COMP9414: 人工智能作业1：周计划表

交付日期。第6周，7月6日，星期三，晚上11:59。

价值: 15%

这项作业的动机是在繁忙的一周中安排你所有的个人活动，包括大学学习、工作、吃饭、旅行等等。在活动的日期和时间上既有*约束*也有*偏好*。这些约束是 "硬 "约束（在任何解决方案中都不能违反），而偏好是 "软 "约束（可以或多或少地满足）。每个软约束都有一个每小时的成本，即未能在首选时间安排活动的 "惩罚"（我们将不考虑对日子的偏好）。我们的目标是安排所有的活动，使所有成本的总和*最小化*，并满足所有的约束。

更准确地说，让我们假设活动被安排在周日到周六的某一天，从早上7点到晚上7点的某个时间开始。每个活动将被赋予一个固定的持续时间（小时），并且只在一天内发生（注意，一个活动有可能在晚上7点以后结束）。约束可以是指活动的开始和结束时间或活动的日期，也可以是活动之间的关系（比如 "讲座 "必须在 "辅导 "之前）。一个偏好是指*一个*活动应该*在*给定的时间*前后*开始（忽略了日期）。限制条件和偏好的完整列表定义如下。

作为一个例子，我们可能会在周一安排一个 "晚餐 "活动，从晚上7点开始，持续1个小时--因此活动将在晚上8点结束。你的代码**不需要**检查活动是否在开始的同一天结束：你可以**假定**这一点。

从技术上讲，这个作业是一个*约束优化问题*的例子，这个问题有像标准约束满足问题（CSP）一样的约束，但也有与每个解决方案相关的*成本*。在这项任务中，你将实现一个*贪婪*算法，以寻找这些在文件中指定的调度问题的最佳解决方案。然而，与搜索讲座中描述的贪婪搜索算法不同，这种贪婪算法的特性是它能保证为任何此类问题找到一个最优解（如果有解存在的话）。

你*必须*使用AIPython代码进行约束满足和搜索，开发一种贪婪的搜索方法，使用成本来指导搜索，就像启发式搜索一样（启发式搜索与A* 搜索相同，其中路径成本都为零）。搜索将使用一个由启发式函数值排序的优先级队列，该函数为搜索中的每个节点提供一个成本。在此分配中使用的启发式函数定义如下。这个搜索中的节点是CSP，即*每个*状态都是一个具有变量、域和相同约束（以及成本估计）的CSP。状态空间中的转换在弧形一致性的前提下实现域的分割（AIPython代码实现了这一点）。一个目标状态是对所有变量的赋值，满足所有的约束条件。一个解决方案的成本是时间表中各项活动的成本之和。

这项任务的CSP是一组代表活动的变量，对活动的二进制约束，以及对活动的单进制约束（硬或软）。领域是 "日"、"月"、"二"、"三"、"四"、"五 "和 "六 "的所有组合，以及 "早7点"、"早8点"、"早9点"、"早10点"、"早11点"、"晚12点"、"晚1点"、"晚2点"、"晚3点"、"晚5点"、"晚6点 "和 "晚7点"。因此，可能的值是一天的时间对，如'上午9点'。每个活动名称是一个字母或数字的字符串（没有空格）。

可能的输入（任务和约束）如下。

```
# 活动的名称和持续时间 活动
（名称）（持续时间）

# 二元约束
约束（A1）在（A2）之前      # A1在A2开始时或之前结束约束(A1)在
(A2)之后                    # A1在A2结束之后或之后开始约束(A1)
开始(A2)                    # A1和A2在相同的日期和时间开始约
束(A1)结束(A2)              # A1和A2在同一天和时间结束
约束（A1）与（A2）重叠      # A2在A1开始之后开始，而不是在A1结束之后。
                            # 并在A1结束后结束
约束(A1)在(A2)期间          # A1在A2开始后开始，在A2结束前结束，约束
(A1)等于(A2)                # A1和A2在同一天和同一时间开始和结束约束
(A1) 同一天(A2)             # A1和A2在同一天开始和结束

# 硬域限制
域(A)在(d)上                # A开始（和结束）于d 日，域(A)
在(d)之前                   # A在d 日之前开始（和结束） 域
(A) after (d)               # A在d日之后开始（和结束）。
域(A)开始-之前(t)           # A在任何一天的时间t或之前开始域
(A) starts-after (t)        # A在任何一天的时间t或之后开始域
(A) ends-before (t)         # A在任何一天的时间t或之前结束 域
(A) ends-after (t)          # A在任何一天的时间t或之后结束

# 软领域约束
域（A）周围（t）（成本）。  #不满足时间偏好的每小时成本t
```

