浅谈特殊穷举思想的应用

——河北唐山一中　鬲融

目录

浅谈特殊穷举思想的应用

——河北唐山一中　鬲融

【摘要】

本文分别就完全穷举和部分穷举思想的一些特殊应用，结合具体范例进行了分析。认为穷举是我们解题时的有效思想，我们应对其给予充分的重视。对于完全穷举，可以通过适当选取穷举对象来达到我们所希望的结果；而部分穷举思想则是解决最大最小问题的有力武器。部分穷举思想还扩展了图论、贪心等算法的应用范围，而且它的效率一般也是可以满足要求的。

【关键字】

穷举　完全/部分穷举　参变量法　最大最小问题　最大最小匹配

【引言】

穷举思想是信息学中最重要的思想之一，计算机的高速度使其具备了进行穷举的条件。然而，随着图论、数论、动态规划等方法的发展，以及搜索算法的不断改进，穷举似乎越来越不受重视，成为了‘低效’的代名词。但是，穷举的思想仍有许多应用。本文就穷举思想的一些特殊应用，结合一些范例进行了分析。

【正文】

1.穷举的思想

所谓穷举思想，顾名思义，就是把所有可能（有时包括部分不可能）的情况列举出来，然后进行处理的一种思想。

穷举分为完全穷举和部分穷举两种。其中完全穷举由于可能产生许多不必要的情况，往往被各种搜索算法所取代，但我们也将看到，完全穷举不一定比有许多剪枝的搜索慢。部分穷举则是针对这样一个问题：在这种问题中，有一个很重要的量是未知的，如果我们知道这个量，那么我们就可以找到一种有效的方法解决问题，这时对这个量进行部分穷举，很可能会产生高效的算法。

用穷举思想解题，需要分为几步处理

1. 准确理解题意

   这一步对所有的思想和题目都是必要的。

2. 确定使用穷举思想

   使用穷举思想，需要一定的条件。首先，穷举是一种非常普遍的方法，如果

题目有其他专用的办法，那么应该避免使用穷举来解题。其次，题目中需要穷举的状态数不能太大，否则算法在时间上难以承受。

3. 明确穷举对象

穷举对象的选择是非常重要的。它直接影响着算法的时间复杂度。选择适当的穷举对象可以使穷举获得很高的效率（见例 1）。

4. 解题

1.1 例一 聪明的打字员[1]

题目大意：

使用一个只有加减 1（Up/Down），左右移动光标（Left/Right），与 1，6 交换（Swap0/Swap1）六个键的键盘，用最少的步数把一个 6 位数转化成另外一个。

例如，初始状态是 123456，要求的状态是 633451，那么最简单的转化方法

123456 (swap1) **è** 623451(right) **è** 623451(up) **è** 633451

是：

初步分析

由于 Swap0/Swap1 两个键的出现，此题似乎没有很好的方法，于是进行搜索。但是，该题的状态总数是很多的，达到了 $6*10^6$，如果直接搜索，时空上难以承受。虽然这道题有比较有效的估值函数和剪枝措施，但是一般的搜索方法还是不能胜任。于是使用搜索就只好用效率较高的 Hash+A*或双向广度优先，这样虽然可以在规定时间内出解，但是效率仍然不高，而且编程复杂度很高。

思考：使用穷举思想

考虑本题的难点：Swap0/Swap1。若没有这两个键，那么该题恐怕连分区联赛的复赛都不会考。既然这样，我们为什么不穷举这两个键的使用情况呢？

表面看来，我们不能这样做。因为我们不能确定这两个键与加减 1 两键的使用次序，也不知道需要进行多少次按键操作，这样就使状态数不可预测，达到不可忍受的地步。

事实上，我们可以看到，只要是对同一个数进行加减，Up/Down 放在 Swap0/Swap1 前后实际上是等效的。例如，原来的数是 133456，我们可以通过一个 Up,一个 Right,一个 Swap0 把它变成 323456，也可以先进行 Right, Swap0, 然后再用 Up，也可以把它变成 323456。因此，我们可以把任务分成两个阶段处理。在第一阶段，我们只要通过这两个键把已有的数进行一下排列。那么，因为 6 的全排列只有 720 种，我们完全可以通过穷举（或叫广度优先搜索）计算出达到这些排列的最少步数，然后再进行第二阶段的计算。

在第 2 阶段中，由于已经没有了 Swap0/Swap1，使问题大大简化。

例如我给出的样例中：

123456（第一阶段） **è** 623451（第二阶段） **è** 633451

但是，这样我们又遇到了另外一种障碍：把原始状态转化为一种排列的过程中，可能并不会经过所有的位置，因此也就不能对所有位置进行调整。

[1] 2001 年全国青少年信息学奥林匹克竞赛

例如样例中只有 1，6 位置可以不用额外的 Left/Right 进行调整，而我们要调整位置 2，只能通过增加一个 Right 命令来实现，这种情况给第二阶段的处理带来了很大的麻烦。不过，即使我们把每个位置能否调整的 64 种状态都考虑进去，也只有 46080 种状态，只相当与搜索算法的 0.77%。完全可以接受。

如果我们进一步分析，可以知道：

1. 位置 1 可以调整
2. 2，3，4，5 四个位置，如果靠右的位置可以调整，那么靠左的一定可以调整。这样 2，3，4，5 是否可能调整形成的状态只有{},{2},{2,3},{2,3,4},{2,3,4,5}这五种。
3. 6 这个位置可能可以调整，也可能不可调整，关键是是否使用了 Swap1。

