GT-Project

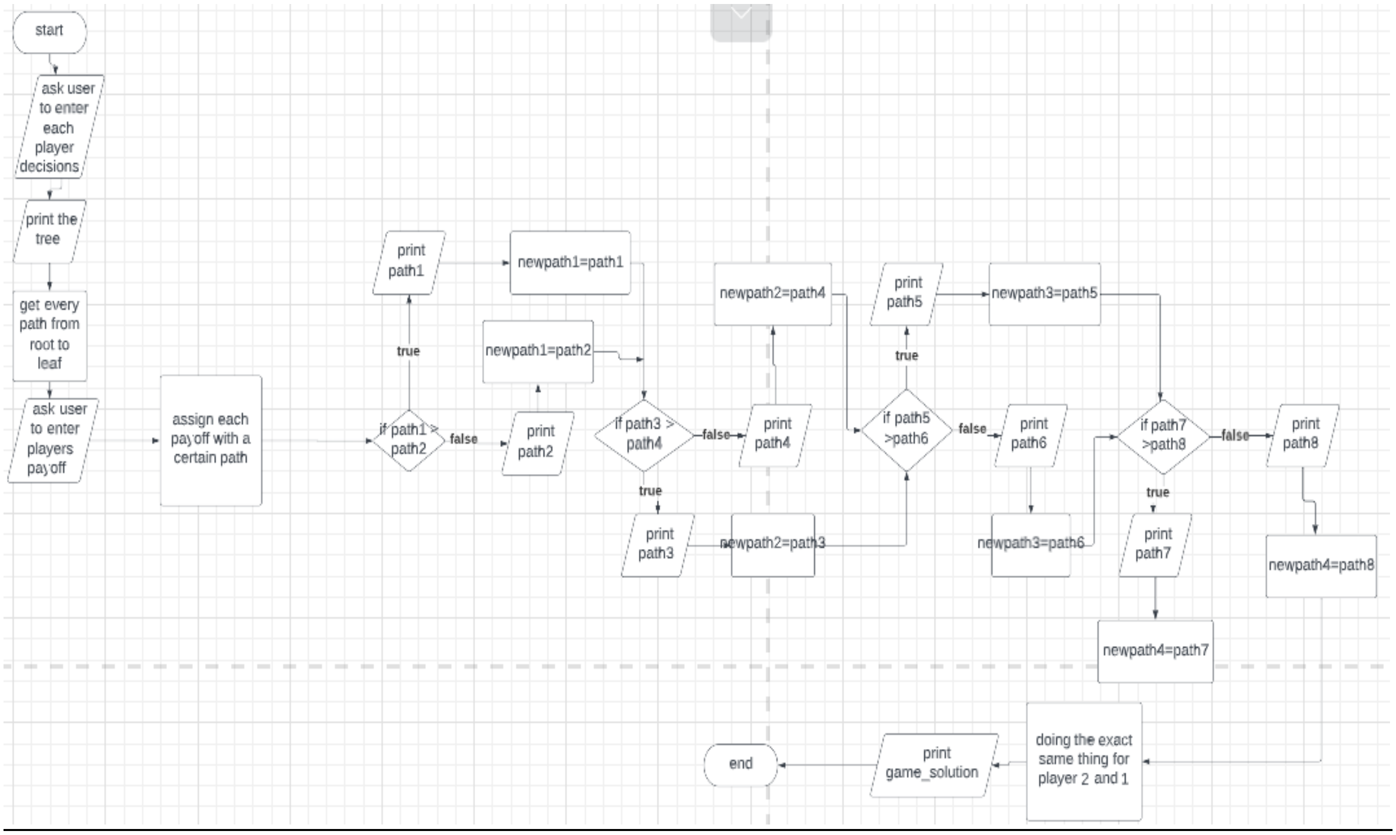
Part 3

In this part we will solve a 3-player sequential game where each player has two actions at each decision node.

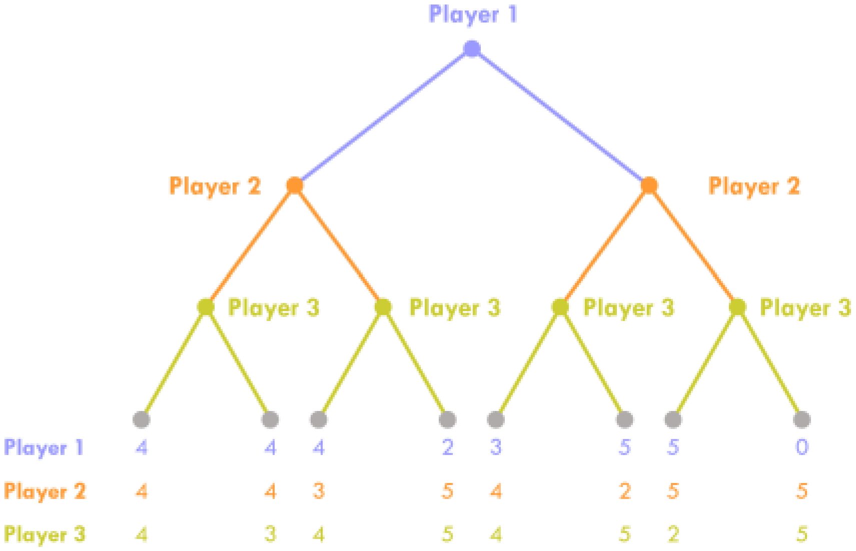
In the following steps our program will be explained:

- First, we will start by implementing a tree using treelib library.
- Ask the user to input the actions for each player.
- In get_path function we use a dictionary to link each action with its id in the tree then we get all the paths from root to leaf.
- In players_payoff function we ask the user to input each player payoff for a specific path then we append these payoffs to its path.
- In backward _induction function we start to compare player 3 payoffs to get only four paths, then we compare player 2 payoffs to get only 2 paths, then we compare player 1 payoffs to get only one path and that is our game solution.

