浅谈类比思想

长沙市长郡中学周戈林

【目录】

【摘要】

信息学是一门变幻莫测的艺术，它包含着海量的知识点。我们不能奢求掌握所有的知识，只能在已有知识的基础上，尽可能的把不熟悉的问题转化为熟悉的问题。类比思想，就是一种非常优秀的转化方法。本文尝试诠释一些常用的类比模式。

【关键字】

类比思想 模型 算法 理性认识

【正文】

一、引言：

类比是最有创造力的一种思维方法。它关注两个对象在某些方面的相同或相似，从而推测它们在其它方面也可能存在相同或相似之处。举例来说，我们在小学一年级学到正确的握铅笔方法是“笔杆放在拇指、食指和中指的三个指梢之间。食指在前，拇指在左后，中指在右下，食指应较拇指低些，手指尖应距笔尖约 3 厘米。笔杆与作业本保持六十度的倾斜，掌心虚圆，指关节略弯曲”。学会了握铅笔，那么在三年级也可以用类似的方法使用钢笔书写。概括一下，这次握笔类比的形式为：

对象 A 具有性质 P、Q；

对象 A’具有性质 P’（P 与 P’类似）；

对象 A’可能具有性质 Q’（Q 与 Q’类似）。

拿握笔来说，铅笔（对象 A）笔杆比较细（性质 P），所以我们采用上述“笔杆放在三个指梢之间”的方法握笔（性质 Q）；而钢笔（对象 A’）笔杆也比较细（性质 P’），所以我们采用同样的方法握笔（性质 Q’）。很幸运，这次类比是正确的，我们成功地学会了写字。

但有些时候就没那么幸运了，譬如说，当面对一支毛笔时，以上的握笔方法写出的字就会产生相当的幽默效果。为什么我们的握笔方法面对毛笔失败了呢？这是因为毛笔是软笔，并且笔杆粗细不同，因此类比失败了。正确的握毛笔方法是用拇指和食指捏住笔的上端，用中指和无名指活动笔的下端，小指随无名指自

然活动。概括这次握笔方法的转换，就是：

对象 A 具有性质 P、Q 和关系 R；

对象 A’具有性质 P’；

对象 A’具有性质 Q’和关系 R’。

具体到握毛笔这个例子，铅笔（对象 A）是笔（性质 P），并且是硬笔（关系 R），需要用三根手指托笔（性质 Q）。而毛笔同样是笔（性质 P’），但却是软笔（关系 R’），只需要两根手指夹笔（性质 Q’）。这种类比形式考虑到了性质之间的关系，因此准确性提高了。

总结一下对握笔的研究：第一次握笔类比关键在于铅笔和钢笔恰好都是硬笔，因此其成功具有偶然性，它是基于直观上的感性认识，称之为简单类比；第二次握笔类比注意到铅笔与毛笔的不同点，其成功带有某种必然性，它是基于逻辑上的理性认识，称之为科学类比。在信息学竞赛中需要的类比，往往是科学类比。

下文将试图论述一些常见的类比模式：具体事物类比抽象模型；相似算法之间的类比；图形类比数式。

二、常见的类比模式：

2.1 具体事物类比抽象模型：

这是一种最常见的类比。现实事物不是严格的数学模型，在研究它们的过程中必须根据需要提炼相应的数学模型，否则便失去了建模的意义。在建模中要联想具体事物的固有属性和抽象模型的独有特点，才能恰如其分地建立模型和解决问题。

举例来说：研究地球的公转可以把地球看成质点，这是因为相对于公转半径来说地球半径极其微小，可以忽略。但是研究地球的自转时又不能忽略地球半径，这是因为地球半径比起质点来又远远大得多了。

从下面这个例子也可以看到，建模的角度不同，效果便截然不同。

例一：山顶问题

✧ 题目描述

奶牛成群、土地众多的 FJ 有一个地形狭长的农场，农场被分成了 *n* 块土地，

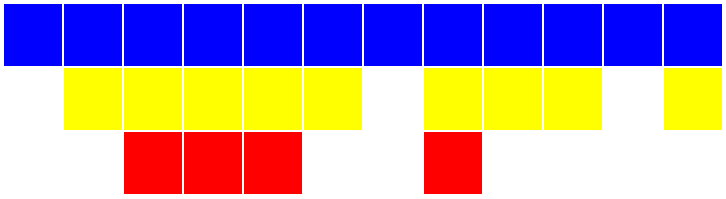
n 不超过 1000。这些土地位于一条直线上，并从左到右编号为 1 至 *n*。每块土地的面积都相同，但是高度不一定相同。每块土地都拥有一个海拔高度值，这个值不超过 1000000。如果一段相同高度土地的两边都比它低或者是农场的边界，那么这段土地将被称之为“山顶”。FJ 希望通过搬走泥土来降低某些土地的海拔高度，使“山顶”的数目不超过 *k*，其中 $1 \leqslant k \leqslant 25$。在这一前提下，FJ 希望搬运的泥土体积最小，也就是所有的土地减少的高度和最小。