这样总共的可能性只有 2*5=10 种，总共需要穷举的状态数只有 7200，大大优于搜索的方法。

注意：这样我们就完成了第三步：确定穷举对象，这是较完善地解决本题的关键，也是使用完全搜索思想的关键！

两种解法的对比：

算法	时间复杂度	实现难度	评价
搜索	很大，非多项式	大	未充分利用题目条件
穷举思想	O(720*10)常数级	较小	巧妙利用穷举思想，确定研究对象，得到了很好的效果。

1.2 例二 逻辑岛[2]

题目大意：

在逻辑岛上有 3 种居民：永远说真话的神，永远说假话的恶魔和在白天说真话，在夜里说假话的人。

一个社会学家访问了该岛，但他无法通过外表区分这 3 种居民，所以他只能靠分析居民们说的话来判断他们的种类。

岛上只有五个居民 A，B，C，D，E。

他们说的话只有 3 种

1. I am [not] divine/evil/human/lying.
2. X is [not] divine/evil/human/lying.
3. It is day/night.

居民们说话的总数不超过 50。你的任务就是通过居民们说的话来判断他们的种类以及现在是白天还是黑夜。

初步分析

先来看一个示例：

A: B is human.

B: A is evil.

A: B is evil.

如果由人来判断的话，A 说了两句相互矛盾的话，显然这两句话都是假的，那么 B 是神。因为神说的话都是真的，所以 A 肯定是恶魔。这样，这个例子的数据就应该是：

[2] 浙江大学在线评测系统 1252

A is evil.
B is divine.
但是，当我们试图把这种思维方式用程序实现时，我们发现情况非常复杂，很难让计算机进行这样的推理。

应用穷举策略

因为居民只有 5 个，而且每个居民都只有 3 种可能的种类，而时间也只有昼夜之分，所以把这些全部计算在内，状态的总数也只有 $3^5*2=486$ 种，使用穷举完全可以接受。这就是我们选择的穷举对象。
在穷举了每个居民的种类和时间之后，我们只需判断输入的每句话是否符合这个假设。如果居民说的所有话都与这种假设相符合，那么这种假设就是一种可能，综合各种可能性就可以得到最终的结果。

1.3 小结

通过上面两个例子我们可以看出，完全穷举虽然有很大的盲目性，但是我们自身是**灵活**的！
通过对题目仔细深入的分析，灵活地选择穷举的对象，就可以达到化难为易的效果，虽然是使用了完全穷举，也能使程序的效率大大提高，还可以减小编程实现的复杂程度，达到很好的效果。

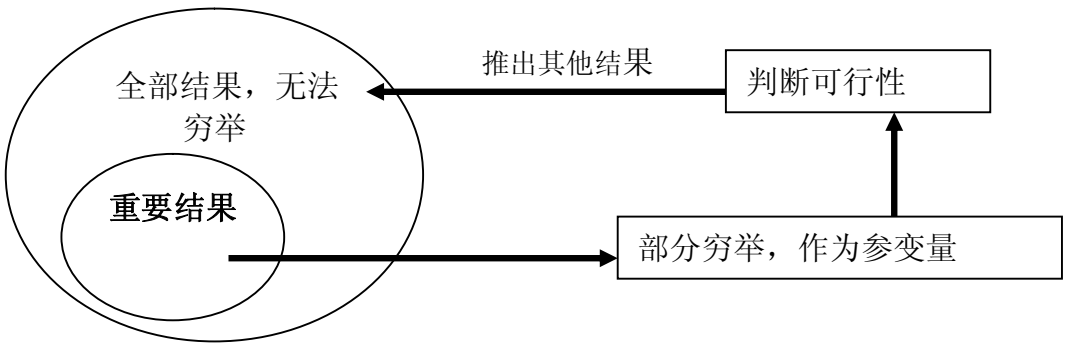
2.部分穷举思想——参变量法

基本思想

图论、贪心、动态规划等方法虽然已经发展得日益完善，但是它们自身仍然有很多限制。一个题目往往只因为一个条件作梗，就使我们寻求高效算法的企图落空。但是，我们仍然有一些办法来打破这些限制。**部分穷举**的思想就是其中重要的一个。

部分穷举的思想，或具体地称为参变量法，是事先通过穷举来确定问题需要的一个重要结果，然后通过贪心，动态规划或图论中的一些方法来判断这个结果是否可行，并推出其他所需结论的方法。

下面就是部分穷举思想的图示



部分穷举思想由于只穷举了重要的结果，其他的结果由高效的算法得出，所以时间复杂度一般都是多项式级的，可以满足题目的要求。但是，有时我们也需要利用题目的一些特殊性质来进行优化。

一般题目最重要的性质是单调性。所谓单调性是指如果参变量为 x_1 时可行，那么参变量大于 x_1 时也可行，而且题目要求的是参变量的最小值。这时，我们有两种选择

1. 仍然用线形的穷举方法，但是当得到一个可行的解时，就停止。
2. 改用二分形式的穷举方法。

这两种方法中，二分法可以保证较低的时间复杂度，但是实现过程稍复杂一些。对于一些范围比较小的题目，或者是解比较靠近边缘的题目，我们可以使用第一种方法，这样可以节约实现的时间，而且同时也避免了一些低级错误。注意二分穷举事实上并没有“穷尽”每一个元素，但是其中的思想和部分穷举是一致的。