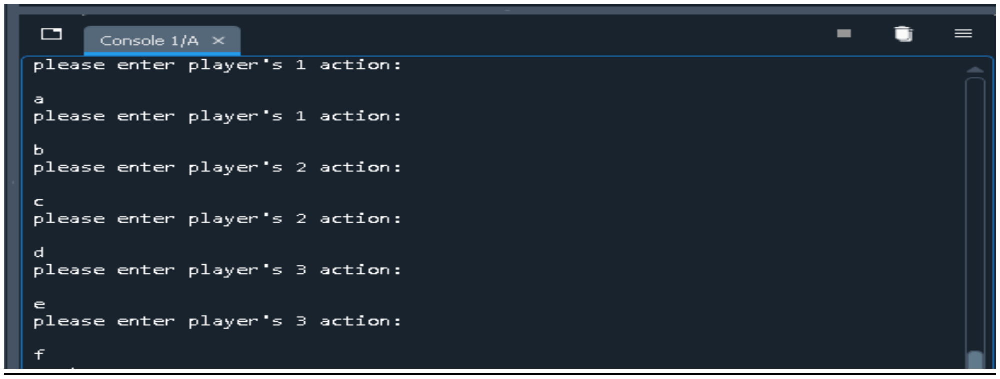
Here's a flowchart explaining the steps:



Example1:

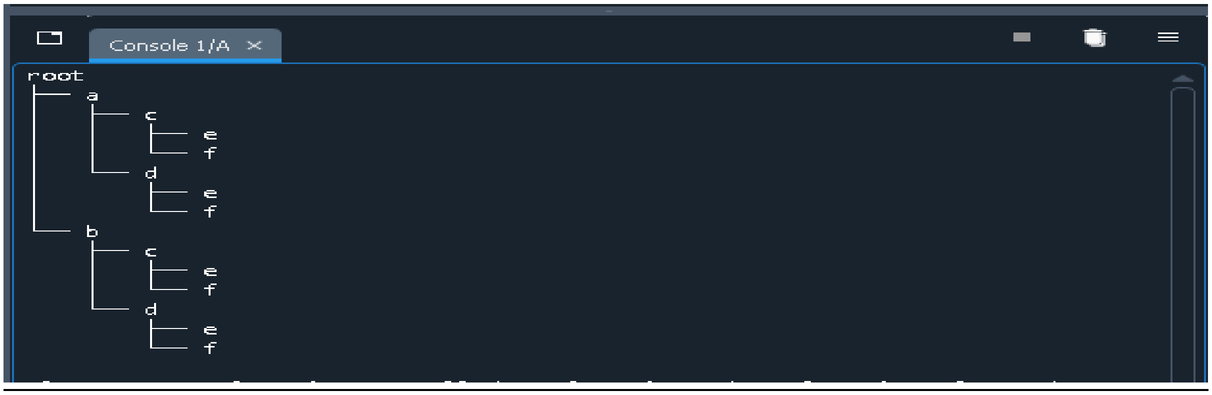


Here’s the input of each player’s decision:



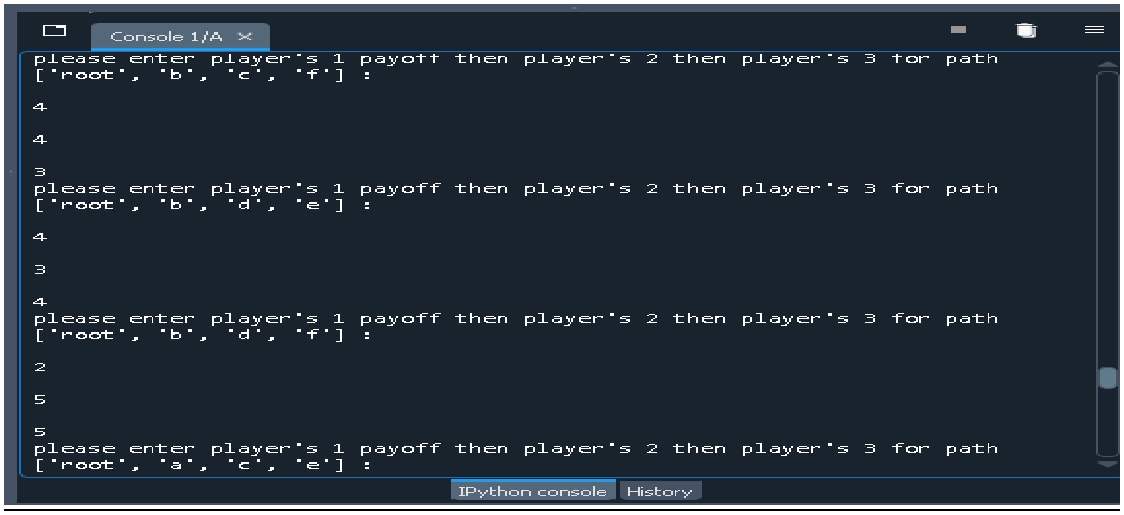
```
please enter player's 1 action:

a
please enter player's 1 action:

b
please enter player's 2 action:

c
please enter player's 2 action:

d
please enter player's 3 action:

e
please enter player's 3 action:

f
```

Here’s the tree:

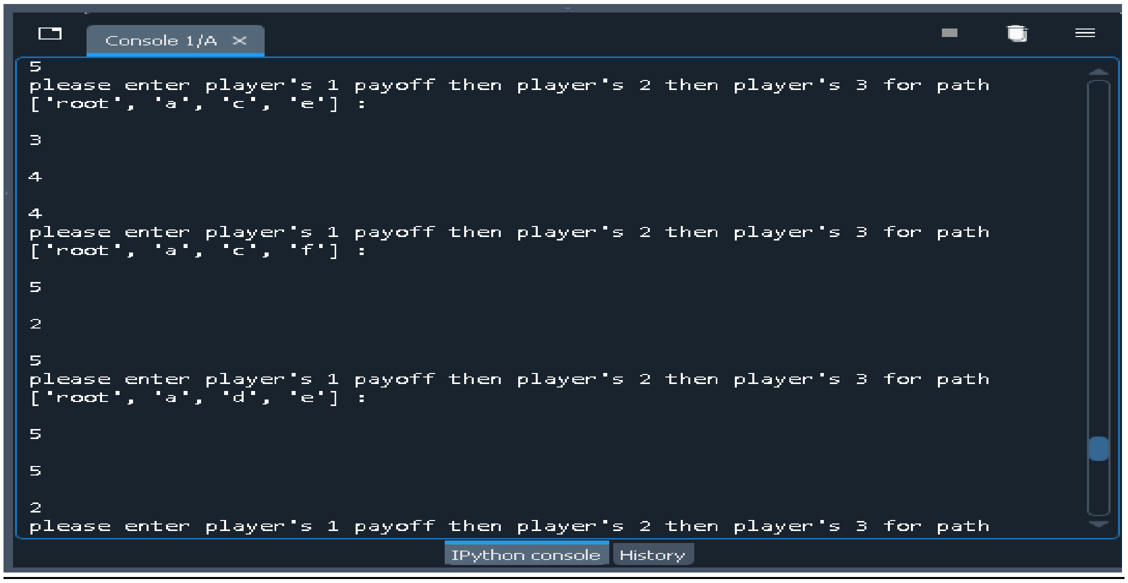


```
root
├── a
│   ├── c
│   │   ├── e
│   │   └── f
│   └── d
│       ├── e
│       └── f
└── b
    ├── c
    │   ├── e
    │   └── f
    └── d
        ├── e
        └── f
```

