$$T(n) = T(n-3) + T(n-1) + T(n-2) + 3 + 3 + 2$$
$$= T(n-3) + T(n-1) + T(n-2) + 8$$
$$= 3^k \cdot T(n-k) + (3^k - 1) \cdot C$$
$$= 3^n \cdot T(n-n) + (3^n - 1) \cdot C$$
$$= 3^n \cdot T(0) \cdot (1 + c) - c$$

let:
$n-k=0$
$k=n$

```
def tribonacci_recursive(n):
  if n == 0 or n == 1:
    return 0
  elif n == 2:
    return 1
  else:
    return tribonacci_recursive(n - 3) + tribonacci_recursive(n - 1) + tribonacci_recursive(n - 2)
```

3 steps

$\boxed{T(n) = 3^n}$

- 3 comparisons, 3 subtractions, 2 additions

```
def tribonacci_iterative(n):
  if n == 0 or n == 1:
    return 0
  elif n == 2:
    return 1
  n3 = 0
  n2 = 1
  n1 = 1
  trib = n3 + n2 + n1
  for i in range(3, n - 1):
    n3 = n2
    n2 = n1
    n1 = trib
    trib = n3 + n2 + n1
  return trib
```

3 steps

n times

$T(n) = n + 3$

Proof by properties of Θ:

$$T(n) = n + 3$$
$$= \Theta(n+3) \quad \text{(trivial)}$$
$$= \Theta(n) \quad \text{(constant factor)}$$

$\boxed{T(n) = n + 3 \in \Theta(n)}$

```
def tribonacci_memoized(n):
  C = [None] * max(n + 1, 3)
  C[0] = 0
  C[1] = 0
  C[2] = 1
  trib = tribonacci_memoized_helper(n, C)
  return trib

def tribonacci_memoized_helper(n, C):
  if C[n] is not None:
    return C[n]
  else:
    result = tribonacci_memoized_helper(n-3, C) + tribonacci_memoized_helper(n-2,
        C)+tribonacci_memoized_helper(n-1, C)
  C[n] = result
  return result
```

$\boxed{T(n) = \Theta(n)}$

n times

```
def tribonacci_dynamic(n):
  A = [None] * max(n + 1, 3)
  A[0] = 0
  A[1] = 0
  A[2] = 1
  for i in range(3, n + 1):
    A[i] = A[i - 3] + A[i - 2] + A[i - 1]
  return A[n]
```

$\boxed{T(n) = \Theta(n)}$

n times

- We are using the for loop to go through the arrays n times.