

# Programming Assignment

Game Theory

February 22, 2021

## Notes

## Teams

This is a team assignment. Odd numbered teams should attempt Question 1 while even numbered teams should attempt Question 2.

## Grading

The code will be evaluated automatically on a number of test cases and marks will be awarded accordingly.

In order to help in verifying your code, you may submit your own custom test cases to the TA along with your output and the TA will check it against the model solution. (You can message Anurag Jain on Teams)

## Programming Languages

You may attempt the assignment in C/C++ or Python.

You may not use any external libraries unless explicitly permitted by the TAs.

## Plagiarism

Please do not copy code from your friends or online sources. Plagiarism cases may be penalized.

## Submission Format

You may upload exactly one file containing the source code with the file name as `<Team_No>.c` for C Programs or `<Team_No>.cpp` for C++ Programs or `<Team_No>.py` for Python Programs. There should be only one submission per team.

## Deadline

- Soft Deadline: 4<sup>th</sup> March 2021 **8:00 PM**
- Hard Deadline: 7<sup>th</sup> March 2021 **8:00 PM** (with 15% Penalty)

## 1 Find PSNE and Weakly Dominant Strategies for a $n$ -Player Game

Given a  $n$ -Player Game list all Pure Strategy Nash Equilibria and list all Weakly Dominant Strategies for each player.

## 1.1 Input Format

The input is a  $n$ -Player Game with the payoffs listed in the NFG Format (as described in the Gambit Project).

- First line contains the number of players  $n$ .
- The second line contains  $n$  space-separated numbers, the  $i^{\text{th}}$  number corresponding to the number of strategies available to the  $i^{\text{th}}$  player. ( $S_i$ )
- The third line contains the list of payoffs in the NFG Format.

## 1.2 Output Format

- First line should contain the number of PSNE. ( $n_{\text{psne}}$ )
- Followed by  $n_{\text{psne}}$  lines, the  $i^{\text{th}}$  line containing  $n$  space-separated numbers corresponding to the equilibrium strategies for each player respectively.
- Next, you should output  $n$  lines, with the  $i^{\text{th}}$  line listing the number of weakly dominant strategies for the  $i^{\text{th}}$  player followed by the dominant strategies.

## 1.3 Constraints

- $\prod_{\forall i} |S_i| \leq 10^6$  where  $|S_i|$  is the number of strategies available to the  $i^{\text{th}}$  player.
- Time Limit: 10 seconds per test case
- Memory Limit: 1 GB

## 1.4 Sample Test Cases

### Input 1

```
2
2 2
-2 -2 -2 -10 -10 -2 -5 -5
```

### Output 1

```
2
1 1
2 2
1 2
1 2
```

### Input 2

```
2
3 3
6 6 10 0 8 0 8 20 5 5 20 0 0 8 2 8 4 4
```

### Output 2

```
1
3 3
0
0
```

### Input 3

```
2
4 4
5 2 0 0 7 0 9 5 2 6 3 2 2 2 1 3 1 4 2 1 1 5 0 2 0 4 1 1 5 1 4 8
```

### Output 3

```
1
2 2
0
0
```

## 2 Find All Nash Equilibria of a Bimatrix Game

Given a Bimatrix Game, find all Nash Equilibria of the Bimatrix Game. If there are multiple Nash Equilibria, output in any order.

### 2.1 Input Format

The input is a *non-degenerate* bimatrix game with the payoffs listed in the NFG Format (as described in the Gambit Project).

- First line contains the number of strategies for the row player. ( $R$ )
- Second line contains the number of strategies for the column player. ( $C$ )
- The third line contains the list of payoffs in the NFG Format.

### 2.2 Output Format

- The first line must contain the number of equilibria. ( $n$ )
- The next  $2n$  lines must contain the description of the  $n$  equilibria as follows:
  - $2i + 1^{\text{th}}$  line contains the  $R$  floating point numbers  $\in [0, 1]$  denoting the probability that the row player should play the  $R$  strategies respectively.
  - $2i + 2^{\text{th}}$  line contains the  $C$  floating point numbers  $\in [0, 1]$  denoting the probability that the row player should play the  $C$  strategies respectively.

### 2.3 Constraints

- $R + C \leq 25$
- Time Limit: 10 seconds per test case
- Memory Limit: 1 GB

### 2.4 Sample Test Cases

#### Input 1

```
2
2
-2 -2 -1 -10 -10 -1 -5 -5
```

### Output 1

1  
0.0 1.0  
0.0 1.0

### Input 2

2  
2  
2 1 0 0 0 0 1 2

### Output 2

3  
1.0 0.0  
1.0 0.0  
0.0 1.0  
0.0 1.0  
0.6666666666666666 0.3333333333333333  
0.3333333333333333 0.6666666666666666