离散优化建模：作业六
称象：第二部分

1 问题描述

曹冲已经设法令到大象上船了，而且把船身由于大象的重量而没入水中的位置已经标记好了。他现在需要移动石头到船上来，令到船下沉到相同的位置，然后称石头的重量。

他有 g 个守卫来移动石头。他们可以轻松地搬动一些石头而不感到疲惫。他们也可以搬动更多的石头，只不过这样他们会感到疲惫，而且有一阵子不能再搬动石头。在码头上，河岸与船之间移动的守卫的数目是有限制的，而且在任何的时间里所有在码头上的守卫应该向相同的方向移动，也就是要么都是从河岸到船，要么都是从船到河岸。而在任意时间里船上停留的人的数量也是有限制的。曹冲需要计划如何用最少的步数把至少 E 块石头从河岸搬到船上。

2 数据格式说明

称象问题的输入是名为 `data/elephant2_p.dzn` 的文件，其中 p 是问题的序号。T 是最大允许的步数。E 是最少应该搬到船上的石头。G 是最多可使用的守卫数。$easy$ 是一个数组代表每个守卫可以搬运多少石头而不感到疲惫。$hard$ 是一个数组，代表每个守卫可以搬运的石头数目的上限。$tired$ 是一个数组，代表每个守卫在搬了多于 $easy[g]$ 块石头之后需要休息的步数。p 是码头上在船和岸边的路上最多可以有多少守卫同时移动。b 是船上最多可以同时容纳的守卫的数目。

搬运的计划可以用每个守卫每个时间做的事情来表示，他们可以：

$-\mathbf{1}$: 表示从船走向岸边;

$\mathbf{0}$: 表示在原地等待（休息）;

$\mathbf{n} > \mathbf{0}$: 表示把 n 块石头从岸边移动到船上。

搬运计划结束的时间应该是有至少 E 块石头在船上的那一步。在结束之后所有守卫只能原地等待。

所以数据和决策变量的声明如下：

```
int: T; % maximum time allowed;
set of int: TIME = 1..T;

int: E; % weight of elephant in STONES;
set of int: STONE = 0..E;
```

```
int: G; % number of guards
set of int: GUARD = 1..G;
array[GUARD] of STONE: easy;
array[GUARD] of STONE: hard;
array[GUARD] of TIME: tired;

GUARD: p; % maximum people on pier;
GUARD: b; % maximum people on boat;

set of int: ACT = -1..E; % -1 = goto bank, 0 = wait, > 0 carry stones
int: wait = 0;
int: to_bank = -1;
array[GUARD,TIME] of var ACT: act;        % action at time t
var TIME: end;                            % end time;
```

数据文件的样本如下：

```
T = 10;
E = 50;
G = 4;

easy = [5,5,5,5];
hard = [10,10,10,10];
tired = [3,3,3,3];
p = 3;
b = 4;
```

这里面表示有 4 个相同能力的守卫，每一个可以搬运 5 块石头而不觉得累。他们最多可以搬运 10 块石头，不过他们需要 3 步来休息。码头上最多能容纳的人数是 3，船上最多能容纳的人数是 4。

你的输出应该是需要用的步数（`end`），和每一步每个守卫采取的行动（`act`）。比如上述例子对应的一个答案是：

```
act = array2d(GUARD,TIME,[
10,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 5, -1, 10,  0,  0,  0,  0,  0,  0,  0,
 5, -1, 10,  0,  0,  0,  0,  0,  0,  0,
 0,  0, 10,  0,  0,  0,  0,  0,  0,  0]);
```

```
end = 3;
```

这表明在第一步 1 号守卫搬运 10 块石头，而 2 号和 3 号守卫搬运 5 块石头。2 号 3 号守卫在第二步中返回，而在第三步中 2 号 3 号 4 号搬运 10 块石头。在三步之后船上就有了 50 块石头。需注意到三步之后守卫全部在原地等待。

对于数据文件：

```
T = 10;
E = 40;
G = 2;

easy = [5,10];
hard = [10,15];
tired = [3,4];
p = 1;
b = 2;
```

一个样例输出是：

```
act = array2d(GUARD,TIME,[
 0,  0,  5, -1, 10,  0,  0,  0,  0,  0,
15, -1,  0,  0,  0, 15,  0,  0,  0,  0]);
end = 6;
```

需注意到 2 号守卫虽然在第一步中虽然已经搬了很多石头而在第二步中感到疲惫，不过他还是可以在第二部走回岸边。

需注意到在这里的 `act` 的输出格式并不是必须的。这只是为了方便说明题解。

3 指引

你可以编辑 `elephant2.mzn` 模型文件来解决上述优化问题。你实现的模型 `elephant2.mzn` 可以用提供的数据文件进行测试。在 MiniZinc IDE 中，你可以通过点击 *Run* 按钮在本地测试和运行。或者在命令行中输入

```
mzn-gecode ./raid.mzn ./data/<inputFileName>
```

进行本地测试和运行。两种情况下，你的模型都是用 MiniZinc 进行编译同时用 GECODE 求解器求解。

参考资料 你可以在 `data` 文件夹下找到讲义中的几个问题实例（的数据文件）。

提交作业 这次的作业包含有 4 个答案提交部分和 1 个模型提交部分。对于答案提交部分，我们将会提交求解器求解你的模型所得到的最好 / 最后的答案，然后检查它的正确性和得分。对于模型提交部分，我们将会提交你的模型文件 (.mzn) 然后用一些隐藏的数据文件来做进一步检查。

在 MiniZinc IDE，点击 *coursera* 图标可以用于提交作业。若采用命令行方式，`submit.py` 可以用于提交作业。无论采用那种方法，你都需要根据本指引中的要求完成作业各部分的 MiniZinc 模型。你可以多次提交，最终作业分数是你的最高的一次。[1]作业的打分过程可能需要几分钟，请耐心等候。你可以在课程网站上的*编程作业* 版块查看你的作业提交状况。

4 软件要求

为了完成作业，你需要安装MiniZinc 2.1.x和GECODE 5.0.x 求解器。这些软件都会包含在MiniZinc IDE 2.1.2 (`http://www.minizinc.org`)的集成版本中。如果你需要通过命令行提交作业，你需要安装Python 3.5.x。

[1]答案提交部分并没有次数限制。但是，模型提交部分只能提交有限次。