**Time Complexity**

#include "RecursiveCandels.h"

MaxHeightCount recursive\_candles(int \*candles, int n, int current\_height, int max\_height) { -----------------🡪 O(1)

if (n == 0) {-----------------🡪 O(1)

MaxHeightCount result = {.height = 0, .count = 0};-----------------🡪 O(1)

return result;

}

MaxHeightCount result = {.height = current\_height, .count = 0 -----------------🡪 O(1)

};

int count = 0;

if (candles[0] == current\_height) { -----------------🡪 O(1)

count++;

}

int \*rest\_of\_candles = candles + 1;

MaxHeightCount recursive\_result = recursive\_candles(rest\_of\_candles, n - 1, current\_height, max\_height); -----------------🡪 O(1)

if (count + recursive\_result.count > result.count) { -----------------🡪 O(1)

result.count = count + recursive\_result.count;

}

if (recursive\_result.height > result.height) { -----------------🡪 O(1)

result.height = recursive\_result.height;

result.count = recursive\_result.count;

}

return result; -----------------🡪 O(1)

}

int birthdayCandles(int \*candles, int n) {

MaxHeightCount max\_height\_count = recursive\_candles(candles, n, candles[0], 0) ; -----------------🡪 O(n)

printf("The Tallest Candle Is: %d\n", max\_height\_count.height);

printf("Number of Tallest Candles: %d\n", max\_height\_count.count);

return max\_height\_count.count; -----------------🡪 O(1)

}

**T(n) =** *O*(1)+*O*(1)+*O*(1)+*O*(1)+*O*(1)+*O*(1)+*O*(1)+*O*(1)+*O*(*n*)+*O*(1) = O(n)