Question 1

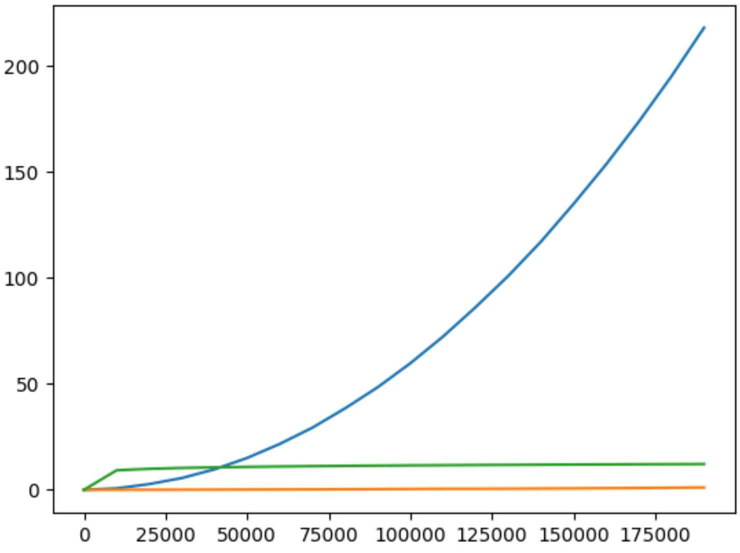
B. Iterative is θ(n) as it accumulates the result n times, while the recursive's the time complexity is

$$T(n) = \begin{cases} T(\frac{n}{2}) + \Theta(1) & \text{if } n > 1 \\ \Theta(1) & \text{if } n = 1 \end{cases}$$

A = 1
B = 2
Cost of nodes = $\frac{n}{2^i}$
Cost of leaves = $n^{log_b a} = n^{log_2 1} = n^0 = 1$
$h = log_2 n$

Total cost = $1 + \sum_{i=0}^{h-1} \frac{n}{2^i}$ = $log_2 n$

C.



Blue line: iteration (I'm not sure why its growing)
Orange line: recursion
Green line: log(n)

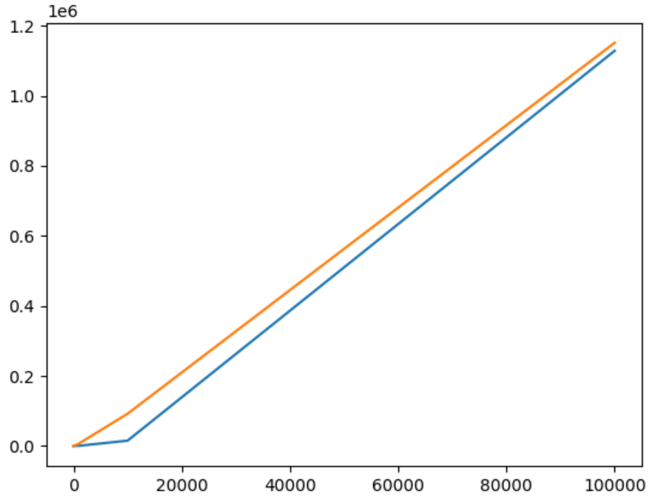
D. The results confirm the analysis since the iterative approach is linear and the recursive logarithmic line is too small to see

Question 2

B. The time complexity of merge sort is $nlog(n)$ and binary search $log(n)$. The pair-finding function's works by sorting the array and then iterating over each element applying a binary search for the difference with the target sum. Therefore, the overall time complexity is

$$T(n) = nlog(n) + n \cdot log(n) = nlog(n)$$

C.



As shown the execution time is quite close to the actual graph of $nlog(n)$