为了定义一个解决方案的成本（可能只是部分满足软约束），将违反所有活动的软约束的相关成本相加。让V是变量（代表活动）的集合，C是所有软约束的集合。假设这样一个具有时间偏好t_c 和*成本费用*$_c$ *的*约束适用于变量v，让(d_v , t_v)为v在解决方案S中的起始日和时间。例如，*成本费用*$_c$ 可能是10，(d_v , t_v)可能是(mon, 5pm)，而首选时间t_c 是3pm；这个变量分配的成本是20（2小时差×成本10）。

t_1 和 t_2 之间的时间差（转换为整数小时）只是
$t_1 - t_2$ ，表示$|t_1 - t_2|$。然后，其中c_v 是适用于变量v的软约束。

$$cost(S) = \sum_{c_v \in C} 成本_{c_v} * |t_v - t_{c_v}|$$

启发式

在这项作业中，你将使用一个优先级队列实现贪婪搜索，根据启发式函数h对节点进行排序。这个函数必须接受一个任意的CSP，并返回从任何状态S到解决方案的距离的估计值。因此，与一个解决方案相比，每个变量v都与*一组*可能的值（当前域）相关。

启发式估计从一个给定的状态S所能达到的最佳解决方案的成本，假设每个变量可以被分配到适用于该变量的软约束成本最小的值。启发式函数对所有变量集的这些最小成本进行加总，类似于计算一个解决方案的成本$cost(S)$。让S是一个带有变量V的CSP，让v的域，写成$dom(v)$，是v的一组时间（忽略分配给v的日子）。然后，其中的总和是对所有软约束c_v ，如上所述。

$$h(S) = \sum_{c_v \in C} \min_{t_v \in dom(v)} cost_{c_v} * |t_v - t_{c_v}|$$

实施

把你**所有的**代码放在一个叫做weekPlanner.py的Python文件中。你可以（在一两种情况下）从 AIPython 中复制代码到 weekPlanner.py 并修改这些代码，但不要把大量的AIPython 代码复制到你的文件中。相反，最好是在weekPlanner.py中写一些扩展AIPython类的类（下面附录中绿色的类）。

使用 searchGeneric.py 中的通用搜索算法的 Python 代码。这段代码包括一个具有方法search() 的 Searcher 类，它使用一个列表（被当作堆栈）实现深度优先搜索，以解决任何搜索问题（如 searchProblem.py 中定义的）。对于这个任务，扩展 AStarSearcher类，该类扩展了 Searcher 并利用优先级队列来存储搜索的前沿。根据使用启发式函数计算出的节点的成本对优先级队列中的节点进行排序，但要确保路径成本始终为0。使用这段代码时，将从输入创建的CSP问题传入searchProblem（子）类，使其成为一个搜索问题，然后将这个搜索问题传入Searcher（子）类，当对这个搜索问题调用search()方法时运行搜索。

使用cspProblem.py中的Python代码，它定义了一个带有变量、域和约束的CSP。通过扩展这个类来增加CSP的成本，包括一个成本和一个启发式函数h来计算成本。也可以使用cspConsistency.py中的代码。这段代码实现了解决CSP所需的状态空间的转换。该代码包括一个带有CSP的AC的Search类，该类调用了一个用于领域分割的方法。每当一个CSP问题被分割时，产生的CSP都会被做成弧形一致（如果可能的话）。与其扩展这个类，你可能更喜欢写一个新的类 Search with AC from Cost CSP，它有同样的方法，但可以处理过度约束优化问题。这只是在相关的方法中加入成本，并修改构造函数，以便在创建新的CSP时通过计算h来计算成本。

