《人 工 智 能 导 论》

第一次大作业

项目报告

班级：自 45

姓名：林子坤

学号：2014011541

目录

《人工智能导论》第一次大作业

项目报告

林子坤

（自动化系 自 45 班 2014011541）

一、项目简介

本次项目选择了给出的四道题目中的第二道，并达到以下目标：1、求解题目要求中给出的两道例题；2、求解所有这类型的题目；3、随机生成这类型的题目，并保证随机生成的题目满足：符合中文语法、符合语言逻辑、有且只有一个解；4、生成难度更大的题目。

希望使用的评分模板是模板 B。

二、程序属性

本程序使用 C#语言编写，使用 Visual Studio 2013 Community 编译器。支持的运行环境是 Windows 7 至 Window 10 各版本。

程序界面如下图所示。

三、程序功能与使用方法

3.1 选择解题模式

3.1.1 计算机解题模式

在计算机解题模式中，计算机能够求解题目并以对话框形式显示答案。这道题目可以是默认的题目，也可以是计算机随机出的题目。

使用时，通过点击“模式”栏中“计算机解题”单选框的选择项，选择计算机解题模式。

3.1.2 人工解题模式

在人工解题模式中，玩家需要在限定的时间内完成推理问题的求解并提交答案，计算机将检验答案的正确性并以对话框形式提示玩家。

使用时，通过点击“模式”栏中“人工解题”单选框的选择项，选择人工解题模式。

3.2 选择问题类型

3.2.1 “牧师/骗子/赌棍”类型

在“牧师/骗子/赌棍”类型的题目中，题目中的三名角色将对自己的身份进行一句话阐述，并基于以下规则：牧师从不说谎，骗子总说谎，赌棍有时说真话有时说谎话。

使用时，通过点击“问题类型”栏中的“牧师/骗子/赌棍”单选框的选择项，选择这个类型的题目。

3.2.2 “正确/错误/部分”类型

在“正确/错误/部分”类型的题目中，题目中的三或四位角色将对一件未知的物品进行两句猜测，并获得以下结果：有一个人判断完全正确，一个人完全说错，剩余的人恰好说对一半。

使用时，通过点击“问题类型”栏中的“正确/错误/部分”单选框的选择项，选择这个类型的题目。

3.3 选择出题方式

3.3.1 默认题目

本程序为两种类型的题目分别提供一道默认题目，即大作业必做要求中所给出的两道推理题。

使用时，通过点击“出题方式”栏中的“默认题目”单选框的选择项，选择默认题目出题。

3.3.2 随机出题

除了必做要求中的两道推理题外，本程序能够为两种类型的题目随机供题，并确保这道题目有且仅有一个可行解。

使用时，通过点击“出题方式”栏中的“随机出题”单选框的选择项，选择随机题目出题。

3.4 选择“牧师/骗子/赌棍”类型问题的题目形式

注意：“题目形式”栏中的选项仅在选择“牧师/骗子/赌棍”类型问题时有效，在“正确/错误/部分”类型的出题和求解中，“题目形式”栏中的选项将被禁用。

3.4.1 无否定句（简单）

在“无否定句”形式的题目中，三名角色所说的话只能是类似“我是牧师”这样的肯定句，因此在部分题目中，为了保证题目的多样性，可能出现有两名角色说了相同的句子的情况。因此三名角色具有可信度优先级。即如有两人说了相同的话，那么先说的人可信度较高。

使用时，通过点击“题目形式(类型 1)”栏中的“无否定句(易)”单选框的选择项，选择无否定句的形式。

3.4.2 有否定句（困难）

在“有否定句”形式的题目中，三名角色所说的话有可能是“我是牧师”这样的肯定句，也有可能是“我不是骗子”这样的否定句。因此在这部分题目中，不会出现两名角色说了相同的句子的情况。因此三名角色没有可信度优先级，角色说话的先后与可信程度没有关联。

使用时，通过点击“题目形式(类型 1)”栏中的“有否定句(难)”单选框的选择项，选择可能有否定句的形式。

3.5 选择“正确/错误/部分”类型问题的题目背景

注意：“题目背景”栏中的选项仅在选择“正确/错误/部分”类型问题时有效，在“牧师/骗子/赌棍”类型的出题和求解中，“题目背景”栏中的选项将被禁用。

3.5.1 矿石分析（简单）

在“矿石分析”背景中，甲、乙、丙三名地质学院的学生将对一种矿石进行分析，并发表两句猜测，并获得以下结果：有一个人判断完全正确，一个人只说对了一半，一个人则完全说错。