✧ 解法分析

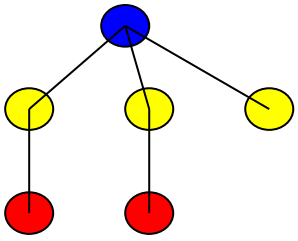
题目中要求了一个很奇怪的“削平山顶”的任务。一个或几个“山顶”往往由一个“山脊”支撑。值得注意的是，“山顶”被削平后有可能会使“山脊”变成“山顶”。

从解题的一般感觉上看，这似乎是一道动态规划的题目，尝试着用动态规划来解这道题目。根据一般的状态设计方法，我们用 *f(i,j)*表示前 *i* 块土地留 *j* 个“山顶”的需要搬走泥土的最小体积。但是光用这两个值无法完全描述出当前土地的高度，因此也无法得知它们对以后状态的影响。为了能准确表述土地的高度，我们需要记录一个描述高度的序列集合，但存储这个集合的费用是让人无法忍受的。

换一种思维看问题，不妨把整个农场 180 度翻转，使“山顶”朝下。以下是翻转样例得到的图形：



一块土地按照高度被剖分成了最多 n 个层面，每个层面按照高度被染成了不同的颜色。当然，较高的层面需要较低的层面支撑。而如果一个层面两边的层面都比它低，那么就是“山顶”。把层面看作结点，在相互支撑和被支撑的层面之间连一条边。让人惊讶的是，这是一棵有根树：



为什么会是一棵有根树呢？这一点很好解释：每个层面都只跟它支撑的层面与支撑它的层面相连。一个层面可以支撑多个层面，但是每个层面都只能被一个层面支撑，这正好与树的定义（每个结点有多个后继，但除根结点以外只有一个前趋）类似。

既然农场可以类比成一棵有根树，那么“山顶”在树中又对应着什么呢？显然“山顶”是不能支撑其它层面的，否则其它层面才是“山顶”。由于“山顶”没有“后继”层面，它就对应着树中的叶子结点。而搬走泥土相当于把结点删除，搬走泥土的体积就是删除这个结点的费用。

至此已经完全把题目类比转化成一个我们非常熟悉的问题：给定一棵有根树，每个结点有一个权值。删除某些结点，使删除后树的叶子数目不超过 K，在这一前提下删除的结点权值和最小。特别的，根结点永远不算叶子。这是一个经典且基础的树形动态规划问题，可以用 $O(nk^2)$的算法解决，在此略去。

于是问题转化为如何高效、优美地建树。以下介绍一个算法，利用栈在 $O(n)$ 时间内建树和将树转化为左儿子右兄弟形式，同时求出其拓扑序。

栈内每个元素记录着一段高度相同的连续土地，也就是一个层面。从左到右扫描整个农场，每扫描到一块新的土地就分情况处理：

1．这块土地的高度大于栈顶层面，直接将其入栈；

2．这块土地的高度等于栈顶层面，将栈顶层面土地块数加一；

3．这块土地的高度小于栈顶层面，表明已经出现了一个栈顶层面支撑的层面已经扫描完毕，可以导出成为一个新的结点了，遂退栈。当前扫描的土地再与新的栈顶元素比较。

同时为了求出结点的权值与结点间的连结关系，还要分配两个额外的域记录层面的高度与支撑的结点。算法的具体实现可以参见源程序。由于每个层面只有当它支撑的层面都被处理后才会被导出，因此顶点标号是满足动态规划计算的拓扑顺序的。而每块土地都最多进栈一次，出栈一次，时间复杂度为 $O(n)$。

整个算法的时间复杂度即为 $O(nk^2)$，对于题目中的数据范围绰绰有余。

小结

本例尤其体现了处理具体事物时模型选取的重要性。题目给出的 n 块土地是线性排列的，给人的第一感是建立线性的动态规划模型。但我们经过仔细分析，

发现线性模型对问题的研究没有很大帮助。与此相反，我们把农场按高度剖分成层面，层面与节点类比，层面的连接关系与有根树的性质类比，关键的“山峰”正好对应了树的叶子。复杂的“移山”就变成了简单的删除节点。