适用范围

一般来说，最大最小（或最小最大）问题适合使用部分穷举的思想。所谓最大最小的问题，就是题目中定义一种权值，要求产生一种划分（或其他类似的结构），使划分的每一部分权值的最大值达到最小。这时候，我们往往采用部分穷举的思想使问题简化为：**已知一个权值，判断是否有满足要求的划分？**这个问题是我们使用部分穷举思想解决原问题的关键，它不但要有正确的，可构造出解答的解法，而且要高效，因为我们可能要多次调用这种解法。

2.1 例三 草莓[3]

题目大意

森林中的精灵们把这片草莓田分成 k 块种到 k 个空地中去，以免被粗鲁的棕熊搞乱。她们希望每块空地上恰好放上一块用触须连接在一起的草莓田。不过，如果一块空地里的草莓太少，它们就会感到孤单，所以精灵们希望无论哪块空地含有草莓的总重量都不要太小。

定义：sum_i 表示第 i 块草莓田中所有草莓重量的和（$1 \le i \le k$）。

$$x = \min\{sum_i \mid 1 \le i \le k\}$$

你的任务就是要把一片草莓田分割成 k 块，且分割方案需要满足如下的条件：

- 每一块中的草莓必然是通过触须直接或者间接和其他草莓相连接的；
- 这种分割方案所对应的 x 尽可能的大。

最后输出你的分割方案和结果。

附加的限制

这是一道提交答案的问题。题目给出的数据有许多都是树。这类情况是可以用参变量法来解决的。在这里我们假设我们处理的都是树状结构，而且我们需要得到最优解。

初步分析

提到树上求最值的问题，人们容易联想到树的动态规划。这道题似乎也可以这样解决。首先我们把草莓作为树的节点。我们可以把某子树分成 K 份，所能达到的最小总重量作为状态，然后列出状态转移方程（设 $D(n,k)$表示把第 n 个点和它的子孙节点构成的树分成 k 份，所能达到的最优解,而第 n 个节点有 t 个儿子）：

[3] 2003 年全国青少年信息学奥林匹克竞赛

$D(n,k)$=MAX: 对任意的一个 $k=a_1+a_2+\cdots+a_t$ {min($D(child_1,a_1)$, $D(child_2,a_2),\cdots, D(child_t,a_t)$)+把节点 n 分配到第 i 个儿子的调整值}

这个状态转移方程似乎是正确的，但是注意到其中有一项：把节点 n 分配到第 i 个儿子的调整值。要求得这一项，我们必需知道把第 n 个点和它的子孙节点构成的树分成 k 份的最小值和第二小值。而求第二小值的状态转移方程又包含第三小值，因此，我们必需同时存储所得到的所有值。这在事实上是做不到的。而且，即使我们可以使用这个方程，时间上也是不允许的。

选择参变量

如果已知一个分割方案所对应的 x，我们如何去寻找一个解答，或者证明这种分割是不存在的？（问题★）

这是我们使用参变量法的关键。如果这个问题得以解决，我们就可以应用部分穷举的思想，穷举每一个 x，然后判断是否有解。如果有解，便可求出划分方案。

问题★的解答：动态规划或贪心

问题★事实上并不难解决。我们完全可以用动态规划的方法：设 $T(n)$表示 n 和 n 的子孙分割成重量大于 x 的最大块数，而 $Left(n)$表示进行这样的分割以后，所剩余的最大草莓重量。如果剩余的重量加上 n 的重量够再分一块，那么我们就把它分出来，这时 $Left(n)$就是 0。否则就令 $Left(n)$等于所有这些重量。

这样，我们得到状态转移方程如下：

如果 $Left(child_1)+Left(child_2)+\cdots+Left(child_t)+$第 n 株的重量$\geqslant x$，那么
$T(n)=T(child_1)+T(child_2)+\cdots+T(child_t)+1, Left(n)=0$
否则：$T(n)= T(child_1)+T(child_2)+\cdots+T(child_t)$
$Left(n)= Left(child_1)+Left(child_2)+\cdots+Left(child_t)+$第 n 株的重量

贪心的思想和这个动态规划是类似的：即从下向上处理，遇到可以分的便分出来。由于动态规划的方法的正确性比较容易证明，而贪心的结果事实上和动态规划是相同的，这样我们就证明了这个贪心算法的正确性。

总体的解法

由于 x 值的范围很大，而这个问题恰好满足单调性，所以我们应该采用二分的方法进行穷举，然后选择贪心或动态规划来进行处理。

这样，我们利用参变量法解决了一个最小最大的问题。但是，如果把题目的要求改成最大最小，贪心或动态规划的方法就不再成立，于是很难找到有效的算法。

2.2 最大最小匹配

部分穷举思想也被一些比较经典的算法所采用。最大最小匹配（或最小最大匹配）就是其中很重要的一种算法。

问题的提出

我们都知道如何求二分图的最大匹配，最大权匹配，但是有时我们需要这样一种匹配：

1. 这个匹配是在一个带权的二分图上进行的

2. 这个匹配是一个完备匹配
3. 这个匹配是满足 2 的条件中，匹配边的最大权最小的匹配。

即定义 x=max{匹配边的权},求使 x 最小的完备匹配。

初步分析

这个问题不能直接转化为一个已知解法的匹配问题。所以一般的想法是尝试用网络流的模型来解决。但是，已知的网络流算法所能包含的信息都是可相加的权，而本题中的权需要取最小值，不是简单的相加关系。所以我们很难用网络流的方法直接解决本题。

采用部分穷举的思想

如果已知一个 x，是否可以很快产生一个满足题目要求的匹配？

如果我们解决了这个问题，那么我们就可以解决最大最小匹配问题。而这个问题事实上是比较简单的。既然权的最大值是 x，那么所有权小于等于 x 的边都可用，权大于 x 的边都不可用。于是我们得到了一个不带权的二分图，如果我们可以在这个图中找到完备匹配，那么就产生了一个满足要求的匹配；否则就说明这样的匹配不存在。