使用时，通过点击“题目背景(类型 2)”栏中的“矿石分析(易)”单选框的选择项，选择矿石分析背景的题目。

3.5.2 挂科惨案（简单）

在“挂科惨案”背景中，某学渣的三位舍友将对这位学渣所挂的科发表两句猜测，并获得以下结果：有一个人判断完全正确，一个人只说对了一半，一个人则完全说错。

使用时，通过点击“题目背景(类型 2)”栏中的“挂科惨案(易)”单选框的选择项，选择挂科惨案背景的题目。

3.5.3 咖啡猜测（简单）

在“咖啡猜测”背景中，三名可爱的初中生将对即将端上来的咖啡种类发表两句猜测，并获得以下结果：有一个人判断完全正确，一个人只说对了一半，一个人则完全说错。

使用时，通过点击“题目背景(类型 2)”栏中的“咖啡猜测(易)”单选框的选择项，选择咖啡猜测背景的题目。

3.5.4 考试成绩（中等）

在“考试成绩”背景中，大学神林若谷的四位高中同学将对他的年级排名发表两句猜测，并获得以下结果：有一个人判断完全正确，两个人只说对了一半，一个人则完全说错。

本题较之前的题目有一定的难度提升，在于人物增加了一人，只说对一半的人也增加了一人，造成计算量增大。

使用时，通过点击“题目背景(类型 2)”栏中的“考试成绩(中)”单选框的选择项，选择考试成绩背景、难度中等的题目。

3.5.5 考试成绩（困难）

在“考试成绩”（困难）背景中，大学神林若谷的四位高中同学将对他的年级排名发表两句猜测，并获得以下结果：有一个人判断完全正确，两个人只说对了一半，一个人则完全说错。

本题和上一题目具有相同的背景，但具有不同的难度，其原因在于这个背景对于每个人肯定句和否定句的数量进行了控制，从而增加了计算难度，且较难通过多次解题的经验快速解出。关于肯定句和否定句的数量对题目难度的影响，以及控制难度的机制，将在后文进行阐述。

使用时，通过点击“题目背景(类型 2)”栏中的“考试成绩(难)”单选框的选择项，选择考试成绩背景、难度困难的题目。

3.6 出题

在以上题目属性选择完毕后，点击“出题”按钮即可生成符合以上属性的题目。若处于“人工解题”模式下，计时器将会启动，从设定时间开始倒计时。

3.7 计算答案/提交答案

在“计算机解题”模式下，提交按钮将自动切换为“计算答案”按钮，点击“计算答案”按钮，程序将解出刚才出的题目，并弹出对话框显示每个人的身份（类型 1）或每个人话语的正确性和实际物品（类型 2）。

在“人工解题”模式下，提交按钮将自动切换为“提交答案”按钮，点击“提交答案”按钮，程序将确认人工解出的答案是否与实际答案相符，并弹出对话框显示实际物品。若答案错误，对话框将显示每个人的身份（类型 1）或每个人话语的正确性（类型 2）。

由于点击提交键后答案将在对话框中被全部显示，因此答案不允许重复提交，点击提交按钮之后，提交按钮将会被禁用以防重复提交，直到下一题出现后才允许使用提交按钮。

四、逻辑建模

4.1 “牧师/骗子/赌棍”类型问题

在“牧师/骗子/赌棍”类型问题中，牧师、骗子和赌棍所能说的话是有限可能的，具体对应表如下表所示。

身份\话语	我是牧师	我是赌棍	我是骗子	我不是牧师	我不是赌棍	我不是骗子
牧师	√				√	√
骗子	√	√				√
赌棍	√	√	√	√	√	√

由表可见，本题的逻辑突破口在于“我是骗子”和“我不是牧师”这两句话上，因为这两句话只有赌棍能够说出。因此在解题时，如果遇到这样的话可以直接判断这人为赌棍；在出题时，应当注意不能同时有超过一人说出这两句话。