2.2 相似算法之间的类比：

有些算法是相似的：有的在算法思想上相似，有的在算法依据上相似，有的在算法实现上相似。因此我们可以对某些算法进行改造，利用与其它算法的相似性，扩展功能和提高效率。

类比思想在算法改造的过程中起到相当大的作用。通过类比，就可以发现算法的相似之处；通过类比，就可以分析相似点的本质；通过类比，就可以对算法进行组合应用。

例二：最小最大边问题

✧ 题意描述

有 n 个城市，p 条双向道路把这些城市连接起来，一对城市之间可能有多条道路连接。FJ 要找到 t 条从城市 1 到城市 n 的路径，不同的路径不能包含相同的道路。在这一前提条件下，FJ 希望所有路径中经过的最长的道路最短。

✧ 解法分析

很明显这是一个关于流的问题。题目给定 n 个点和 p 条容量为 1 的无向边，每条边都拥有一个边权，要求找到一个流量至少为 t 的流，同时流通过的边权最大的边最小。很容易就能得到如下的朴素算法：二分枚举最长边的长度 m，忽略长度超过 m 的边，求出此时的最大流 t'。若 $t'<t$，则 m 应增加；否则应减少，同时用 m 更新当前最优解。求最大流的费用为 $O(tp)$，因此这个算法的时间复杂度为 $O(tplog_2p)$。

这个算法已经很不错了，但是有没有更好的算法呢？本题是一个要求最小权的流问题，这让人想到了另外一个跟边权有关的流问题——最小费用最大流问题。最小费用最大流问题要求流通过每条边的流量与边权乘积总和最小，本问题要求最大边权最小。再回顾一下两个似乎跟本题无关的问题：单源最短路问题，它要求找到一棵生成树，使源点通过树边到达其它点经过的边权和最小；最小生成树问题，它同样要求找到一棵生成树，使任意点通过树边到达其它点经过的边中权最大的最小。两组问题都是“一个要求总和最小，另一个要求最小边最小“这

种形式，解决问题的算法也应该有相似的地方。

求最小费用最大流的经典算法是连续最短路算法：根据当前流的分布设定费用函数，不断找到费用最小的增广链增广，直到找不到增广链为止。这时我们不禁猜想，如果把求最短路的部分用求最小生成树的算法替代会如何？

联合后的算法如下：仍然是不断寻找增广轨增广。寻找增广轨时设立临时距离标号 dist[*i*]，表示当前能扩展到 *i* 的增广轨中最长边长度的最小值，初始时除源点以外的临时距离标号都为正无穷大。在计算距离标号时，假设 dist[*u*]已经被扩展，正在考察边(*u*,*v*)：

1.若 *u* 到 *v* 的流量为 0 且 *v* 到 *u* 的流量为 0，那么 dist[*v*] ← min{dist[*v*],max{dist[*u*],w(*u*,*v*)}}；

2.若 *v* 到 *u* 的流量为 1，那么 dist[*v*]←min{dist[*u*],dist[*v*]}；

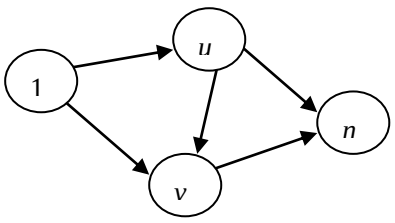
以上两种处理方式分别对应了增流和减流的情况。以下是对算法正确性的证明：

引理 1：在算法依次找到的每条增广路中，*n* 的距离标号是单调不减的。

证明：根据最短增广路原则，算法优先扩展最短的增广路。若存在增广路 *Path* 与 *Path'*，dist(*Path*)<dist(*Path'*)，则 *Path* 必在 *Path'*前被扩展，所以 *n* 的距离标号必然单调不减。

引理 2：扩展方式 2 不会使当前流经过的最长边的值变小。

证明：如下图，假设原来存在一条流的路径 1→*u*→*v*→*n*，方式 2 将其扩展成 1→*v*→*n* 和 1→*u*→*n*。若 1→*u*、1→*v*、*u*→*n*、*v*→*n* 四条路径中存在长度不小于 w(*u*,*v*)的边，那么最长边边权的值不会减少；但如果所有边权都小于 w(*u*,*v*)，那么根据引理 1，算法会优先选择 1→*v*→*n* 和 1→*u*→*n* 两条路径，不会从(*u*,*v*)经过，这与假设矛盾。综上可知引理 2 成立。



