Gavin Browning

8/31/2018

CS 5050 Advanced Algorithms

Assignment One: DP NIM Multiple Piles

```
# Trade time for space using memo-ization.
look_up_table = {(0, 0, 0): True}

# Returns whether you win or not if there is one pile left.
def one_piles(a, b, c):
    if a + b + c = 1:
        look_up_table[(a, b, c)] = False
        return False
    look_up_table[(a, b, c)] = True
    return True

# Returns whether you win or not if there are two piles left.
def two_piles(a, b, c):
    piles = [a, b, c].remove(0)
    result = piles[0] != piles[1]
    look_up_table[(a, b, c)] = result
    return result

# If there are three piles left and two of them are the same return True otherwise False.
def three_piles(a, b, c):
    if a == b or a == c or b == c:
        look_up_table[(a, b, c)] = True
        return True
    return False
```

```
# Recursive function that determines whether you win the game of NIM from the current state.
def win(a, b, c):
    if (a, b, c) in look_up_table:
        return look_up_table[(a, b, c)]
    if a == b == c == 0:  # zero piles
        return True
    if a * b == 0 or a * c == 0 or b * c == 0:  # one pile
        return one_piles(a, b, c)
    if a * b * c == 0:  # two piles
        return two_piles(a, b, c)
    if three_piles(a, b, c):  # three piles
        return True
    for x in range(a):  # depth search for possible win from removing stones from pile a
        if not win(a-x-1, b, c):
            look_up_table[(a, b, c)] = True
            return True
    for x in range(b):  # depth search for possible win from removing stones from pile b
        if not win(a, b-x-1, c):
            look_up_table[(a, b, c)] = True
            return True
    for x in range(c):  # depth search for possible win from removing stones from pile c
        if not win(a, b, c-x-1):
            look_up_table[(a, b, c)] = True
            return True
    return False  # could not find a win
```