接下来的逻辑存在两种可能。第一种可能是，如果第一个人说出“我是骗子”，只需要考虑是否有人说出“我是赌棍”或“我不是赌棍”。若有人说出“我是赌棍”，则他为骗子，剩下一人为牧师；若有人说出“我不是赌棍”，则他为牧师，剩下一人为骗子。如下表所示。

可能性 1

身份\话语	我是牧师	我是赌棍	我是骗子	我不是牧师	我不是赌棍	我不是骗子
牧师	√				√	√
骗子	√	√				√
赌棍	√	√	√	√	√	√

同理，如果第一个人说出“我不是牧师”，也只需要搜索是否有人说出“我是赌棍”或“我不是赌棍”。

当然还存在另一种可能，没有人说出“我是骗子”或“我不是牧师”。此时存在 12 种分支可能，在下表中以 12 种颜色表示。

可能性 2

身份\话语	我是牧师	我是赌棍	我是骗子	我不是牧师	我不是赌棍	我不是骗子
牧师	√√√√				√√√√	√√√√
骗子	√√√√	√√√√√√				√√
赌棍	√√	√√√	√	√	√√√√	√√√

通过以上逻辑建模可以得出，在解题时，可以通过遍历加上适当的剪枝进行搜索；在出题时，可以进行随机参数设置并进行逻辑检查。这将在算法说明部分进行详细阐述。

4.2 “正确/错误/部分”类型问题

在“正确/错误/部分”类型问题中，共有 3 或 4 人分别发表两句判断，且有一个人判断完全正确，一个人完全说错，剩余的人恰好说对一半。以 4 人题目为例，假设实际物品为 thing，错误选项的三件物品分别为 other1thing、other2thing、other3thing，那么以下语句的真假情况如下表所示。

是 thing	√	不是 thing	×
是 other1thing	×	不是 other1thing	√
是 other2thing	×	不是 other2thing	√
是 other3thing	×	不是 other3thing	√

考虑语言逻辑，第一，一位正常的人是不会同时对一个物品进行“是 A，是 B”这样的判断（如：“这块矿石既是铁，又是锡”是不符合语言规范的）；第二，一位正常的人是不会进行“是 A，不是 A”这样的判断（如：“这门课既是应用随机过程，又不是应用随机过程”是不符合语言规范的）；第三，在正常的陈述句中不会出现“是 A，是 A”这样的重复判断（如：“这位小英雄是哪吒，是哪吒”是带有赞叹情感的感叹句，它并不能算提供了两句判断），同理，“不是 A，不是 A”的情况也是不能出现的。

从以上限制可以得出：每个人的两句判断语句不能都是肯定句。同时，考虑到“是 A，不是 B”和“不是 B，而是 A”是逻辑等价的（如：“是福不是祸”和“不是祸，而是福”是逻辑等价的），因此只考虑每个人第一句话均为否定句的情况以简化搜索且不减少出题种类多样性。

考虑到以上限制，这类问题的突破口就在于对 0 句话的人物，他只能说出“不是 thing，而是 other1thing”或“是 other1thing，不是 thing”这类判断，这将有助于我们的出题搜索过程。

值得一提的是，在后期测试人工解题模式的过程中，我和其他的同学发现发现：随着句子中“不是 A，而是 B”形式句子的增加，人工解题的难度将会逐渐上升。因此，在 4 人局面的随机出题时，程序将通过控制“不是 A，而是 B”的数量来达到改变难度的目的，分别对应背景 4 和背景 5。

通过以上逻辑建模和逻辑分析可以得出，在出题时，可以通过逐人搜索的方式进行随机出题。而在解题时，可以通过分析场上正确判断的个数进行搜索。这将在算法说明部分进行详细阐述。

五、算法说明

5.1 “牧师/骗子/赌棍”类型问题

在“牧师/骗子/赌棍”类型问题中，为了平衡不同题目代码量和时间复杂度的代价差异，对于不同难度的问题采取了不同的算法。对于较简单的问题，通过多个判断分支进行剪枝，减少无效枚举，从而减少程序运算量，降低时间复杂度；对于较复杂的问题，付出一定时间复杂度的代价，将判断分支的剪枝过程由人工列举改为程序枚举。下文将对这些算法进行详细介绍。