定理：此算法是正确的。

证明：根据引理 1 我们知道算法在贪心式地寻找增广路，而根据引理 2 我们知道算法得到的永远是当前流量下的最优解。因此算法是正确的。

综上可知算法是正确的。注意到修改过的距离标号不可能为负，而边权都为正数，可以直接套用普里姆算法计算距离标号。在题目中的数据范围我们选择最简单的 $O(n^2+p)$ 的算法。当图为稀疏图时，可以利用堆使复杂度降为 $O(n\log_2 n+p\log_2 n)$。距离标号需要求 t 次，总时间复杂度仅为 $O(t(n^2+p))$。

小结

在本题的解决中，作者抓住两组算法的相似之处进行类比，取得了相当不错的效果。这提示我们，在学习过程中对知识的归纳和总结是很有必要的。只有掌握的知识形成了一定体系，才能灵活地对问题进行类比，才能从类比中提炼优秀的算法。

2.3 图形类比数式：

从感性上去认识，图形和数式差异很大。但有时候忽略图形的次要信息而将主要信息转化为数式，把难以处理的几何问题转化为较易处理的代数问题，反而能对解题产生帮助。

这种类比模式的关键在于区分什么是图形中的主要信息，什么是次要信息。对于不同的问题，其主要信息与次要信息是不同的。例如当求多变形质心时必须知道每个点的坐标，但是求周长时知道每条边的长度就可以了。一般来讲，对解答问题有帮助的信息才叫做主要信息。下文的例子就体现了舍弃次要信息的必要性。

例三：点集同构问题

✧ 题意描述

给定包含 n 个点的集合 S 和 S'中每个点的坐标，判断它们是否同构。两个点集同构，指它们在旋转、翻转、平移、缩放后能互相重合。

✧ 解法分析

本题给人的第一感觉就是条件太多太宽了。翻转操作好处理，因为只能翻或者不翻。但是旋转、平移、缩放的操作都有无限种可能。

关注点集中的不变量质心。设 n 个点的坐标分别为$(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$，则质心 O 坐标为

$$(\frac{\sum_{i=1}^{n} xi}{n}, \frac{\sum_{j=1}^{n} yj}{n})$$

下文可以看到，求出质心的好处在于可以将相对于直角坐标系的的坐标转化成相对于质心的量。

缩放是对点集整体缩放。假设点集中任意一点与质心间都有线段连接，并且点 *P* 缩放后变为 *P'*，则有下列恒等式：

$$\frac{OA}{OB} = \frac{O'A'}{O'B'}$$

其中 *A,B* 是点集 *S* 中任意两点。这个式子表明无论如何缩放，点集中任意两条线段长度之比是恒定的。也就是说

$$\frac{OA}{OA'} = \frac{m}{m'} = k$$

其中 *m* 是点集中所有线段长度的最大值。易知这个最大值可以在 O(*n*)的时间内求出。由此我们便求出了 *S* 到 *S'*的缩放比 *k*，按照缩放比将坐标扩大 *k* 倍即可。

现在到了关键的部分，注意每个点 *P* 与质心的关系，点 *P* 可以看作从质心出发，引一条长度为|*OP*|的线段，再从与横轴平行的方向顺时针旋转 *alpha* 度（也就是说 *P* 关于 *O* 的方向角是 *alpha* 度）。把所有的点按照方向角优先、到质心距离次优先的规则进行排序，那么一个点集就可以看成 *n* 次操作，每次操作将一条一个端点在 *O* 的线段顺时针旋转一个量，再拉长或缩短。一次操作可以用一个二元组(*delta,distance*)来描述，其中 *delta* 就是排序后相邻两点的方向角之差。这样一个 *n* 个点的点集 *S* 就被对应成了一个长度为 *n* 的串 *T*。并且点集可以旋转，也就是串的起始位置不确定，是一个环串。因此两个点集同构，即是它们转换后得到的环串相同。而判断环串 *T* 与 *T'* 是否相同，可以把 *T* 和 *T'*从某个地方截断得到 *U* 和 *U'*，再使用 kmp 算法判断 *U'*在串 *UU* 中是否出现即可。