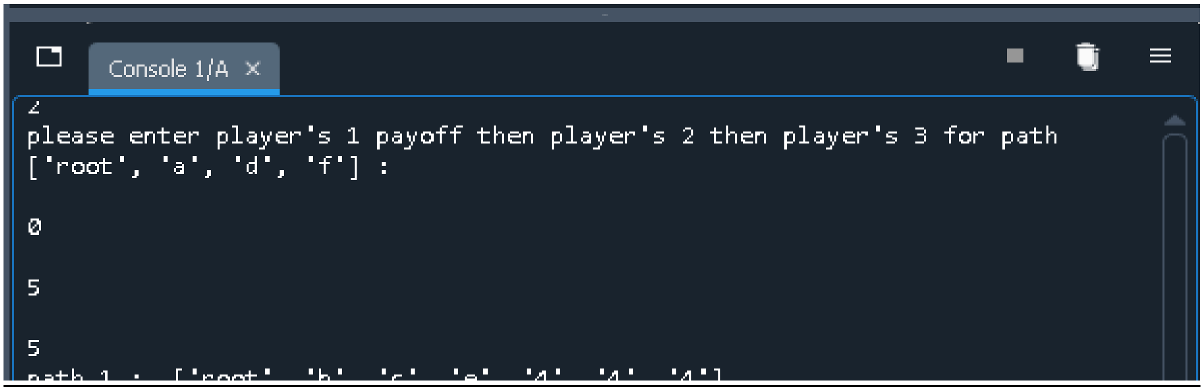
User entering each player payoff:



```
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'b', 'c', 'f'] :

4

4

3
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'b', 'd', 'e'] :

4

3

4
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'b', 'd', 'f'] :

2

5

5
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'a', 'c', 'e'] :
```

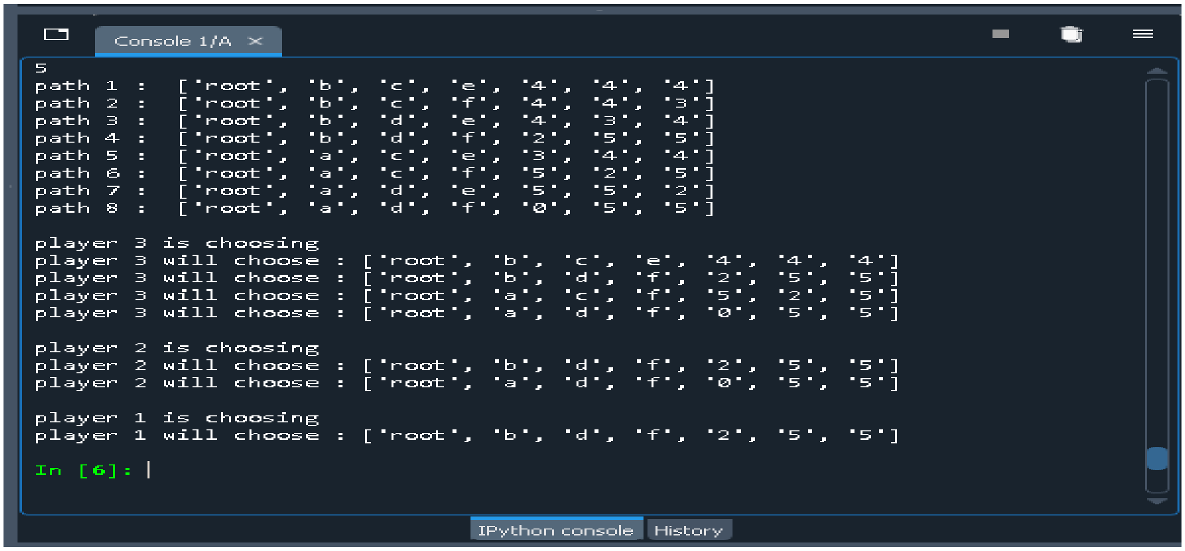


```
5
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'a', 'c', 'e'] :

3

4

4
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'a', 'c', 'f'] :

5

2

5
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'a', 'd', 'e'] :

5

5

2
please enter player's 1 payoff then player's 2 then player's 3 for path
```



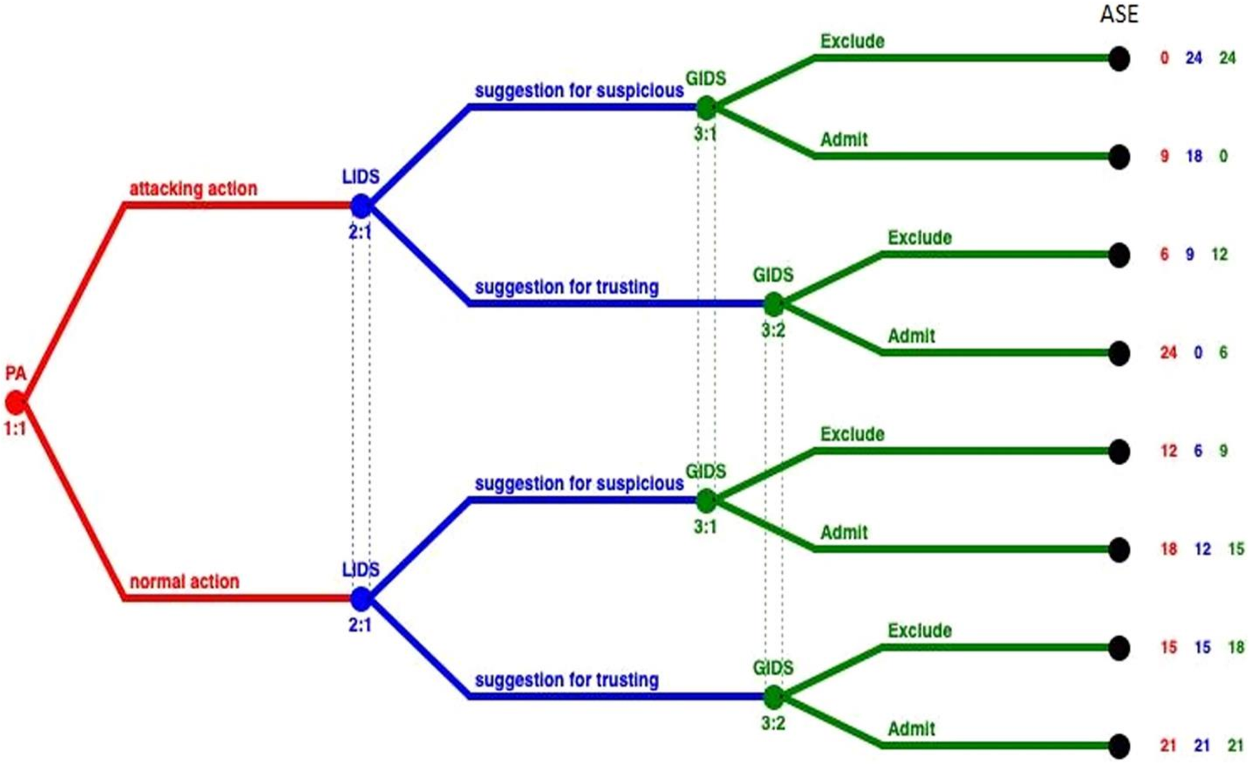
```
2
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'a', 'd', 'f'] :

0

5

5
path 1 :  ['root', 'b', 'c', 'e', '4', '4', '4']
```

