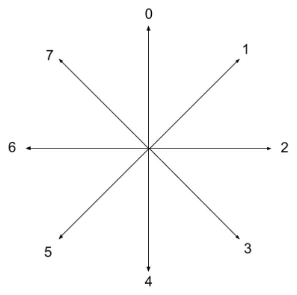
Etude 10: Coroutine Puzzles

Jono Sue (4097307), Stefan Pedersen (1427681)

Structures and Variables

An array of '**moves'** where the indices correspond to the 8 possible directions. Each index contains the x and y values to be incremented/decremented depending on the direction selected.



```
moves = [{'x': 0, 'y': -1},
         {'x': 1, 'y': -1},
         {'x': 1, 'y': 0},
         {'x': 1, 'y': 1},
         {'x': 0, 'y': 1},
         {'x': -1, 'y': 1},
         {'x': -1, 'y': 0},
         {'x': -1, 'y' -1}]
```

```
BOARD =   [[[0,4,5],[1,2,4,5],[0,1,4,5]],
           [[1,4,5,7],[ ],[1,3,6]],
           [[0,6,7],[1,5,7],[0,2,7]]]
```

A **BOARD**, considered as a 2D array containing the directions present on tile. The centre tile (1,1) representing the win, is empty.

TokenA is given the initial value of 0,0 on the BOARD, while **TokenB** is given the initial value of 2,2. These values represent current position of the token. In order to move, the current Token uses the other Token's co-ordinates to access the BOARD and look up the possible directions it may move.

A **path** list will be used to keep track of **moveSignature**s, each containing three variables: Token start position, Token end position, and the Opposite Token's position. These three variables represent a signature which if seen again will indicate that the exact scenario has been experienced and will result in a loop unless broken out of.

path = [[tokenStart, tokenEnd, otherToken]…]

A **baseline** is initially set to the maximum number of possible moves for the given board.

A **counter** will be used to keep track of the number of moves taken. Once a solution has been found, if the counter is less than the **baseline** the baseline is assigned the value from the counter. If the counter ever exceeds the baseline on a further search, that search can be terminated as a quicker solution has already been found.

A Boolean **turn** flag to indicate which Token is currently moving. It is initially set to True, indicating token A.

Description of move(A, B, turn, counter, path) Algorithm

1) The turn flag is used to assign **token** and **otherToken.**
2) Move counter is incremented.
3) Check if token is on 1,1 which would indicate a WIN. If a win AND **counter** is less than **baseline**, assign the value of counter to **baseline.**
4) If **counter** exceeds the **baseline**, terminate search on this branch.
5) Potential directions are retrieved from **otherToken**'s coordinates.
6) For each of these directions, a copy of **token**'s x and y coordinates are adjusted by accessing the **moves** array an applying the relevant changes. Each of these potential moves is then tested for validity. A valid move must be on the board ((x or y) >= 0, (x or y) <= 2), and cannot move onto an occupied tile ((x and y) != (x and y of **otherToken**)). If a potential move passes these tests, it is added to a list of **validDirections**.
7) For each valid direction a **moveSignature** is created, which consists of the **token**'s current position, the position the **token** will end up in after a valid move, and the position of the **otherToken**. The **moveSignature** is checked against the existing move signatures in the **path** list. If it exists ignore this move, and try the next valid direction. If the signature doesn't exist, append the **moveSignature** to the **path** list.
8) Call **move()** again by passing in **token**'s new position, and the flipped Boolean **turn** value.

Etude 10: Coroutine Puzzles

Jono Sue (4097307), Stefan Pedersen (1427681)

Solutions to two puzzles

1	2	3
4	5	6
7	8	9

For the sake of human-readability, we have represented the tiles as squares 1-9 rather than as the coordinate values used in the algorithm.