解决问题、出题采取的算法对应表如下。

	无否定句（简单）	有否定句（困难）
解题	基于逻辑的逐人搜索 （搜索次数少，代码量大、判断分支多）	基于枚举的逐可能判断 （搜索次数多，代码量小、判断分支少）
出题	基于逻辑的逐人随机生成 （搜索次数少，代码量大、判断分支多）	基于枚举的多次随机题试解 （搜索次数多且随机，代码量小、判断分支少）

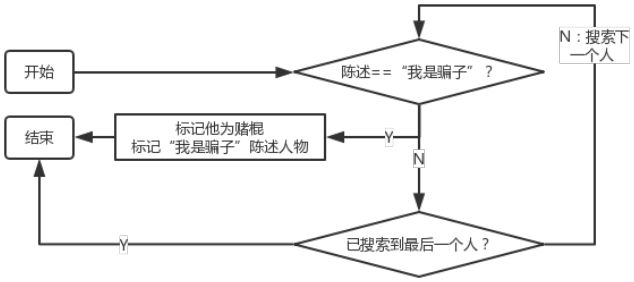
5.1.1 解题

5.1.1.1 无否定句形式的解题：基于逻辑的逐人搜索

基于逻辑的逐人搜索方法根据上一部分所述的逻辑架构，进行四次搜索，填写每人的属性信息。这四次搜索的具体算法如下。

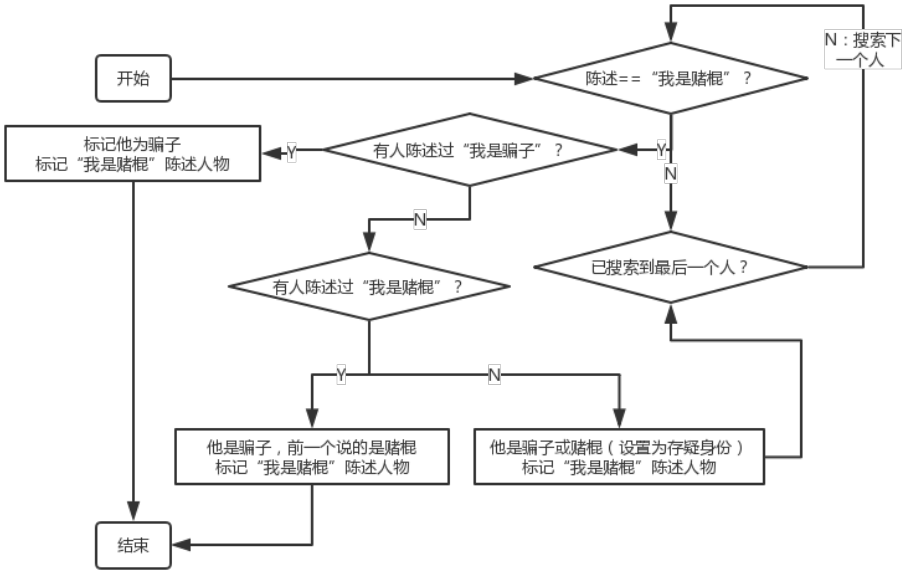
第一次搜索：遍历三人陈述，搜索“我是骗子”陈述。若存在“我是骗子”陈述，标记该人为赌棍，停止搜索（即剪去剩下的搜索枝）。