Printing the game solution using backward induction:

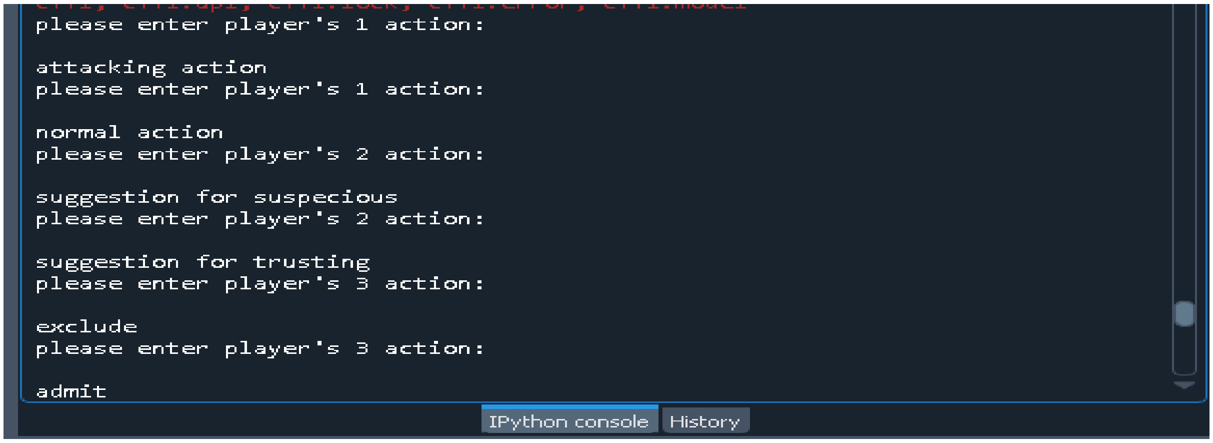


```
5
path 1 :  ['root', 'b', 'c', 'e', '4', '4', '4']
path 2 :  ['root', 'b', 'c', 'f', '4', '4', '3']
path 3 :  ['root', 'b', 'd', 'e', '4', '3', '4']
path 4 :  ['root', 'b', 'd', 'f', '2', '5', '5']
path 5 :  ['root', 'a', 'c', 'e', '3', '4', '4']
path 6 :  ['root', 'a', 'c', 'f', '5', '2', '5']
path 7 :  ['root', 'a', 'd', 'e', '5', '5', '2']
path 8 :  ['root', 'a', 'd', 'f', '0', '5', '5']

player 3 is choosing
player 3 will choose : ['root', 'b', 'c', 'e', '4', '4', '4']
player 3 will choose : ['root', 'b', 'd', 'f', '2', '5', '5']
player 3 will choose : ['root', 'a', 'c', 'f', '5', '2', '5']
player 3 will choose : ['root', 'a', 'd', 'f', '0', '5', '5']

player 2 is choosing
player 2 will choose : ['root', 'b', 'd', 'f', '2', '5', '5']
player 2 will choose : ['root', 'a', 'd', 'f', '0', '5', '5']

player 1 is choosing
player 1 will choose : ['root', 'b', 'd', 'f', '2', '5', '5']

In [6]: |
```

Example 2:

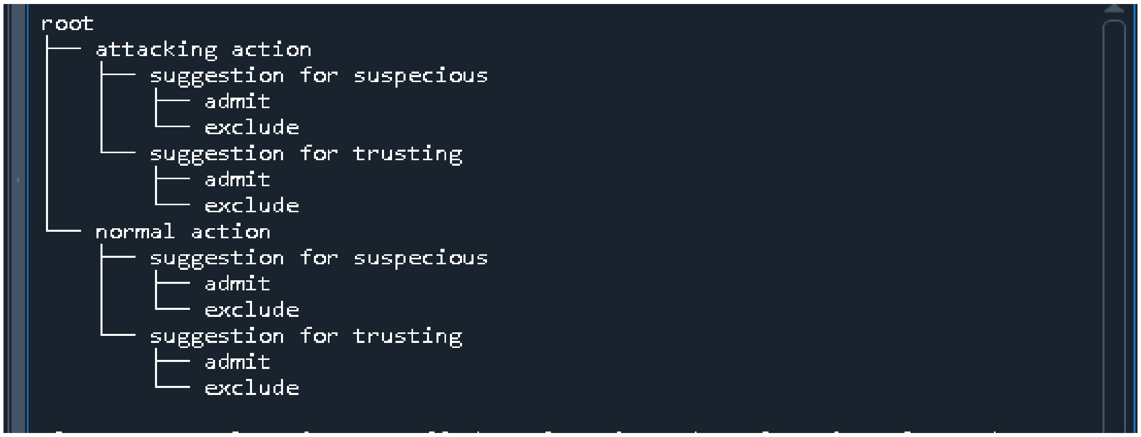


Here’s the input of each player’s decision:



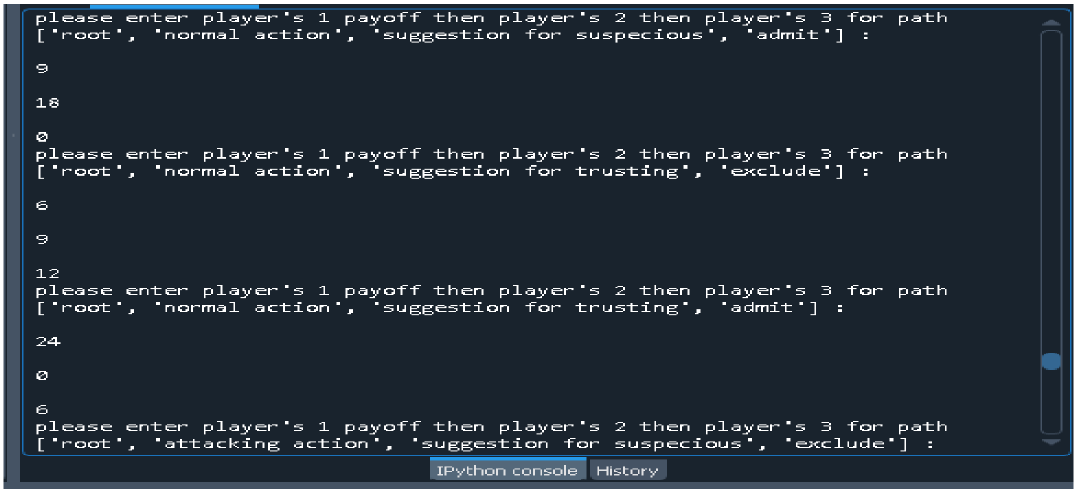
```
cffi, cffi.api, cffi.lock, cffi.error, cffi.model
please enter player's 1 action:

attacking action
please enter player's 1 action:

normal action
please enter player's 2 action:

suggestion for suspecious
please enter player's 2 action:

suggestion for trusting
please enter player's 3 action:

exclude
please enter player's 3 action:

admit
```

