## PSEUDO-CODE

### a) Non-Recursive(Iterative) Algorithm Pseudo-code

```
Function TallestCandles(candles[], n):
    max_height ← candles[0]
    count ← 1
    For i from 1 to n-1:
        If candles[i] > max_height:
            max_height ← candles[i]
            count ← 1
        Else if candles[i] == max_height:
            count ← count + 1
    Return count
```

### b)Recursive Algorithm Pseudo-code

```
Function TallestCandlesRecursive(candles[], index, max_height, count):
    If index == length of candles:
        Return count
    If candles[index] > max_height:
        max_height ← candles[index]
        count ← 1
    Else if candles[index] == max_height:
        count ← count + 1
    Return TallestCandlesRecursive(candles, index + 1, max_height, count)
```

## TIME COMPLEXITY ANALYSIS

### a) Non-Recursive Algorithm

Time Complexity Analysis

Time Complexity: O(n)

Space Complexity: O(1)

### b)Recursive Algorithm Pseudo-code

Time Complexity Analysis

Time Complexity: O(n)

Space Complexity: O(n) due to recursive call stack

## COMPARISON SUMMARY

| Metric | Iterative | Recursive |
| --- | --- | --- |
| • Time Complexity | • O(n) | • O(n) |
| • Space Complexity | • O(1) | • O(n) (due to recursion) |
| • Speed | • Faster (no call overhead) | • Slightly slower |
| • Readability | • High | • Medium (if recursion is clear) |