In [1]:
```
# Solve(position) → { win, tie, lose, draw }
def Solve(position):
    pos = PrimitiveValue(position)
    if pos == 'not_primitive':
        children = []
        for move in GenerateMoves(position):
            children.append(Solve(DoMove(position, move)))
        if "lose" in children:
            win_s.add(position)
            return "win"
        elif "tie" in children:
            tie_s.add(position)
            return "tie"
        else:
            lose_s.add(position)
            return "lose"
    elif pos == "lose":
        lose_primitive_s.add(position)
        return "lose"
    elif pos == "tie":
        tie_primitive_s.add(position)
        return "tie"
    else:
        print("There is an error!")
        return
```

In [2]:
```
# Variables for analysis
lose_s = set()
win_s = set()
tie_s = set()
lose_primitive_s = set()
win_primitive_s = set()
tie_primitive_s = set()
```

In [ ]:

In [8]:
```
%run Solver.ipynb
```

In [9]:
```
# Position format: "xxoo-----"
GAMENAME = 'TIC_TAC_TOE'
STARTING_POS = "---------"
```

In [10]:
```
def DoMove(position, move):
    if position.count('x') > position.count('o'):
        position = list(position)
        position[move] = 'o'
        position = "".join(position)
    else:
        position = list(position)
        position[move] = 'x'
        position = "".join(position)
    return position
```

In [11]:
```
def GenerateMoves(position):
    return [i for i, char in enumerate(position) if char == '-']
```

In [12]:
```
def triple_equal(string, a, b, c):
    return string[a] != '-' and \
string[a] == string[b] and \
string[b] == string[c]

def haspattern(pos):
    return triple_equal(pos,0,1,2) or \
triple_equal(pos,3,4,5) or triple_equal(pos,6,7,8) \
or triple_equal(pos,0,3,6) or triple_equal(pos,1,4,7) or \
triple_equal(pos,2,5,8) or \
triple_equal(pos,0,4,8) or triple_equal(pos,2,4,6)

def modify_str(string, index, char):
    new = list(string)
    new[index] = char
```

In [13]:
```
def PrimitiveValue(position):
    if haspattern(position):
        return "lose"
    elif '-' not in position:
        return "tie"
    else:
        return "not_primitive"
```

In [16]:
```
%%capture cap

result = Solve(STARTING_POS)

print("result is a " + result + '!')

losses = len(lose_s)
wins = len(win_s)
ties = len(tie_s)
pr_losses = len(lose_primitive_s)
pr_wins = len(win_primitive_s)
pr_ties = len(tie_primitive_s)

total_losses = losses + pr_losses
total_wins = wins + pr_wins
total_ties = ties + pr_ties
total = total_losses + total_wins + total_ties
pr_total = pr_losses + pr_wins + pr_ties

print("Lose: ", total_losses, ' (', pr_losses, ' primitive)')
print("Win: ", total_wins, ' (', pr_wins, ' primitive)')
print("Tie: ", total_ties, ' (', pr_ties, ' primitive)')
print("Total: ", total, ' (', pr_total, ' primitive)')

with open(GAMENAME + '_output.txt', 'w') as f:
    f.write(cap.stdout)
```

In [ ]:
```
```

```
result is a tie!
Lose:  1574  ( 942  primitive)
Win:  2836  ( 0  primitive)
Tie:  1068  ( 16  primitive)
Total:  5478  ( 958  primitive)
```