Here’s the tree:

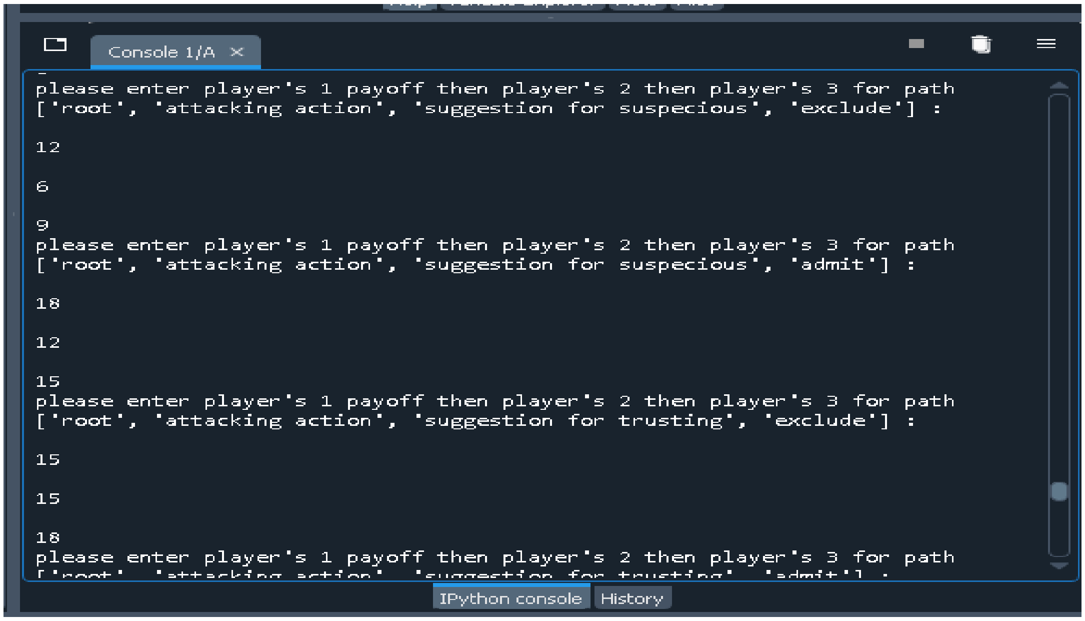


```
root
├── attacking action
│   ├── suggestion for suspecious
│   │   ├── admit
│   │   └── exclude
│   └── suggestion for trusting
│       ├── admit
│       └── exclude
└── normal action
    ├── suggestion for suspecious
    │   ├── admit
    │   └── exclude
    └── suggestion for trusting
        ├── admit
        └── exclude
```

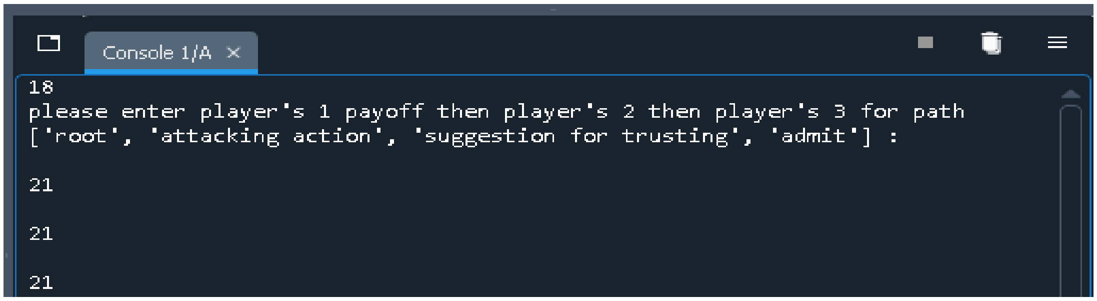
User entering each player payoff:



```
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'normal action', 'suggestion for suspecious', 'admit'] :

9

18

0
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'normal action', 'suggestion for trusting', 'exclude'] :

6

9

12
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'normal action', 'suggestion for trusting', 'admit'] :

24

0

6
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'attacking action', 'suggestion for suspecious', 'exclude'] :
```

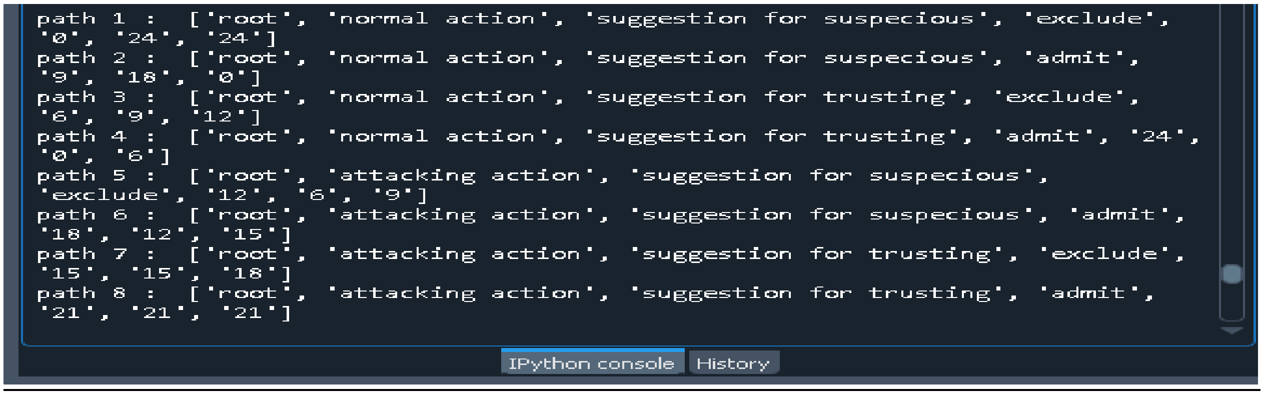


```
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'attacking action', 'suggestion for suspecious', 'exclude'] :

12

6

9
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'attacking action', 'suggestion for suspecious', 'admit'] :

18

12

15
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'attacking action', 'suggestion for trusting', 'exclude'] :

15

15

18
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'attacking action', 'suggestion for trusting', 'admit'] :
```