搜索过程中的判断分支详见以下流程图。



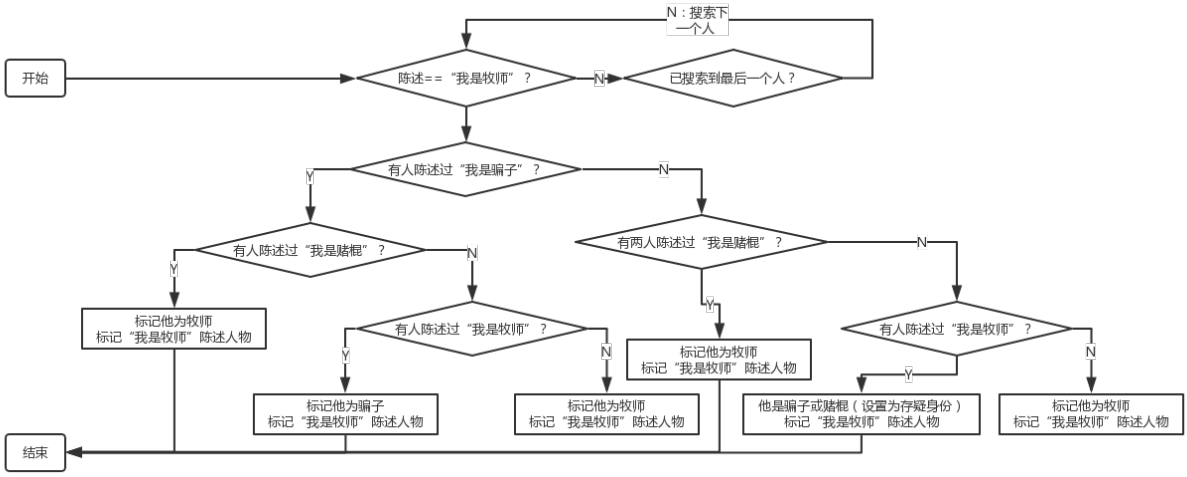
第二次搜索：遍历三人陈述，搜索“我是赌棍”陈述。对于“我是赌棍”陈述者，若已有人陈述“我是骗子”，则“我是赌棍”陈述者为骗子。否则若无人陈述“我是骗子”，继续分情况讨论：若已有人陈述“我是赌棍”，则先陈述者为赌棍，后陈述者为骗子；否则他的身份存疑（可能为赌棍或骗子）。

搜索过程中的判断分支详见以下流程图。



第三次搜索：遍历三人陈述，搜索“我是牧师”陈述。可能有以下情况：1、若有赌棍声明过“我是骗子”且有骗子声明过“我是赌棍”，那么最后这人就是牧师了。2、若有赌棍声明过“我是骗子”且无人声明过“我是赌棍”，则有两人声明“我是牧师”，那么首先声明的人是牧师。3、若有两人声明“我是赌棍”，那么剩最后这人就是牧师了。4、若只有一人声明“我是赌棍”，那么首先声明“我是牧师”的人是牧师。剩下两人可能是赌棍或者骗子，那么首先声明的人是赌棍。

搜索过程中的判断分支详见以下流程图。



第四次搜索：考虑所有存疑的人，先声明的是赌棍。

以上为该算法所需的四次搜索。容易看出，这种算法的时间复杂度是$O(n)$。

由此可见，这种算法的优势是：时间复杂度小。但其也存在着判断分支冗杂、代码量大的不足。该算法适合于各人可能的陈述种类较少时，通过人工提前剪去所有不可能的枝，减少无效枚举，从而减少程序运算量。然而当各人可能的陈述种类增加时，判断分支将会快速增加，人工剪枝变得更加复杂。因此对于有否定句形式的解题，将采用枚举法遍历所有组合，判断各个组合的可能性。

5.1.1.2 有否定句形式的解题：基于枚举的逐可能判断

该种算法采取枚举方法，将牧师、骗子、赌棍的身份分别分配给每个人，并分析每个人的话语是否对应他的身份。由于赌棍可以说出任意正误的陈述，判断时只需判断牧师的陈述是否为真、骗子的陈述是否为假即可。若满足牧师的陈述为真、骗子的陈述为假，则判断完毕，该解为可行解，剪去剩下的枚举分支。