穷举方式选择

表面看来边权的范围可能很大，只能采用二分穷举。但是，事实上“有效”的边权最多只有 n^2 个，所以使用线形的穷举也并非不可。当然，从算法时间效率的稳定性方面考虑，最好还是选用二分穷举。

一些扩展

1.带权图的最大最小匹配

2.权最大的带权二分图最大最小匹配

【总结】

本文应用完全穷举和部分穷举两种穷举方法，处理了一些问题。我们可以看出，**穷举≠低效！**对于完全穷举，我们可以通过适当选取穷举对象来达到我们所希望的结果；而部分穷举思想则是解决最大最小问题的有力武器，对其他问题也有一些影响。部分穷举思想还扩展了图论贪心等算法的应用范围，而且它的效率一般也是可以满足要求的。

完全穷举和部分穷举的比较

穷举方法	完全穷举	部分穷举
时间效率	与穷举对象选择关系很大	一般比较好
空间效率	好	取决于后续算法
实现复杂度	较低	取决于后续算法
难点	穷举对象的选择	选择穷举的变量
适用范围	没有其他好方法的题目 J毕竟穷举是一种通用的方法	最大最小问题，当然其他问题只要能找到参变量就可以用。

总之，完全穷举和部分穷举是我们解题时的有效思想，我们应对其给予充分的重视，根据题目的具体情况选择应用，不应简单地认为穷举方法效率低而弃之不用。

【参考文献】

俞玮的 NOI2001 解题报告
《图论的算法与程序设计》清华大学出版社,吴文虎，王建德著。

【附录】:

题目 1，聪明的打字员：

聪明的打字员

clever.pas/c/cpp

阿兰是某机密部门的打字员，她现在接到一个任务：需要在一天之内输入几百个长度固定为 6 的密码。当然，她希望输入的过程中敲击键盘的总次数越少越好。

不幸的是，出于保密的需要，该部门用于输入密码的键盘是特殊设计的，键盘上没有数字键，而只有以下六个键：Swap0, Swap1, Up, Down, Left, Right，为了说明这 6 个键的作用，我们先定义录入区的 6 个位置的编号，从左至右依次为 1，2，3，4，5，6。下面列出每个键的作用：

Swap0：按 Swap0，光标位置不变，将光标所在位置的数字与录入区的 1 号位置的数字（左起第一个数字）交换。如果光标已经处在录入区的 1 号位置，则按 Swap0 键之后，录入区的数字不变；

Swap1：按 Swap1，光标位置不变，将光标所在位置的数字与录入区的 6 号位置的数字（左起第六个数字）交换。如果光标已经处在录入区的 6 号位置，则按 Swap1 键之后，录入区的数字不变；

Up：按 Up，光标位置不变，将光标所在位置的数字加 1（除非该数字是 9）。例如，如果光标所在位置的数字为 2，按 Up 之后，该处的数字变为 3；如果该处数字为 9，则按 Up 之后，数字不变，光标位置也不变；

Down：按 Down，光标位置不变，将光标所在位置的数字减 1（除非该数字是 0），如果该处数字为 0，则按 Down 之后，数字不变，光标位置也不变；

Left：按 Left，光标左移一个位置，如果光标已经在录入区的 1 号位置（左起第一个位置）上，则光标不动；

Right：按 Right，光标右移一个位置，如果光标已经在录入区的 6 号位置（左起第六个位置）上，则光标不动。

当然，为了使这样的键盘发挥作用，每次录入密码之前，录入区总会随机出现一个长度为 6 的初始密码，而且光标固定出现在 1 号位置上。当巧妙地使用上述六个特殊键之后，可以得到目标密码，这时光标允许停在任何一个位置。

现在，阿兰需要你的帮助，编写一个程序，求出录入一个密码需要的最少的击键次数。

输入文件（clever.in）

文件仅一行，含有两个长度为 6 的数，前者为初始密码，后者为目标密码，两个密码之间用一个空格隔开。

输出文件（clever.out）
文件仅一行，含有一个正整数，为最少需要的击键次数。

输入样例
123456　654321

输出样例
11

样例说明：
初始密码是 123456，光标停在数字 1 上。对应上述最少击键次数的击键序列为：

击键序列：	击键后的录入区 （下划线表示光标所在位置）
	123456
Swap1	623451
Right	623451
Swap0	263451
Down	253451
Right	253451
Up	254451
Right	254451
Down	254351
Right	254351
Up	254361
Swap0	654321