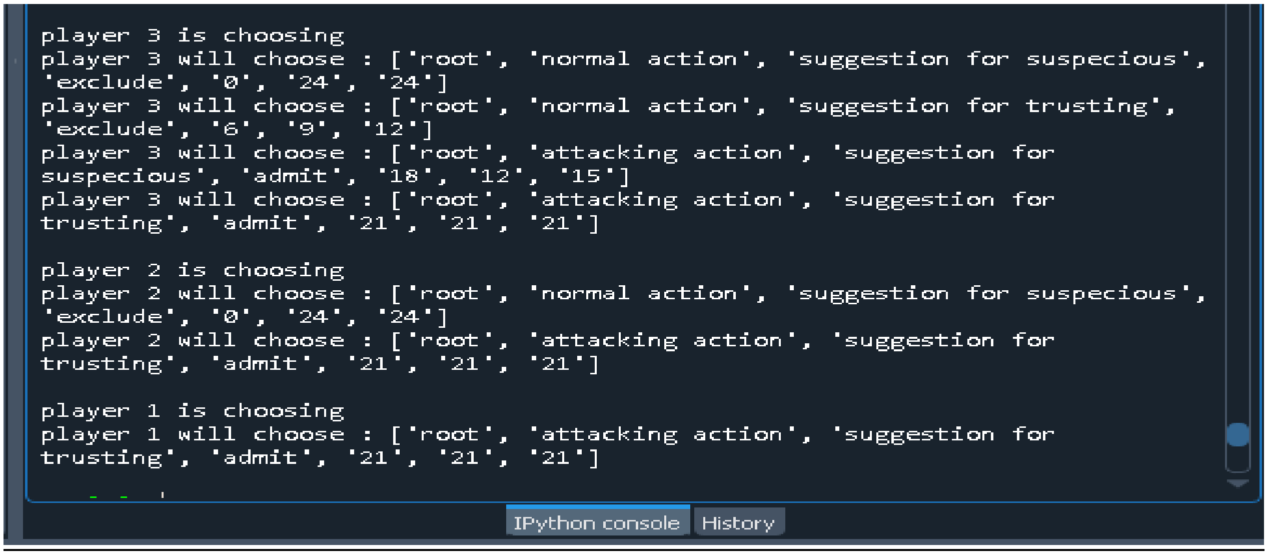


```
18
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', 'attacking action', 'suggestion for trusting', 'admit'] :

21

21

21
```

Printing the game solution using backward induction:

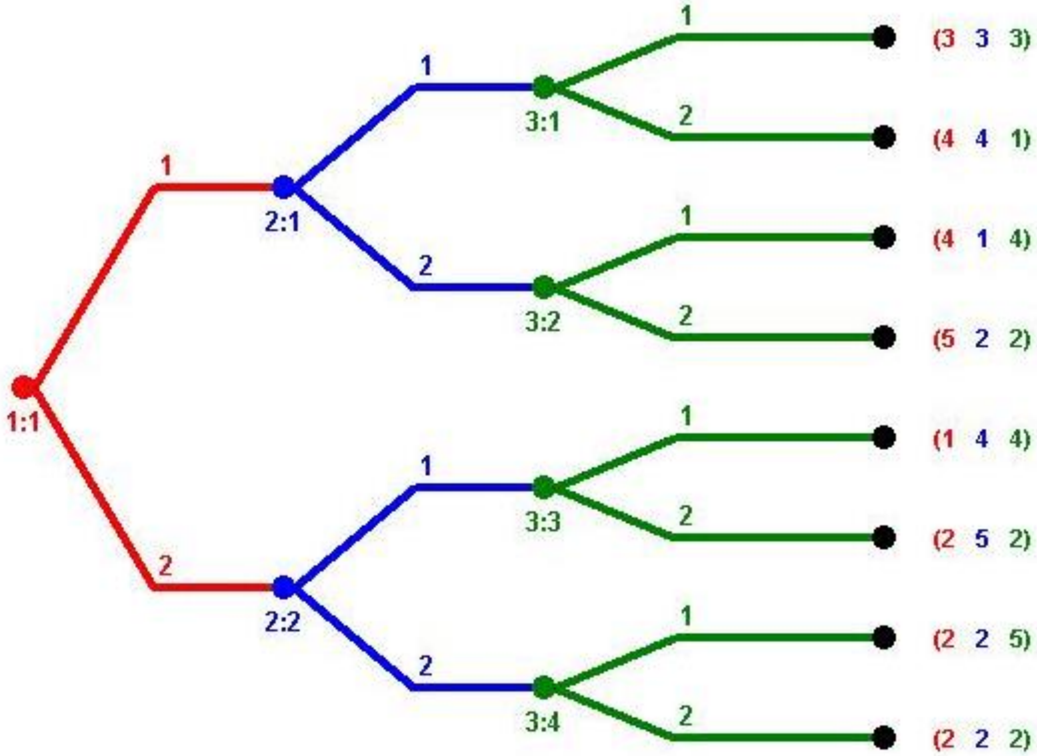


```
path 1 :  ['root', 'normal action', 'suggestion for suspecious', 'exclude',
'0', '24', '24']
path 2 :  ['root', 'normal action', 'suggestion for suspecious', 'admit',
'9', '18', '0']
path 3 :  ['root', 'normal action', 'suggestion for trusting', 'exclude',
'6', '9', '12']
path 4 :  ['root', 'normal action', 'suggestion for trusting', 'admit', '24',
'0', '6']
path 5 :  ['root', 'attacking action', 'suggestion for suspecious',
'exclude', '12', '6', '9']
path 6 :  ['root', 'attacking action', 'suggestion for suspecious', 'admit',
'18', '12', '15']
path 7 :  ['root', 'attacking action', 'suggestion for trusting', 'exclude',
'15', '15', '18']
path 8 :  ['root', 'attacking action', 'suggestion for trusting', 'admit',
'21', '21', '21']
```