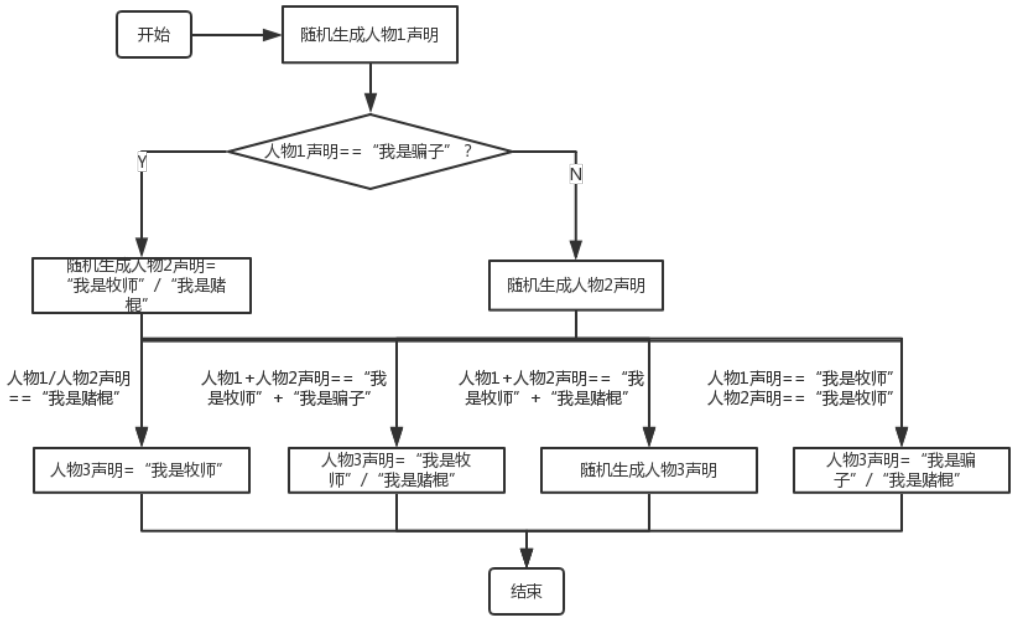
由于以上算法需要对三个人的身份进行分别枚举，因此这种算法的时间复杂度是$O(n^3)$。

由此可见，这种算法的优势是：思路简单、代码量小，将判断分支由人工列举改为程序枚举。但其存在着时间复杂度大的缺点。

5.1.2 出题

5.1.2.1 无否定句形式的出题：基于逻辑的逐人随机生成

为了既保证出题的完全随机性，又保证出的题有且仅有一个解，该种算法采取逐人考虑方法，先随机生成第一个人的人物陈述，再通过以下流程图所示的方法，生成第二、第三个人的人物陈述。



由于以上算法不需要进行任何枚举，因此这种算法的时间复杂度是O(1)。

由此可见，这种算法的优势是：时间复杂度小，出题一步到位，不生成不符合条件的题目。但其也存在着判断分支冗杂、代码量大的不足。

5.1.2.2 有否定句形式的出题：基于枚举的多次随机题试解

为了既保证出题的完全随机性，又保证出的题有且仅有一个解，该种算法采取纯随机出题的方法，随机赋予三个角色各自的声明，形成一道题进行试解。若出现多解情况或无解情况，再次随机出一道题进行试解。如此循环直至一道有且仅有一个解的题目出现。

由于以上算法生成的用于试解的题目完全随机，因此有可能需要付出较多的时间代价来产生一道有且仅有一个解的题目。

由此可见，这种算法的优势是：思路简单、代码量小，将判断分支由人工列举改为程序随机试解。但其存在着运算时间不确定、期望运行时间较大的不足。

5.2 “正确/错误/部分”类型问题

和上一类问题相似，为了平衡不同题目代码量和时间复杂度的代价差异，在“正确/错误/部分”类型问题中，对于不同难度的问题采取了不同的算法。对于较简单的问题，通过多个判断分支进行剪枝，减少无效枚举，从而减少程序运算量，降低时间复杂度；对于较复杂的问题，付出一定时间复杂度的代价，将判断分支的剪枝过程由人工列举改为程序枚举。下文将对这些算法进行详细介绍。

解决问题、出题采取的算法对应表如下。

	背景 1~3（简单）	背景 4~5（中等/困难）
解题	积分制判断是否符合条件	
出题	基于逻辑的逐人逐句随机生成 （搜索次数少，代码量大、判断分支多）	基于枚举的多次随机题试解 （搜索次数多且随机，代码量小、判断分支少）

5.2.1 解题

该种算法包含以下几个步骤：首先，通过遍历枚举猜测这个物品种类。其次，对每个人所说的第一句正确判断赋予 5 分，第二句正确判断赋予 3 分。最后，计算所有判断赋分之积为最终总分，考虑总分是否符合条件。

