**Pseudocode for** non-recursive **code** :

Algorithm birthdayCakeCandles( ) { Time complexity

1. Declare numCandles 1
2. Input( numCandles ) 1
3. Initialize arr[ numCandles ] 1
4. For i = 0 to numCandles -1 do n+1

Input( arr[ i ] ) n

1. result ← findMaxHeight ( arr , numCandles ) T( findMaxHeight )
2. Print ( result ) 1

}

T(n) => T( findMaxHeight )+ n

findMaxHeight ( arr , numCandles ){

1. max ← 0 , number\_of\_occurs ← 0 1
2. for i = 0 to numCandles -1 do n+1

if arr[ i ] > max n

max ← arr[ i ] n

number\_of\_occurs ← 1 n

else if arr[ i ] == max n

number\_of\_occurs +=1 n

1. return number\_of\_occurs 1

}

T(n) => max(n,1) => n

The ***Time*** complexity of the whole code is n + n => Θ(n)

The ***Space*** complexity of the whole code is Θ(numCandles)

**Pseudocode for** recursive **code** :

Algorithm birthdayCakeCandles( ) { Time complexity

1. Declare numCandles 1
2. Input( numCandles ) 1
3. Initialize arr[ numCandles ] 1
4. For i = 0 to numCandles -1 do n+1

Input( arr[ i ] ) n

1. result ← findMaxHeight1 ( arr , numCandles ) T( findMaxHeight1 )
2. Print ( result ) 1

}

T(n) => T( findMaxHeight1 )+ n

findMaxHeight1( arr , numCandles ){

1. Return findMaxHeight( arr + 1 , numCandles -1 , arr[ 0 ] , 1 )

}

T(n) => T( findMaxHeight )

findMaxHeight( arr , numCandles , max , number\_of\_occurs ){

1. If numCandles == 0 Then 1

Return number\_of\_occurs 1

1. If arr[ 0 ] > max Then

return findMaxHeight (arr + 1, n - 1, arr[ 0 ] , 1) T(n-1)

else if arr[ 0 ] == max Then

return findMaxHeight ( arr + 1 , n – 1 , max , number\_of\_occurs + 1) T(n-1)

else Then

return findMaxHeight ( arr + 1, n - 1, max , number\_of\_occurs ) T(n-1)

}

T(n) => T(n-1) + 1 => n

The ***Time*** complexity of the whole code is n + n => Θ(n)

The ***Space*** complexity of the whole code is Θ(numCandles)

**Comparison between Non-Recursive and Recursive**

* Both have the same time complexity and space complexity .
* Choosing the recursive one may occur a STACK OVERFLOW for large input which isn’t good practice , so we tend to use non recursive one which is simpler and avoiding the STACK OVERFLOW problem.