你应该提交weekPlanner.py和所有其他运行你的程序所需的AIPython文件。weekPlanner.py中的代码将与你提交的AIPython文件在同一目录下运行。你的程序应该从标准输入中读取输入（即**不是**硬编码的输入1.txt），并将输出打印到标准输出（即**不是**硬编码的输出1.txt）。

输入样本

所有的输入将是定义活动、二进制约束和领域约束的行的序列，按顺序排列。注释行（以'#'字符开头）也可能出现在文件中，你的程序应该能够处理和丢弃这些行。所有的输入文件都可以被认为是正确的格式--不需要对输入文件进行任何错误检查。

下面是一个输入表格和意义的例子。请注意，你必须随你的作业提交至少三个输入测试文件。这些测试文件应包括一个或多个注释，以说明测试的是什么情景。

```
#在同一天有两个活动，而时间偏好无法满足，活动讲座3
活动教程1
# 两种二元约束的约束讲课前
教程
约束讲座当天的教程#领域约束
领域讲座
领域讲座开始--下午1点前领域讲座
开始--下午1点后领域辅导下午3点
左右 10
```

输出样本

使用Python的标准打印函数将输出结果打印成一系列的行，给出每个活动的开始日期和时间（按照活动的定义顺序）以及最优解的成本。如果问题没有解决方案，就打印 "没有解决方案"（大写的 "N"）。如果有多个最优解，你的程序应该产生其中任何一个。**重要的是。**对于自动标记，要确保在行尾没有多余的空格，在打印出代价后没有多余的空行（即在显示代价的解决方案的最后一行后没有额外的换行字符）。这是Python打印函数的标准行为。将AIPython代码中的所有显示选项设置为0。

与上述输入对应的输出如下。

讲座：周一下午1点
辅导：周一下午4点
费用：10元

提交

- 使用以下命令提交你的所有文件（这包括相关的AIPython代码）。

    Give cs9414 ass1 weekPlanner.py search*.py csp*.py display.py *.txt

- 你提交的材料应包括。
  - 你的.py源文件，包括运行你的代码所需的任何AIPython文件
  - 至少有三个用于测试你的系统的输入文件（包括说明所测试的情景的注释），以及相应的输出文件（称其为input1.txt、output1.txt、input2.txt、output2.txt等）；**只提交正确格式的输入文件**
- 当你的文件提交后，将进行测试以确保你的Python文件在CSE机器上的**9414环境下**运行；注意任何错误信息
- 使用命令检查是否收到你提交的材料。

    9414 classrun -check ass1

评估

本作业的分数分配如下。

- 正确性（自动记分）。10分
- 编程风格。5分

迟到的惩罚。你的分数每迟一天或部分时间减少0.75分，在到期日之后的5个日历日内，分数为0。

评估标准

- 正确性。对*有效的*输入测试进行评估，如下所示，输入是从重定向到标准输入的文件中读取的，而输出是重定向到标准输出的（**不是**硬编码的文件名）。

    python3 weekPlanner.py < input1.txt > output1.txt

- 编程风格。可理解的类和变量名称，易于理解的代码，对AIPython代码的良好重用，充分的注释，合适的测试文件

剽窃行为

请记住，为这项作业提交的所有作品必须是你自己的作品，不允许分享或复制代码。你可以使用互联网上的代码，但必须在你的程序中适当注明来源。**不要使用github等网站上的公共代码库，如果你使用代码库，请确保你的代码库是私有的。**所有提交的作业将通过抄袭检测软件来检测与其他提交的作业的相似性，包括过去几年的。在课程期间**和结束后，**都不要与任何人分享你的代码。你应该仔细阅读新南威尔士大学关于学术诚信和抄袭的政策（从课程网页上链接），特别是注意到*串通*（一起完成作业，或分享部分作业解决方案）是一种抄袭形式。

不要使用任何来自合同作弊 "学院 "的代码或 "辅导 "服务。这是严重的不当行为，会受到严重的惩罚，甚至会被自动取消课程，并得到0分。

附录。AIPython类