最少的击键次数为 11。

程序 1，聪明的打字员：
{这个程序只穷举了一部分情况，但是可以通过所有的测试数据。完整的穷举需要先用广度优先搜索}
```
program clever;
var
d,f,t:array[1..6] of integer;
swap:array[1..6] of -2..2;{穷举当前位置进行 swap0/swap1 情况}
min:integer;
procedure init;
```

```
    var
   i:integer;
    begin
   for i:=1 to 6 do read(f[i]);
   for i:=1 to 6 do read(d[i]);
   for i:=1 to 6 do swap[i]:=-2;
   min:=9999;
   end;
procedure solve;
    var
   k:integer;
   nswap:integer;
   nmin:integer;
   ys6:boolean;
   procedure pswap;{进行交换操作}
    procedure sw(var sw1,sw2:integer);
        var
         swt:integer;
         begin
         swt:=sw1;
         sw1:=sw2;
         sw2:=swt;
         end;
      var
      pi:integer;
      begin
      nswap:=0;
      t:=f;ys6:=false;
      for pi:=1 to 6 do
            if swap[pi]=-1 then begin
            sw(t[pi],t[1]);inc(nswap);
           end
        else if swap[pi]=1 then begin
            sw(t[pi],t[6]);inc(nswap);
           ys6:=true;
           end;
      end;
   function mmc:integer;
    var
      mr,mi,mj:integer;
      begin
      mj:=6;
      while (mj>0)and(t[mj]=d[mj])and(swap[mj]=0)do mj:=mj-1;
      if (swap[6]=0)and(mj>5)and(ys6=true) then mj:=5;
```

```
    if mj=0 then begin writeln(0);halt;end;
    mr:=0;
    for mi:=1 to 6 do mr:=mr+abs(d[mi]-t[mi]);
    mr:=mr+nswap+mj-1;
    mmc:=mr;
    end;
  begin
  k:=1;
  while k>0 do begin
   swap[k]:=swap[k]+1;
    if swap[k]<2 then begin
      if k=6 then begin
         pswap;
         nmin:=mmc;
         if nmin<min then min:=nmin;
         end
       else begin
          k:=k+1;
         swap[k]:=-2;
         end;
       end
    else
      k:=k-1;
    end;
  end;
begin
init;
solve;
writeln(min);
end.
```

题目 2，逻辑岛：

Island of Logic

Time limit: 1 Seconds　Memory limit: 32768K

Total Submit: 84　Accepted Submit: 35

The Island of Logic has three kinds of inhabitants: divine beings that always tell the truth, evil beings that always lie, and human beings that are truthful during the day and lie at night. Every inhabitant recognizes the type of every other inhabitant.

A social scientist wants to visit the island. Because he is not able to distinguish the three kinds of beings only from their looks, he asks you to provide a communication analyzer that deduces facts from conversations among inhabitants. The interesting facts are whether it is day or night and what kind of beings the speakers are.

Input

The input contains several descriptions of conversations. Each description starts with an integer n, the number of statements in the conversation. The following n lines each contain one statement by an inhabitant. Every statement line begins with the speaker's name, one of the capital letters A, B, C, D, E, followed by a colon `:'. Next is one of the following kinds of statements:

I am [not] (divine | human | evil | lying).

X is [not] (divine | human | evil | lying).

It is (day | night).

Square brackets [] mean that the word in the brackets may or may not appear, round brackets () mean that exactly one of the alternatives separated by | must appear. X stands for some name from A, B, C, D, E. There will be no two consecutive spaces in any statement line, and at most 50 statements in a conversation.

The input is terminated by a test case starting with n = 0.

Output

For each conversation, first output the number of the conversation in the format shown in the sample output. Then print ``This is impossible.'', if the conversation cannot happen according to the rules or ``No facts are deducible.'', if no facts can be deduced. Otherwise print all the facts that can be deduced. Deduced facts should be printed using the following formats:

X is (divine | human | evil).

It is (day | night).

X is to be replaced by a capital letter speaker name. Facts about inhabitants must be given first (in alphabetical order), then it may be stated whether it is day or night.

The output for each conversation must be followed by a single blank line.

Sample Input

1
A: I am divine.
1
A: I am lying.
1
A: I am evil.
3
A: B is human.
B: A is evil.
A: B is evil.
0

Sample Output

Conversation #1
No facts are deducible.

Conversation #2
This is impossible.

Conversation #3
A is human.
It is night.

Conversation #4
A is evil.
B is divine.

Reasoning made easy

To make things clearer, we will show the reasoning behind the third input example, where A says ``I am evil.''. What can be deduced from this? Obviously A cannot be divine, since she would be lying, similarly A cannot be evil, since she would tell the truth. Therefore, A must be human, moreover, since she is lying, it must be night. So the correct output is as shown.

In the fourth input example, it is obvious that A is lying since her two statements are contradictory. So, B can be neither human nor evil, and consequently must be divine. B always tells the truth, thus A must be evil. Voil‘a!

程序 2，逻辑岛：

```
{comment:ZJU1252 island of logic}
type
```

```
sentence=record
          typ:integer;{0-sb. is sth.;1-sb. is lying;2-It's day/night}
          speaker,sb:integer;
          sth:integer;{0-evil;1-human;2-divine;}
          isnot:boolean;{true-yes,day;false-not,night}
          end;
var
n:integer;
tp:array[1..5] of integer;
day:boolean;
number:integer;
st:array[1..50] of sentence;
lying:array[1..5] of boolean;
fact:array[1..5] of integer;{-1-unknown;0,1,2-see above;3-uncertain}
rtime:integer;{-1-unknown;0-day;1-night;2-uncertain}
procedure outp;
  var
  nfact,i:integer;
  begin
  if fact[1]=-1 then writeln('This is impossible.')
  else begin
      nfact:=0;
      for i:=1 to 5 do if fact[i]<>3 then begin
         write(chr(i+64),' is ');
         case fact[i] of
             0:writeln('evil.');
             1:writeln('human.');
             2:writeln('divine.');
             end;
         inc(nfact);
         end;
      if rtime<2 then begin
        if rtime=0 then writeln('It is day.') else writeln('It is
night.');
        inc(nfact);
        end;
      if nfact=0 then writeln('No facts are deducible.');
      end;
  end;
procedure combineresult;
  var
  i:integer;
  begin
  for i:=1 to 5 do
```

```
      if fact[i]=-1 then fact[i]:=tp[i]
      else if fact[i]<>tp[i] then fact[i]:=3;
  if rtime=-1 then begin
    if day then rtime:=0 else rtime:=1;
    end
  else if (rtime=0)xor day then rtime:=2;
  end;
function checkt(ts:sentence):boolean;
  begin
  if ts.typ=0 then begin
    if  (tp[ts.sb]=ts.sth)xor  ts.isnot  then  checkt:=true  xor
lying[ts.speaker]
    else checkt:=false xor lying[ts.speaker];
    end
  else if ts.typ=1 then begin
      if  lying[ts.sb]  xor  ts.isnot  then  checkt:=true  xor
lying[ts.speaker]
      else checkt:=false xor lying[ts.speaker];
      end
  else begin
      checkt:=not (day xor ts.isnot xor lying[ts.speaker]);
      end;
  end;
function check:boolean;
  var
  i:integer;
  begin
  for i:=1 to n do if not checkt(st[i]) then begin
     check:=false;exit;end;
  check:=true;
  end;
procedure number2st;
  var
  i:integer;
  tnum:integer;
  begin
  tnum:=number;
  if tnum>=243 then begin
    day:=true;tnum:=tnum-243;
    end
  else day:=false;
  for i:=1 to 5 do begin
     tp[i]:=tnum mod 3;
     tnum:=tnum div 3;
```

