Pratical 1

Write a program to implement Water Jug Problem using `Breadth First Search` algorithm.

[1]:
```
from collections import deque

def BFS(a, b, target):
        m = {}
        isSolvable = False
        path = []
        q = deque()
        q.append((0, 0))

        while (len(q) > 0):
                u = q.popleft()# If this state is already visited
                if ((u[0], u[1]) in m):
                        continue
                if ((u[0] > a or u[1] > b or
                        u[0] < 0 or u[1] < 0)):
                        continue
                path.append([u[0], u[1]])
                m[(u[0], u[1])] = 1

                if (u[0] == target or u[1] == target):
                        isSolvable = True
                        if (u[0] == target):
                                if (u[1] != 0):
                                        path.append([u[0], 0])
                        else:
                                if (u[0] != 0):
                                        path.append([0, u[1]])
                        sz = len(path)
                        for i in range(sz):
                                print("(", path[i][0], ",",
                                        path[i][1], ")")
                        break
                q.append([u[0], b]) # Fill Jug2
                q.append([a, u[1]]) # Fill Jug1
```

```
                for ap in range(max(a, b) + 1):
                        # Pour amount ap from Jug2 to Jug1
                        c = u[0] + ap
                        d = u[1] - ap
                        if (c == a or (d == 0 and d >= 0)):
                                q.append([c, d])
                        # Pour amount ap from Jug 1 to Jug2
                        c = u[0] - ap
                        d = u[1] + ap
                        # Check if this state is possible or not
                        if ((c == 0 and c >= 0) or d == b):
                                q.append([c, d])
        # No, solution exists if ans=0
        if (not isSolvable):
                print("No solution")

Jug1, Jug2, target = 5, 3, 2
print("Path from initial state to solution state ::")
BFS(Jug1, Jug2, target)
```

```
Path from initial state to solution state ::
( 0 , 0 )
( 0 , 3 )
( 5 , 0 )
( 5 , 3 )
( 3 , 0 )
( 2 , 3 )
( 2 , 0 )
```