The following labels apply to the images below. **p** is the currently moving player. **p Start** and **p End** are the before and after positions of the player's move. **p' Pos** is the position of the other token, while **p' Directions** lists the directions corresponding to that tile. **Dir Used** indicates the direction taken for the current move. **Valid Remain** are the remaining valid moves possible.
Stepping through each turn, we record the direction used, and the valid directions remaining. If the move signature matches one already seen (indicated by same colour highlighting) we know to stop and return the last branch (with a valid remaining move). **Sig seen** indicates that the signature has been seen and the algorithm backtracks to explore the next valid branch.

PUZZLE 2

Turn	p	p Start	p End	p' Pos	p' Directions	Dir Used	Valid Remain
1	A	1	2	9	0,2,7	2	~
2	B	9	9	2	1,2,4,5	none	~
3	A	2	3	9	0,2,7	2	~
4	B	9	6	3	0,1,4,5	0	~
5	A	3	2	6	1,3,6	6	~
6	B	6	9	2	1,2,4,5	4	5
7	A	2	3	9	0,2,7	2	Sig seen
6	B	6	8	2	1,2,4,5	5	~
7	A	2	4	8	1,5,7	5	~
8	B	8	6	4	1,4,5,7	1	~
9	A	4	2	6	1,3,6	1	3
10	B	6	9	2	1,2,4,5	4	Sig seen
9	A	4	8	6	1,3,6	3	~
10	B	6	2	8	1,5,7	7	~
11	A	8	6	2	1,2,4,5	1	2
12	B	2	1	6	1,3,6	6	~
13	A	6	3	1	0,4,5	0	4,5
14	B	1	4	3	0,1,4,5	4	~
15	A	3	6	4	1,4,5,7	4	5
16	B	4	2	6	1,3,6	1	Sig seen
15	A	3	5	4	1,4,5,7	5	WIN
13	A	6	9	1	0,4,5	4	5
14	B	1	2	9	0,2,7	2	Sig seen
13	A	6	8	1	0,4,5	5	~
14	B	1	1	8	1,5,7	none	~
15	A	8	5	1	0,4,5	0	WIN
11	A	8	9	2	1,2,4,5	2	~
12	B	2	3	9	0,2,7	2	Sig seen

PUZZLE 3

Turn	p	p Start	p End	p' Pos	p' Directions	Dir Used	Valid Remain
1	A	1	2	9	2,7	2	~
2	B	9	9	2	3,4,5	none	
3	A	2	3	9	2,7	2	~
4	B	9	8	3	3,6	6	~
5	A	3	3	8	0,3	none	
6	B	8	7	3	3,6	6	
7	A	3	6	7	0,1,4	4	~
8	B	7	7	6	4,7	none	
9	A	6	3	7	0,1,4	0	4
10	B	7	7	3	3,6	none	
11	A	3	6	7	0,1,4		Sig seen
9	A	6	9	7	0,1,4	4	~
10	B	7	8	9	2,7	2	~
11	A	9	6	8	0,3	0	~
12	B	8	4	6	4,7	7	~
13	A	6	3	4	0,5	0	5
14	B	4	8	3	3,6	3	~
15	A	3	3	8	0,3	none	Sig seen
13	A	6	8	4	0,5	5	~
14	B	4	1	8	0,3	0	~
15	A	8	7	1	5,6	6	~
16	B	1	4	7	0,1,4	4	~
17	A	7	7	4	0,5	none	
18	B	4	1	7	0,1,4	0	1
19	A	7	7	1	5,6	none	
20	B	1	4	7	0,1,4	4	Sig seen
18	B	4	2	7	0,1,4	1	~
19	A	7	7	2	3,4,5	none	
20	B	2	5	7	0,1,4	4	WIN

Generating Coroutine Puzzles

For each of the 8 squares (excluding the centre winning square) in the grid that can have directions, generate 1-8 directions randomly. So for a particular square, each potential direction is randomly chosen or not chosen to be displayed as a direction. As a result, a complete grid will be generated with random directions.

Our algorithm can then be run on the generated grid in order to see if there exists a winning set of moves. If a successful set of moves is achieved, this grid can be used as a solvable coroutine puzzle.

The counter in the algorithm can be used to record the minimum number of moves to solve the puzzle, as well as being able to find a puzzle with that can be solved in a user specified number of moves.