由于该算法需要分别枚举物品种类各一次，并遍历每句话。因此，这种算法的时间复杂度是$O(n^2)$。

这样的积分逻辑优势是具有较强的可复用性。若需要出 5 人、6 人局面的更难的题目，随着人数的增加，只需以一定倍数修改要求的总分即可。

5.2.2 出题

5.2.2.1 背景 1~3：基于逻辑的逐人逐句随机生成

为了既保证出题的完全随机性，又保证出的题符合中文语法和语言逻辑，还要保证出的题有且仅有一个解，该种算法包含以下几个步骤。

首先，随机生成一个实际物品，并随机生成 3 名角色的正确性。

其次，按照以下步骤随机生成三个人的每一句话。（注 1：题中实际物品记为 thing，另外两件物品记为 other1thing、other2thing；注 2：表格字体为红色的话语为判断错误的话。）

1.1 将“不是 thing”分配为人物 1 的第一句话。

1.2 在除了 thing 中的另外两件物品中随机挑选一个，记为 other1thing，将“不是 other1thing”分配为人物 1 的第二句话。

2.1 在三件物品中随机挑选一个，作为人物 2 的第一句话的宾语。

2.2 根据下表所述对应规则，随机生成人物 2 可能的第二句话。

人物 1 （随机生成句子 2）	人物 2 句子 1 （随机生成）	人物 2 句子 2 （根据句子 1 随机生成）
不是 thing 而是 other1thing	不是 thing	不是 other2thing
	不是 other2thing	而是 other1thing
		不是 thing

	不是 other1thing	而是 other2thing

3 根据下表所述规则，分两步随机生成人物 3 可能的两句话

人物 2 的错判	人物 2 的正判	人物 3 句子 1 （根据人物 2 随机生成）	人物 3 句子 2 （根据句 1 随机生成）
而是 other1thing	（随意）	不是 other2thing	而是 thing
			不是 other1thing
		不是 other1thing	不是 other2thing
不 thing	不 other1thing	不是 other1thing	不是 other2thing
	不 other2thing		而是 thing
			不是 other2thing
	（随意）	不是 other2thing	而是 thing
			不是 other1thing
而是 other2thing	（随意）	不是 other1thing	而是 thing

由于以上算法不需要进行任何枚举，因此这种算法的时间复杂度是O(1)。

由此可见，这种算法的优势是：时间复杂度小。但其也存在着判断分支冗杂、代码量大的不足。

5.2.2.2 背景 4~5：基于枚举的多次随机题试解

为了既保证出题的完全随机性，又保证出的题有且仅有一个解，该种算法采取纯随机出题的方法，随机赋予三个角色各自的两句陈述，形成一道题进行试解。若出现多解情况或无解情况，再次随机出一道题进行试解。如此循环直至一道有且仅有一个解的题目出现。为了实现背景 4、背景 5 在人工解题难度上的区分，在循环的判断语句中还加入了对“不是 A，而是 B”语句数量的判定条件，从而形成不同难度的题目。

由于以上算法生成的用于试解的题目完全随机，因此有可能需要付出较多的时间代价来产生一道有且仅有一个解的题目。

由此可见，这种算法的优势是：思路简单、代码量小，将判断分支由人工列举改为程序随机试解。但其存在着运算时间不确定、期望运行时间较大的不足。

六、程序结构

6.1 程序功能及其实现所使用的类

程序功能	类(对象)	类(对象)的功能
界面选项信息→游戏属性	Property (gameproperty)	构造并记录游戏属性
记录人物信息	Character (character[4])	构造并记录 4 位人物信息
实现自然语言段落结构	Paragraph (para)	构造并记录自然语言段落关键字
记录各类问题的问题属性	TypeAProperty (TypeAproperty)	记录“牧师/骗子/赌棍”类型的问题属性
	TypeBProperty (TypeBproperty) (TypeBRandomproperty)	记录“正确/错误/部分”类型的问题属性和随机问题属性