概括一下上面的讨论：求质心的费用为 O(*n*)，排序的耗费为 O($nlog_2n$)，执行 kmp 算法的费用为 O(*3n*)，最后翻转再判断的费用为 O(*n*)，因此整个算法的复杂度就是 O($nlog_2n$)。

有一点需要补充的是，以上的讨论都忽略了某个点与质心重合的情况，但处理此问题只需要特判即可。

小结

本题的关键是处理旋转和平移。我们忽略图形的次要信息——绝对坐标，把关注的重点变为点跟质心的关系以及点跟点的关系，因此平移问题就不存在了，用环串类比旋转操作自然就产生了最后的解法。

三、总结：

本文对类比作了一定的介绍，同时较深入地讨论了一些常见的类比模式。以上所有的解题方法都包含了一种思想，就是类比思想。类比思想是一种转化的思想，但是类比思想在转化过程中突出的不是演绎式的化归，而是跳跃式的比较。固然，类比在思考过程中可能是逐步进行的，但是在外在表现上一般是天马行空的。从这一点上来说，类比是一种相当不好掌握的思想方法。但无论怎么样的类比，都会有以下的特性：

1．可类比性：原问题和转化后的问题具有某些相似之处，正是这些相似之处为类比提供了理论依据和思考动机。

2．可简化性：转化后的问题必然要比原问题更简单。这个“简单”可以是更直观，便于解题者思考；可以是更常见，能利用的理论较多；也可以是更规范，能用标准的方法解决。

3．可移植性：转化后问题的解法要与类比对象密切相关。例如在例三中把点集类比为环串后，要密切注意串的匹配、求最长某某子串这一类算法。解法与类比对象相关，才能体现出类比的必要与优越。

尽管类比思想非常优秀，但并非没有局限性。类比要在转化的前提下才能完成，也就是说对于那些不需要转化的问题，类比是没有任何作用的。

四、感谢：

衷心感谢向期中老师在我写这篇论文时对我的指导和帮助。

衷心感谢刘汝佳教练对我的指导和启发。

衷心感谢陈启峰、杨沐、郭华阳、杨浩、冯子明、吴戈、袁昕颢、周玉姣、刘靓等同学在临近期末考试时还能帮助审阅我的论文，并提出很多宝贵的意见。

五、参考文献：

[1]《数学闯关——数学思想和方法的领悟》 袁银宗编著

[2]《算法艺术与信息学竞赛》 刘汝佳 黄亮 著

[3] USACO OPEN05： peaks

[4] USACO FEB05： secret

[5] Poland Olympiad of Informatics 2005 Stage I：PUN

六、附录：

例一的原题：

Problem 4: Landscaping [Brian Dean, 2005]

Farmer John is making the difficult transition from raising mountain goats to raising cows. His farm, while ideal for mountain goats, is far too mountainous for cattle and thus needs to be flattened out a bit. Since flattening is an expensive operation, he wants to remove the smallest amount of earth possible.

The farm is long and narrow and is described in a sort of two-dimensional profile by a single array of N (1 <= N <= 1000) integer elevations (range 1..1,000,000) like this:

1 2 3 3 3 2 1 3 2 2 1 2,

which represents the farm's elevations in profile, depicted below with asterisks indicating the heights:

```
    * * *       *
  * * * * *   * * *   *
* * * * * * * * * * * *
1 2 3 3 3 2 1 3 2 2 1 2
```

A contiguous range of one or more equal elevations in this array is a "peak" if both the left and right hand sides of the range are either the boundary of the array or an element that is lower in elevation than the peak. The example above has three peaks.

Determine the minimum volume of earth (each unit elevation reduction counts as one unit of volume) that must be removed so that the resulting landscape has no more than K (1 <= K <= 25) peaks. Note well that elevations can be reduced but can never be increased.

If the example above is to be reduced to 1 peak, the optimal solution is to remove 2 + 1 + 1 + 1 = 5 units of earth to obtain this set of elevations:

```
     * * *       -
   * * * * *    - - -    -
* * * * * * * * * * * *
1 2 3 3 3 2 1 1 1 1 1 1
```

where '-'s indicate removed earth.

PROBLEM NAME: peaks

INPUT FORMAT:

* Line 1: Two space-separated integers: N and K

* Lines 2..N+1: Each line contains a single integer elevation. Line
  i+1 contains the elevation for index i.

SAMPLE INPUT (file peaks.in):

```
12 1
1
2
3
3
3
2
1
3
2
2
1
2
```

INPUT DETAILS:

This is the example used above.

