了自己村里的美丽姑娘。每天早晨，姑娘都要去小村B里的面包房工作，傍晚6点回到家。年轻人终于决定要向姑娘表白，他打算在小村C等着姑娘路过的时候把爱慕说出来。问题是，他不能确定小村C是否在小村B到小村A之间的路径上。你可以帮他解决这个问题吗？

**输入要求：**输入由若干组测试数据组成。

每组数据的第1行包含一正整数N(1<=N<=50000），代表神秘国度中小村的个数，每个小村即从0到N-1编号。接下来有N-1行输人，每行包含一条双向道路的两个端点小村的编号，中间用空格分开。

之后一行包含一正整数M(1<=M<=500000），代表着该组测试问题的个数。接下来M行，每行给出A、B、C三个小村的编号，中间用空格分开。

当N为0时，表示全部测试结束，不要对该数据做任何处理。

**输出要求：**对每一组测试给定的A、B、C，在一行里输出答案，即：如果C在A和B之间的路径上，输出Yes，否则输出No。

**输入例子：**

3

0 1

1 2

3

0 2 1

1 2 0

1 2 1

0

**输出例子：**

注意：该题目应设计大规模的测试数据，所以用C语言中的scanf和printf做输入输出会比用cin和cout快，可以避免因为输入输出而超时。

**分工要求**

•程序员：实现程序。注意源代码必须有充分注释。

•测试员：提供测试输入，并根据测试结果填写测试报告;完成测试结果分析与探讨。

•文档员：完成实验报告第一部分、第二部分内容，要求文档风格统一。

**简要提示**

注意到条件“任意两村之间有且仅有一条路径”表明这是一棵N(1<=N<=50000）个节点的树，每次查询给定点C是否在其余两点A、B之间的路径上。查询的次数高达500000次，因此对于每次查询，超过O(log2N）的复杂度是不能接受的。

最直接的解法是沿着A、B点往上找，直到相遇或者碰到C，不过这样对于全部节点在一条线上的树，每次查询的复杂度是O(N)，肯定超时。

仔细观察，可以发现如果点C在A、B之间的路径上，那么它满足下面这个有趣的规律：

点C在A、B之间的路径上当且仅当C仅是A、B其中一个节点的祖先一一一除了一个非常特殊的情况，就是当C是A、B两点的最低公共祖先时，点C也在A、B的路径上（其实这道题的关键就是判断这个特殊情况）。

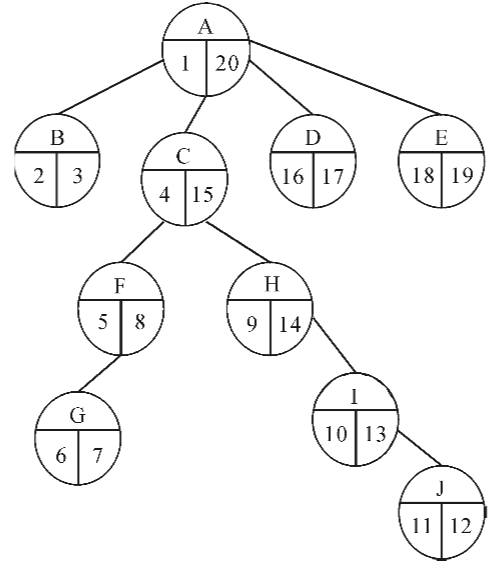
因此，得到如下的算法：判断点C是否仅是其中一个节点的祖先。如果是，那么c肯定在路径上；否则，如果C是A、B两点的共同祖先，则判断C是否为最低公共祖先，如果是，那么C肯定在路径上，否则C不在路径上。

那么现在剩下两个问题：

* 如何快速判断一个点是否是另外一个点的祖先？
* 如果C是A、B两点的共同祖先，如何快速判断它是否是最低的?

对于第一个问题，可以用深度优先搜索遍历一遍，记录每个节点的入栈时间及出栈时间，然后判断其包含关系。

例如：下图是一棵深度优先搜索树，每个节点左下角的数字表示第一次访问该节点即入栈时间，右下角数字表示离开该节点即出栈时间。要判断点A是否是点C的祖先，只要判断区间[1，20]是否包含了区间[4，15]，因此每次判断的复杂度是O（1）的。



记录出入栈时间的深度优先搜索树

对于第二个问题，例如要判断A是否是G和J的最低公共祖先，其实就是看是否有比A更低的祖先，如果有则说明A不是最低的。由于A下面还有C这个节点，因此可以得出A不是G和J的最低公共祖先，而C则找不到这样的一个儿子节点，因此它是最低的。

很自然想到的方法是遍历A的所有儿子节点，逐个判断它是否是G和J的公共祖先。但是这样有可能退化成O(N)的复杂度。

仔细观察这棵深度优先搜索树，又可以发现一个非常有趣的规律：

如果从左到右列出A的四个儿子节点B、C、D、E的入栈时间及出栈时间：[2，3]、[4,15]、[16,17]、[18,19]，不难发现这个区间数列是递增的！于是得到了一个更快的方法：

只要在这个递增的区间数列中二分查找是否有[6,12]这个区间即可（6是G的人栈时间，12是J的出栈时间），因此复杂度就降到了O(log2N)。