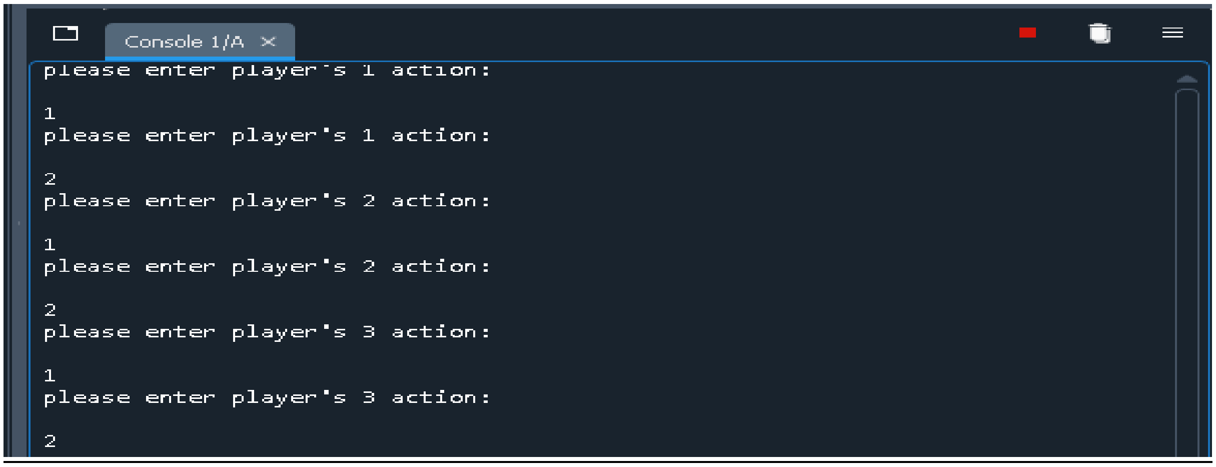


```
player 3 is choosing
player 3 will choose : ['root', 'normal action', 'suggestion for suspecious',
'exclude', '0', '24', '24']
player 3 will choose : ['root', 'normal action', 'suggestion for trusting',
'exclude', '6', '9', '12']
player 3 will choose : ['root', 'attacking action', 'suggestion for
suspecious', 'admit', '18', '12', '15']
player 3 will choose : ['root', 'attacking action', 'suggestion for
trusting', 'admit', '21', '21', '21']

player 2 is choosing
player 2 will choose : ['root', 'normal action', 'suggestion for suspecious',
'exclude', '0', '24', '24']
player 2 will choose : ['root', 'attacking action', 'suggestion for
trusting', 'admit', '21', '21', '21']

player 1 is choosing
player 1 will choose : ['root', 'attacking action', 'suggestion for
trusting', 'admit', '21', '21', '21']
```

Example 3:

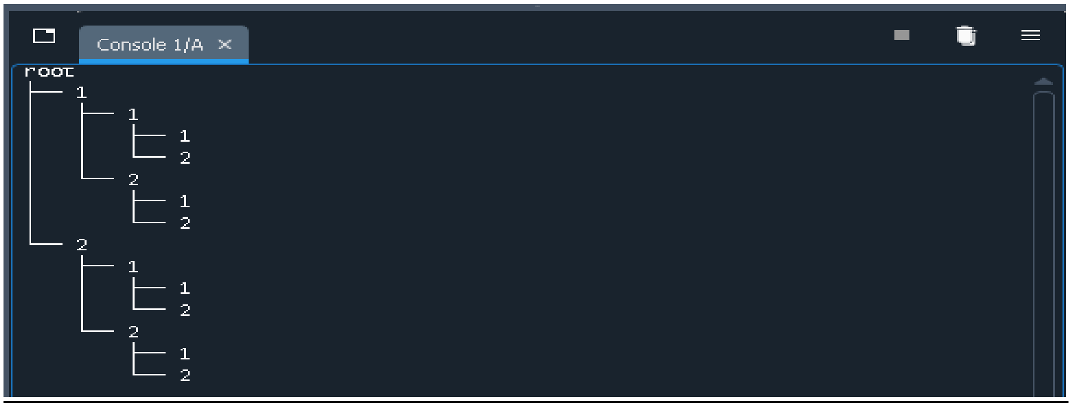


Here's the input of each player's decision:



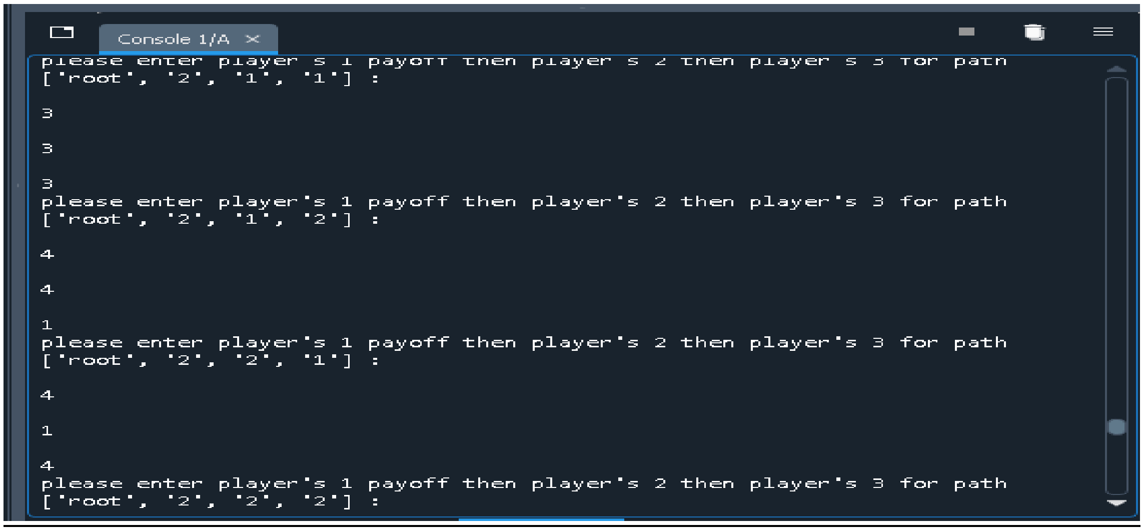
```
please enter player's 1 action:

1
please enter player's 1 action:

2
please enter player's 2 action:

1
please enter player's 2 action:

2
please enter player's 3 action:

1
please enter player's 3 action:

2
```

Here’s the tree:

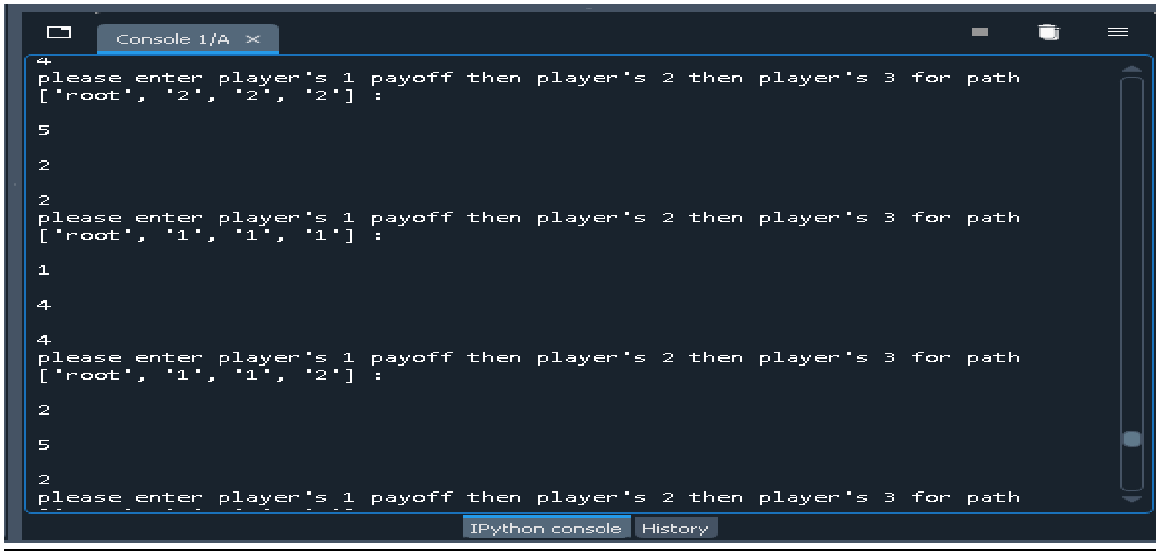


```
root
├── 1
│   ├── 1
│   │   ├── 1
│   │   └── 2
│   └── 2
│       ├── 1
│       └── 2
└── 2
    ├── 1
    │   ├── 1
    │   └── 2
    └── 2
        ├── 1
        └── 2
```

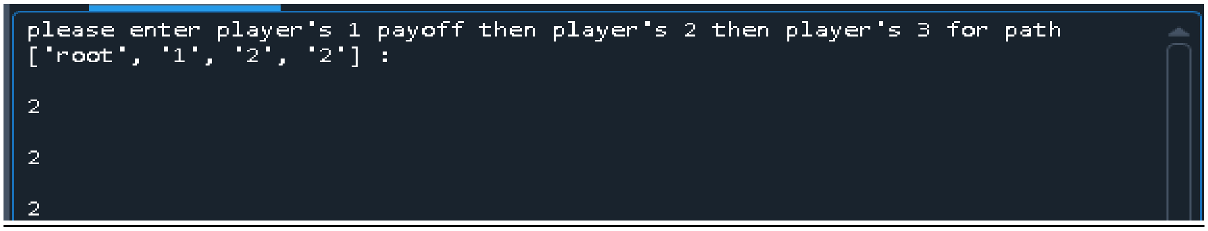
User entering each player payoff:



```
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', '2', '1', '1'] :

3

3

3
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', '2', '1', '2'] :

4

4

1
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', '2', '2', '1'] :

4

1

4
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', '2', '2', '2'] :
```

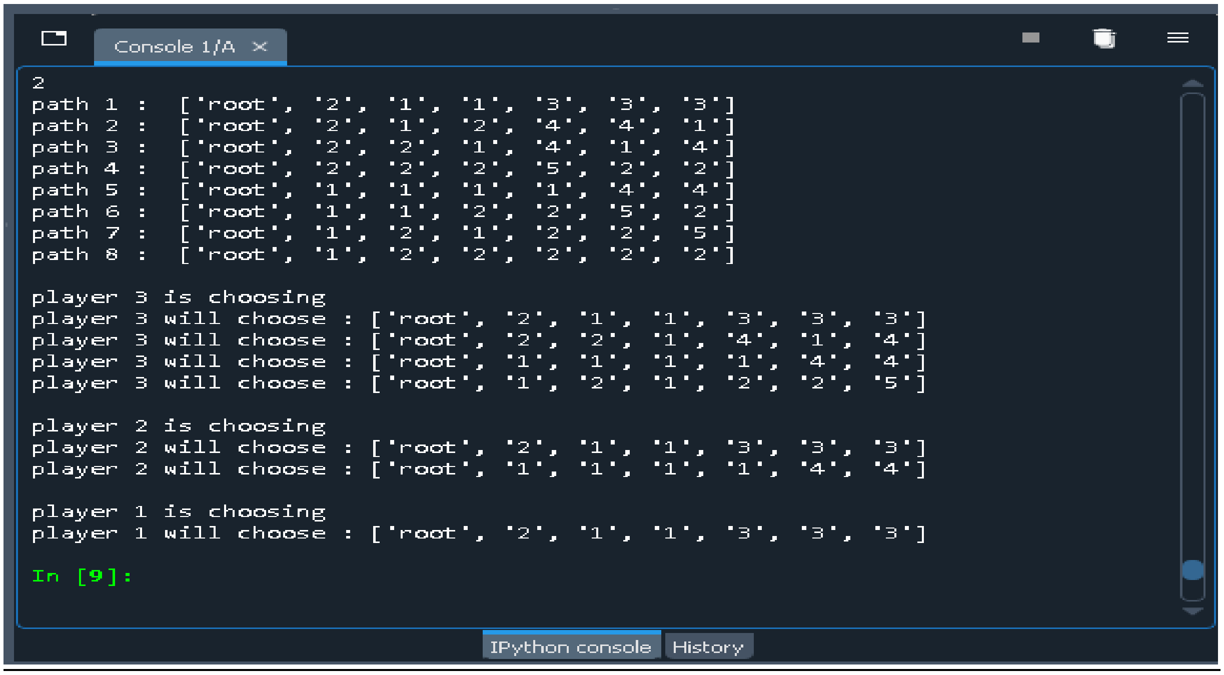


```
4
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', '2', '2', '2'] :

5

2

2
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', '1', '1', '1'] :

1

4

4
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', '1', '1', '2'] :

2

5

2
please enter player's 1 payoff then player's 2 then player's 3 for path
```



```
please enter player's 1 payoff then player's 2 then player's 3 for path
['root', '1', '2', '2'] :

2

2

2
```

Printing the game solution using backward induction:



```
2
path 1 :  ['root', '2', '1', '1', '3', '3', '3']
path 2 :  ['root', '2', '1', '2', '4', '4', '1']
path 3 :  ['root', '2', '2', '1', '4', '1', '4']
path 4 :  ['root', '2', '2', '2', '5', '2', '2']
path 5 :  ['root', '1', '1', '1', '1', '4', '4']
path 6 :  ['root', '1', '1', '2', '2', '5', '2']
path 7 :  ['root', '1', '2', '1', '2', '2', '5']
path 8 :  ['root', '1', '2', '2', '2', '2', '2']

player 3 is choosing
player 3 will choose : ['root', '2', '1', '1', '3', '3', '3']
player 3 will choose : ['root', '2', '2', '1', '4', '1', '4']
player 3 will choose : ['root', '1', '1', '1', '1', '4', '4']
player 3 will choose : ['root', '1', '2', '1', '2', '2', '5']

player 2 is choosing
player 2 will choose : ['root', '2', '1', '1', '3', '3', '3']
player 2 will choose : ['root', '1', '1', '1', '1', '4', '4']

player 1 is choosing
player 1 will choose : ['root', '2', '1', '1', '3', '3', '3']

In [9]:
```