6.2 程序功能及其实现所使用的函数

程序功能	函数	函数的功能
界面选项信息→界面属性	UIMatch()	根据选择的选项修改界面属性
	IdentityBoxLabelChange()	改变身份下拉选择栏和身份答案栏的内容和格式
	paraAnswerRename()	改变消息框答案关键词
界面选项信息→游戏属性	所有元件触发函数	根据选择的选项修改游戏属性
实现自然语言段落结构	StatementMatch()	匹配类型 1 人物声明及自然语言
	GuessMatch()	匹配类型 2 人物声明及自然语言
	ParagraphTrans()	将问题和每个人的对话框语句转化为自然语言
问题参数设置	DefaultSetting()	设置默认问题的参数
	RandomSetting()	设置随机问题的参数
求解/出题过程参数初始化	TypeAInitialize()	类型 1 解题功能参数初始化
	TypeBThingInitialize()	类型 2 解题事物参数初始化
	TypeBInitialize()	类型 2 解题功能参数初始化
	TypeBRandomInitialize()	类型 2 随机出题功能参数初始化
求解问题	TypeA1_Solve()	求解类型 1 形式 1 问题
	TypeA2_Solve()	求解类型 1 形式 2 问题
	TypeB_Solve()	求解类型 2 问题

检查答案与消息显示	AnswerDisplay()	在时间终了或点击计算/提交按钮后显示答案
	MessageDisplay()	在计算机解题模式下弹出对话框显示提示信息和答案
	AnswerCheck()	在人工解题模式下确认提交的答案是否正确，并弹出对话框显示提示信息和答案
倒计时	timer_Tick()	倒计时

七、项目总结与思考

这次的项目和我之前两年所做过的项目大有不同，这体现在之前如此规模的项目大多是在小学期时段与队友合作完成。而这次项目是在日常学期之中一人独立完成的。几天时间内徒手独立完整写上 1600 多行代码也属第一次。在代码编写过程中，我不时产生了多种算法思路。而其中让我产生感悟最大的便在于搜索算法的实现和类型 2 问题的求解上。

首先，在搜索算法实现上，编程解题、出题可以有多种想法。其中最简单的想法应该就是完全枚举。即：在解题时，可以完全枚举所有可能的结果，遇到可行解输出；在出题时，可以随机不断出题，并让程序不断解题，直到生成一道有且仅有一个解的题目。然而，这样的“暴力搜索”势必造成很大的不必要计算，体现在解题时多次无效枚举和出题时可能生成不符合条件的题目，时间复杂度将会变得很大。因此，在形式相对简单易想的问题的解题或出题上上，我使用了逐人、逐句的层层递推。使用这个方法的一大优势便是：解题时通过人工提前剪去所有不可能的枝，减少无效枚举，从而减少程序运算量。出题时一步到位，不生成不符合条件的题目。

为了平衡不同题目代码量和时间复杂度的代价差异，在最后的算法实现上，对于不同难度的问题使用了不同的出题算法和解题算法。而其中基于逻辑的算法具有较小的时间复杂度，基于枚举的算法具有较小的代码量和较强的可复用性。可谓是各有千秋。

第二，在类型 2 问题的求解时，通过积分制减少了枚举数量和空间复杂度。例如，在 3 人局面中，下表所示两种情况均为 3 句正确的情况，然而可以发现第二种情况是不符合的。

甲		乙		丙		是否符合
正	正	正	误	误	误	是
正	误	正	误	正	误	否

而考虑 4 人局面，4 句正确的情况更为多样，例如四个人的正确判断数可以为 2+2+0+0，也可以为 2+1+1+0，也可以为 1+1+1+1，而仅仅第二种情况是符合要求的。

最开始产生的一个思路是，开一个数组记录每个人正确陈述的数量。在最后判断时搜索每个人正确陈述的数量是否满足 2+1+0 或 2+1+1+0 的组合。然而，这势必造成搜索量的提升和空间的使用，且在人数增加时需要重新写一层循环代码，可复用性较差。

因此，为了解决以上问题，对这类问题的解决引入了积分制，对每个人所说的第一句正确判断赋予 5 分，第二句正确判断赋予 3 分，最终总分为所有判断赋分之积，仅考虑总分是否符合条件。那么，对于不同的正误组合，总分都是截然不同的，排除了不符合要求的情况；而对于个人的正误则不做关心，免去了再次循环枚举三人正误判断个数的运算量。

这样的积分逻辑具有较强的可复用性。若需要出 5 人、6 人局面的题目，随着人数的增加，只需以一定倍数修改要求的总分即可。

本次大作业至此已经完全完成，在完成过程中，通过对于算法的思考和优化令我收获颇丰，也大大加深了我对搜索相关知识的了解。