Q1 Time Complexity and Explanations

1. BFS( ) => Decrease by constant. by one Also used dictionary structure for representing the graph. It is like adjacency list.
While doing BFS, I need to traverse every link at least once.
So, there are O(|V|+|E|) links. (V = Vertex count, E = Edge count)
Also, scanning all adjacent vertices takes O(|E|) and all vertices enqueued and dequeued at most once. → O(|V|)
adjacency list ⇩ dictionary values
Worst case is => $O(|E|) + O(|V|) = O(|V|+|E|)$

2. DFS( ) => Decrease by constant. by one Also used dictionary structure for representing the graph. It is like adjacency list.
In DFS, we must traverse each node exactly once. There are O(|V|) nodes. Also I used dictionary so, I traverse the values of dictionary. Sum of size of dictionary values are equal edge count. So it is O(|E|). So time complexity is $O(|E|+|V|)$

Q3 Time Complexity and Explanations

isThereAnIndex( ) → It calls specificBinarySearch()
which is a divide and conquer algorithm. (Divide by two)

$T(n) = T(n/2) + O(1)$ ⇒ Recurrence relation

Applying master theorem ⇒ $T(n) = aT(n/b) + f(n)$

$a=1, b=2, d=0$, $a = b^d \Rightarrow 1 = 2^0$

$O(n^0 \log n) = O(\log n)$ //

Q4 Time Complexity and Explanations

This problem solved with divide and conquer algorithm.
(Divide by two)

$T(n) = 2T(n/2) + O(n)$

→ 2 times called recursion (findLargestSumSubSet)

→ findLargestSumCrossingWithMiddleSubSet

Applying master theorem ⇒ $T(n) = aT(n/b) + f(n)$

n^{1}

$a=2$, $b=2$, $d=1$

$b^d = 2^1 = 2 = a$

$O(n^d \log n) = O(n \log n)$ //

Q5 Time Complexity And Explanations

It is a decrease and conquer algorithm.
Decrease by variable size. Which is word size.

$T(n) = 2T(n - \underset{\text{constant}}{\text{len(word)}}) + \overset{\text{for loop for string iterating}}{O(n)}$

Recursive method called 2 times

$T(n) = 2T(n-1) + n \Rightarrow T(n) = 2^{n+1} - n - 2 \Rightarrow O(2^n)$