task1a: The code reads two lines from an input file, checks if there exists a pair of numbers in the second line that sum up to the target value given in the first line. if such a pair is found, it writes the indices (1-based) of the numbers to an out file otherwise it writes Impossible. The time complexity of the code is $O(n^2)$

task1b: It is same as task1a just a little modified to get $O(n)$. Because of only one for loop it contains $O(n)$ time complexity.

task2a: The code reads four lines from an input file n, temp1, n, and temp2. It performs a merge sort on temp1 and temp2 separately using the mergesort function. Then it merges the sorted arrays using the merge function. The time complexity of the code is $O(n \log n)$

task2b: This code also takes four input like task2a and performs merge operation on temp-1 and temp-2 to creat a new sorted array "new_arr". And the elements are merge into one array. The time complexity is $O(m+n)$ which can be denoted as $O(n)$.

CamScanner

task 3: The code reads two lines from an input file n and arr. it performs a merge sort which is done by mid = len(arr) // 2, then recursively doing the same thing and then lastly by merging all the element we get a sorted array. It follows the divide-and-conquer approach. And it's time complexity is O(nlogn).

task 4: The code reads a list of integers from an input file, recursively finds the maximum value using divide and conquer and writes the result in the output file. The time complexity of the code is O(logn)

CamScanner