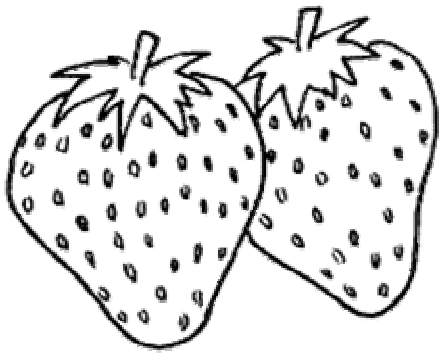
```
      end;
   for i:=1 to 5 do
      if tp[i]=0 then lying[i]:=true
      else if tp[i]=2 then lying[i]:=false
      else lying[i]:=not day;
   end;
procedure init;
   var
   i:integer;
   s:string;
   begin
   readln(n);
   for i:=1 to n do begin
      readln(s);
      st[i].speaker:=ord(s[1])-64;
      delete(s,1,3);delete(s,length(s),1);
      if pos('not',s)<>0 then st[i].isnot:=true else st[i].isnot:=false;
      st[i].typ:=0;
      if copy(s,1,4)='I am' then st[i].sb:=st[i].speaker
      else if copy(s,1,5)='It is' then st[i].typ:=3
      else st[i].sb:=ord(s[1])-64;
      if copy(s,length(s)-3,4)='evil' then st[i].sth:=0
      else if copy(s,length(s)-4,5)='human' then st[i].sth:=1
      else if copy(s,length(s)-5,6)='divine' then st[i].sth:=2
      else if copy(s,length(s)-4,5)='lying' then st[i].typ:=1
      else if copy(s,length(s)-4,5)='night' then st[i].isnot:=false
      else st[i].isnot:=true;
      end;
   end;
begin
init;
fillchar(fact,sizeof(fact),255);
rtime:=-1;
for number:=0 to 485 do begin
    number2st;
    if check then combineresult;
    end;
outp;
end.
```

题目 3，草莓

草 莓

【题目背景】

尽管不少人都吃过鲜美的草莓，却很少有人真正观察过野草莓的生长。它们从自己的枝上伸出一根根长长的触须，遇到合适的地方就会扎根发芽，长出一株新的草莓。所以，当你在森林中遇到一株草莓的时候，通常就意味着你会在它的周围找到一片草莓田。但这些草莓并非能够无忧无虑地生长，树林中穿梭的鸟儿和偶尔路过的鹿群都喜欢吃这种美味的浆果。不过，草莓最大的威胁却是来自那些贪吃的棕熊。他们不但可以吃掉整整一片草莓，而且还会粗鲁地把一片草莓田搞得乱七八糟。于是每当一块草莓田越长越大之后，森林中的精灵们就会把这片草莓田分成 k 块种到 k 个空地中去，以免被粗鲁的棕熊搞乱。她们希望每块空地上恰好放上一块用触须连接在一起的草莓田。不过，如果一块空地里的草莓太少，它们就会感到孤单，所以精灵们希望无论哪块空地含有草莓的总重量都不要太小。可是天真的精灵并不知道怎样来做这件事情，你可以帮助她们吗？



【任务描述】

定义：sum_i 表示第 i 块草莓田中所有草莓重量的和（$1 \le i \le k$）。

$$x = \min\{sum_i \mid 1 \le i \le k\}$$

你的任务就是要把一片草莓田分割成 k 块，且分割方案需要满足如下的条件：

- 每一块中的草莓必然是通过触须直接或者间接和其他草莓相连接的；
- 这种分割方案所对应的 x 尽可能的大。

最后输出你的分割方案和结果。

【输入说明】

第一行为三个整数 n、m 及 k，n 表示草莓的株数，m 表示触须的数目，k 为空地的数目。

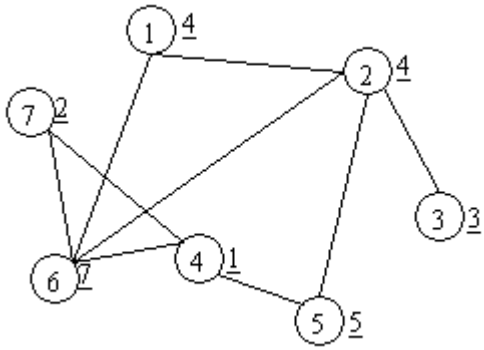
接下来的 n 行每行两个整数 i 及 b_i，表示第 i 株草莓的重量是 b_i 克。顺序下来的 m 行每行两个整数 p 和 q，表示第 p 株草莓和第 q 株草莓之间有一根触须相连接。

