CMSC 474: Project 1

last updated 5:30pm, September 13, 2016

Due date: Friday, Sept 30, 11:59:59pm

Late date (10% penalty): Sunday, Oct 2, 11:59:59pm

Your assignment is to write a Java program to do Iterated Elimination of Strictly Dominated Strategies (IESDS).

Input. The input is a sequence of the following numbers, to specify a payoff matrix and tell what to do with it:

- The integer 1 or 0, followed by a linefeed. 1 means to do IESDS on the matrix, 0 means to just print out the matrix.
- An integer $n \leq 5$ denoting the number of players, followed by a linefeed.
- A line containing n integers $m_1\ m_2\ \ldots\ m_n$ followed by a linefeed. Each m_i tells how many actions player i has, and each player has at most 10 actions (thus $1 \leq m_i \leq 10$).
- The numbers in the payoff matrix, in row-major order, with a linefeed after each row.

For example, here's an example from Lecture 3a, with numbers for the actions instead of names:

	1	2	3
1	3, 1	0, 1	0, 0
2	1, 1	1, 1	5, 0
3	0, 1	4, 1	0, 0

Here is the corresponding input, with the 1 on the first line indicating that IESDS should be performed.

```
1
2
3 3
3 1 0 1 0 0
1 1 1 1 5 0
0 1 4 1 0 0
```

Output. The first line of the input tells your program whether to do IESDS or just print out the matrix. If your program does IESDS, it should start by printing out the matrix, and each time it eliminates a strategy it should print out the reduced matrix.

Each time your program prints a matrix, it should use the following format:

- An integer n denoting the number of players, followed by a linefeed.
- For each player i (in ascending order of i), a line giving a list of i's remaining actions (i.e., all actions that haven't been eliminated), followed by a linefeed. In other words, put player 1's remaining actions on a line, then put player 2's remaining actions on the next line, etc.

  In each line, the actions should be in increasing order. For example, if player i's remaining actions are 1 and 3, you should print `1 3` rather than `3 1`.
- The numbers in the payoff matrix, in row-major order, with a linefeed after each row.

If your program prints out more than one matrix, it should put a blank line after each one except the last. In the earlier example, the printout would be

```
2
1 2 3
1 2 3
3 1 0 1 0 0
1 1 1 1 5 0
0 1 4 1 0 0

2
1 2 3
1 2
3 1 0 1
1 1 1 1
0 1 4 1

2
1 3
1 2
3 1 0 1
0 1 4 1
```

As illustrated in the above example, your program should be capable of handling cases in which the dominating strategy is either a pure strategy or a mixture of two pure strategies.

If the first line of the input were 0 rather than 1, then the printout would be just the first one of the above three matrices.

1 Submission Instructions

We will use the submit server (submit.cs.umd.edu) to handle and test your submissions. To submit your project, please upload a single java file, `IESDS.java`. This java file must include a public static function, main(), with the usual header:

```
public static void main(String[] args)
```

The file may also include other helper methods or classes.

The Javadoc of your class should contain a description of how your program works. At the end of your comment, put the student honor pledge:

```
/* I pledge on my honor that I have not given or received any unauthorized
   assistance on this project.  (your name, your email address) */
```

Your program should read from stdin and write to stdout. It shouldn't read nor write to any files. Before submitting your program, be sure to remove all print/debug statements.

Feel free to post questions and comments on Piazza.

2 Grading Criteria

The grading criteria will be as follows:

- 10 points for programming style.
- 10 points for documentation. This should include Javadoc comments explaining how your program works, and any special techniques and special data structures to speed up the search process.
- 10 points if your program correctly processes input and output. We'll confirm this by running test inputs whose first line contains 0 rather than 1.
- 35 points if your program can correctly remove all strategies that are strictly dominated by pure strategies.
- 35 points if your program can correctly remove all strategies that are strictly dominated by mixtures of two pure strategies.
- *extra credit*: 20 points if your program can correctly remove all strategies that are strictly dominated by mixtures of three pure strategies.

3 Hints

Let c, d, e be three actions for player i. A mixed strategy $\sigma_p = \{(p, d), (1-p, e)\}$ strictly dominates c if and only if for every action profile $\mathbf{a}$ of the other players, $u_i(\sigma_p, \mathbf{a}) > u_i(c, \mathbf{a})$. Here's how to find the set of all mixed strategies σ_p having that property.

- Let $\mathbf{a}_1, \ldots, \mathbf{a}_k$ be all of the action profiles of the players other than i. For every j, find the widest possible interval $q_j < p < r_j$ such that $u_i(\sigma_p, \mathbf{a}_j) > u_i(c, \mathbf{a}_j)$. If q_j and r_j exist, you should be able to find them pretty easily by solving some linear equations. If they don't exist, there's no σ_p that strictly dominates c.
- If q_j and r_j exist for every j, then let (q, r) be the intersection of all of the intervals (q_1, r_1), (q_2, r_2), $\ldots$, (q_k, r_k). If this intersection is nonempty, then for every $p \in (q, r)$, σ_p strictly dominates c. Otherwise, there's no σ_p that strictly dominates c.

If you want to find out whether c is strictly dominated by a mixture of more than two strategies, that's harder. Try doing a web search for the topic "linear programming".