OUTPUT FORMAT:

* Line 1: The minimum volume of earth that must be removed to reduce
  the number of peaks to K.

SAMPLE OUTPUT (file peaks.out):

5

例二的原题：
Problem 2: Secret Milking Machine [Vladimir Novakovski, 2003]

Farmer John is constructing a new milking machine and wishes to keep it secret as long as possible. He has hidden in it deep within his farm and needs to be able to get to the machine without being detected. He must make a total of T (1 <= T <= 200) trips to the machine during its construction. He has a secret tunnel that he uses only for the return trips.

The farm comprises N (2 <= N <= 200) landmarks (numbered 1..N) connected by P (1 <= P <= 40,000) bidirectional trails (numbered 1..P) and with a positive length that does not exceed 1,000,000. Multiple trails might join a pair of landmarks.

To minimize his chances of detection, FJ knows he cannot use any trail on the farm more than once and that he should try to use the shortest trails.

Help FJ get from the barn (landmark 1) to the secret milking machine (landmark N) a total of T times. Find the minimum possible length of the longest single trail that he will have to use, subject to the constraint that he use no trail more than once. (Note well: The goal is to minimize the length of the longest trail, not the sum of the trail lengths.)

It is guaranteed that FJ can make all T trips without reusing a trail.

PROBLEM NAME: secret

INPUT FORMAT:

* Line 1: Three space-separated integers: N, P, and T

* Lines 2..P+1: Line i+1 contains three space-separated integers, A_i, B_i, and L_i, indicating that a trail connects landmark A_i to landmark B_i with length L_i.

SAMPLE INPUT (file secret.in):

7 9 2
1 2 2

2 3 5
3 7 5
1 4 1
4 3 1
4 5 7
5 7 1
1 6 3
6 7 3

OUTPUT FORMAT:

* Line 1: A single integer that is the minimum possible length of the
longest segment of Farmer John's route.

SAMPLE OUTPUT (file secret.out):

5

OUTPUT DETAILS:

Farmer John can travel trails 1 - 2 - 3 - 7 and 1 - 6 - 7. None of
the trails travelled exceeds 5 units in length. It is impossible
for Farmer John to travel from 1 to 7 twice without using at least
one trail of length 5.

例三的原题：
Task: pun

Points

I stage competition

Source file: pun.*xxx* (*xxx*=pas,c,cpp)

Memory limit: 32 MB

Alternative formats: PostScript | PDF

A set of integer points on a plane (points whose both cartesian coordinates are integers) which we shall refer to as *the pattern*, as well as a group of other sets of integer points on the plane are given. We would like to know which of the sets are similar to *the pattern*, *i.e.* which of them can be transformed by rotations, translations, reflections and dilations so that they are identical to *the pattern*. For instance: the set of points *{(0,0), (2,0), (2,1)}* is similar to the set *{(6,1), (6,5), (4,5)}*, it is however not similar to the set *{(4,0),(6,0), (5,-1)}*.

Task

Write a programme which:

reads from the standard input the description of the pattern and the familly of the investigated sets of points,
determines which of the investigated sets of points are similar to the pattern,
writes the outcome to the standard output.

Input

In the first line of the standard input there is a single integer *k (1 <= k <= 25.000)* - the number of points the pattern consists of. In the following *k* lines there are pairs of integers, separated by single spaces. The *i+1st* line containts the coordinates of *ith* point belonging to the pattern: *x_i* i *y_i* (*-20.000 <= x_i, y_i <= 20.000*). The points forming the pattern are pairwise different. In the next line there is the number of sets to be investigated: *n* (*1 <= n <= 20*). Next, there are *n* descriptions of these sets. The description of each set begins with a line containing a single integer *l* - the number of points belonging to that particular set (*1 <= l <= 25.000*). These points are described in the following lines, a single point per line. The description of a point consists of two integers separated by a single space - its coordinates *x* i *y* (*-20.000 <= x, y <= 20.000*). The points which belong to the same set are pairwise different.

Output

Your programme should write to the standard output *n* lines - one for each of the investigated sets of points. The *i*th line should contain the word ***TAK*** (*YES* in Polish), if the *i*th of the given sets of points is similar to the pattern, or the word ***NIE*** (*NO* in Polish) if the set does not satisfy this condition.

Example

For the input data:

```
3
0 0
2 0
2 1
2
3
4 1
6 5
4 5
3
4 0
6 0
5 -1
```

the correct outcome is:

```
TAK
NIE
```