另外，在所有这些数据的最后还有单独的一行包括一个整数 d 用作评分系数，有关 d 的说明，可以参看下面的评分方法。

【输出说明】

你一共要输出 *k+1* 行。第一行为一个整数，表示你的分割方案中的 *x*。接下来的 *k* 行，每行表示一块草莓田。每行的第一个整数为 n_i，表示第 *i* 块田中的草莓株数。第二个到第 n_i+1 个整数为这些草莓的编号。请注意，这些草莓必然是通过触须相连接的。

【输入样例】




```
7 9 3
1 4
2 4
3 3
4 1
5 5
6 7
7 2
1 2
1 6
2 3
2 5
2 6
4 5
4 6
4 7
6 7
2000000000
```

【输出样例】

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

【评分方法】

本题是一道提交答案式题目，你需要针对给定的 10 个输入文件 2/berry1.in~2/berry10.in 提交你的输出文件 berry1.out~berry10.out（放在你的选手目录下）。我们将根据你提交的输出文件评分。对于某一确定的测试点来说，如果你的输出文件中第一行的 *x* 和下面的分割方案不符合，或者是输出文件本身就有错误，那么你将得不到该测试点的分数。这里输出文件的错误可以使用我们提供的 2/berry_check 检查工具进行检查。只有当这个程序输出 Yes 的时候，你的输出才可以确认是可接受的。

对于可接受的输出，评分公式如下：

$$score = \left[10 \times e^{-\frac{2}{5} \times \left(\frac{d \times (best - x)}{best}\right)^2} \right]$$

这里 *d* 为评分系数（输入数据中最后一行的整数），*best* 为我们的最优结果。

注意：可接受的输出不一定能够得分。

【你如何测试自己的输出】

我们提供 2/berry_check 这个工具来测试你的输出文件是否是可接受的。使用这个工具的方法是：

`2/berry_check <输入文件名> <输出文件名>`

在你调用这个程序后，2/berry_check 将根据你给出的输入和输出文件给出测试的结果，其中包括：

- 非法退出：未知错误；
- not connect：你程序输出的行中含有不连通的分量；
- duplicate：同一点输出了两次；
- extra：输出文件中包括多余数据；
- lack：输出的总点数不对；
- answer not match：输出中第一行的 x 和实际结果不符；
- Yes：输出可接受，将进行下一步的评分。

程序 3，草莓树形数据：

```
{comment:NOI2003 berry for tree-like datas}
const
maxn=10000;
maxk=600;
testcase='1';{测试点号}
type
Penode=^enode;
enode=record v:integer;next:Penode;end;
tnode=record value:longint;elist:Penode;end;
var
tr:array[1..maxn] of tnode;
dp:array[1..maxn] of Penode;
vis,vis2:array[1..maxn] of boolean;
n,m,k,nk:integer;
```

```
limit,sum,ans:longint;
maxdep:integer;
outlist:array[1..maxk] of Penode;
number:array[1..maxk] of integer;
procedure destroy(td:Penode);{释放一个链表}
  begin
  if td<>nil then begin
    destroy(td^.next);
    dispose(td);
    end;
  end;
procedure outp;{输出}
  var
  i:integer;
  t:Penode;
  begin
  writeln(ans);
  for i:=1 to k do begin
     write(number[i]);
     t:=outlist[i];
     while t<>nil do begin
          write(' ',t^.v);
          t:=t^.next;
          end;
     writeln;
     end;
  close(output);
  end;
procedure clean;{初始化}
  var
  i:integer;
  begin
  for i:=1 to k do destroy(outlist[i]);
  fillchar(outlist,sizeof(outlist),0);
  fillchar(number,sizeof(number),0);
  nk:=0;
  end;
procedure fillout(id:integer);{划分}
  var
  t:Penode;
  begin
  inc(number[nk]);
  new(t);
  t^.v:=id;
```

```
  t^.next:=outlist[nk];
  outlist[nk]:=t;
  vis2[id]:=true;
  t:=tr[id].elist;
  while t<>nil do begin
       if vis[t^.v] and not vis2[t^.v] then fillout(t^.v);
       t:=t^.next;
       end;
  end;
function scan:boolean;{贪心法处理}
  var
  used:array[1..maxn] of boolean;
  tsum:array[1..maxn] of longint;
  i:integer;
  tn1,tn2:Penode;
  begin
  fillchar(vis,sizeof(vis),false);
  fillchar(vis2,sizeof(vis2),false);
  for i:=maxdep downto 1 do begin
     tn1:=dp[i];
     while tn1<>nil do begin
          if nk=k then begin
            inc(number[k]);
            new(tn2);tn2^.v:=tn1^.v;
            tn2^.next:=outlist[k];
            outlist[k]:=tn2;
            end
          else begin
              used[tn1^.v]:=false;
              tsum[tn1^.v]:=tr[tn1^.v].value;
              vis[tn1^.v]:=true;
              tn2:=tr[tn1^.v].elist;
              while tn2<>nil do begin
                   if vis[tn2^.v] and not used[tn2^.v] then begin
                     tsum[tn1^.v]:=tsum[tn1^.v]+tsum[tn2^.v];
                     end;
                   tn2:=tn2^.next;
                   end;
              if tsum[tn1^.v]>=limit then begin
                inc(nk);
                fillout(tn1^.v);
                used[tn1^.v]:=true;
                end;
              end;{if nk=k}
```

```
          tn1:=tn1^.next;
          end;
      end;
  if nk=k then scan:=true else scan:=false;
  end;
function go(nv:longint):boolean;
  begin
  clean;
  limit:=nv;
  go:=scan;
  end;
function work(lo,hi:longint):longint;{二分法穷举}
  var
  mid:longint;
  tv:boolean;
  begin
  if hi<lo then work:=-1
  else if hi=lo then begin
       tv:=go(lo);
       if tv then work:=lo else work:=-1;
       end
  else begin
       mid:=(lo+hi) div 2;
       tv:=go(mid);
       if tv then begin
          mid:=work(mid+1,hi);
          if mid=-1 then work:=(lo+hi) div 2 else work:=mid;
          end
       else work:=work(lo,mid-1);
       end;
  end;
procedure dfs(id,dep:integer);
  var
  t:Penode;
  begin
  if dep>maxdep then maxdep:=dep;
  vis[id]:=true;
  new(t);
  t^.v:=id;
  t^.next:=dp[dep];
  dp[dep]:=t;
  t:=tr[id].elist;
  while t<>nil do begin
       if not vis[t^.v] then dfs(t^.v,dep+1);
```

```
        t:=t^.next;
        end;
   end;
procedure init;{总初始化}
   var
   i,t1,t2:longint;
   tm:Penode;
   begin
   assign(input,'berry'+testcase+'.in');reset(input);
   read(n,m,k);
   fillchar(tr,sizeof(tr),0);
   sum:=0;
   for i:=1 to n do begin
      read(t1,t2);
      tr[t1].value:=t2;
      sum:=sum+t2;
      end;
   for i:=1 to m do begin
      read(t1,t2);
      new(tm);
      tm^.next:=tr[t1].elist;
      tm^.v:=t2;
      tr[t1].elist:=tm;
      new(tm);
      tm^.next:=tr[t2].elist;
      tm^.v:=t1;
      tr[t2].elist:=tm;
      end;
   fillchar(dp,sizeof(dp),0);
   fillchar(vis,sizeof(vis),false);maxdep:=0;
   dfs(1,1);
   end;
begin
assign(output,'berry'+testcase+'.out');rewrite(output);
init;
fillchar(outlist,sizeof(outlist),0);
ans:=work(1,sum div k);
go(ans);
outp;
end.
```

程序 4,最小最大匹配:

```
{comment:最小最大匹配}
const
max=100;
```

```
var
g:array[1..max,1..max] of integer;
q:array[1..max*max] of integer;
n1,n2,ne:integer;
sy:array[1..max] of integer;
cx,cy:array[1..max] of integer;
limit:integer;
function find(u:integer):boolean;
  var
  v:integer;
  begin
  for v:=1 to n2 do begin
    if (sy[v]=0)and(g[u,v]>=limit) then begin
      sy[v]:=1;
       if (cy[v]=0)or(find(cy[v])) then begin
         cx[u]:=v;cy[v]:=u;
         find:=true;exit;
         end;
       end;
    end;
  find:=false;
  end;
function match:boolean;
  var
  i:integer;
  begin
  fillchar(cx,sizeof(cx),0);
  fillchar(cy,sizeof(cy),0);
  for i:=1 to n1 do begin
   fillchar(sy,sizeof(sy),0);
    if not find(i) then begin
       match:=false;exit;
       end;
    end;
  match:=true;
  end;
function work(lo,hi:integer):integer;
  var
  t:integer;
  begin
  if lo>hi then begin
    work:=-1;exit;end;
  if lo=hi then begin
    limit:=q[lo];
```

```
      if match then work:=q[lo] else work:=-1;
      end
   else begin
       limit:=q[(lo+hi) div 2];
       if not match then begin
          t:=work(lo,(lo+hi)div 2-1);
          if t=-1 then work:=q[(lo+hi) div 2] else work:=t;
          end
       else work:=work((lo+hi)div 2+1,hi);
       end;
   end;
procedure insort(lo,hi:integer);
   var
   i,j,t:integer;
   begin
   for i:=lo to hi-1 do
      for j:=i+1 to hi do
         if q[i]>q[j] then begin
            t:=q[i];q[i]:=q[j];q[j]:=t;
            end;
   end;
procedure qsort(lo,hi:integer);
    var
   i,j,k,t:integer;
   begin
   if lo<hi then begin
      if hi-lo<10 then insort(lo,hi);
    k:=q[lo+random(hi-lo+1)];
      i:=lo;j:=hi;
      while i<=j do begin
        while q[i]<k do inc(i);
         while q[j]>=k do dec(j);
         if i<j then begin
            t:=q[i];q[i]:=q[j];q[j]:=t;dec(j);inc(i);
           end;
         end;
      if i=lo then begin
           q[(lo+hi)div 2]:=q[i];
           q[i]:=k;
           qsort(i+1,hi);end
           else begin qsort(lo,j);qsort(i,hi);end;
      end;
   end;
procedure init;
```

```
  var
  i,t1,t2,t3:integer;
  begin
  fillchar(q,sizeof(q),0);
  assign(input,'matching.in');reset(input);
  read(n1,n2,ne);
  for i:=1 to ne do begin
     read(t1,t2,t3);
     g[t1,t2]:=t3;
     q[i]:=t3;
     end;
  close(input);
  end;
begin
init;
qsort(1,ne);
assign(output,'matching.out');rewrite(output);
writeln(work(1,ne));
close(output);
end.
```