README

Problem 1

algo_X is check the list or array is palindrome or not

1. algo_X does that

   it checks that any element have duplicate than put the first element is equal to last element and second element is equal to second last element and so on and there are no duplicate in list of any element more than one element than it return false and put that element in middle

for example

A=[1,2,3,4,5,6,7,8,9,9,8,7,6,5,4,3,2,1]

the result of this function is

A=[1,2,3,4,5,6,7,8,9,9,8,7,6,5,4,3,2,1]

algo_X does that

firstly it assign the value of i=0 and j = length of list -1 after that it check the i is smaller than j if i is bigger or equal j than it return true.If i is smaller than j than it check the value of i and j is equal if it is equal than it increase i by 1 and decrease j by 1 and go back to while loop and if the value of i and j are not equal than it assign new variable k is i + 1 and check k is smaller or equal to j if it is equal to j than it return false.If k is smaller than j then it check the value of k is equal to value of i if it is same than it swap the value of j and k and break the loop and increase i by 1 and decrease j by 1 and go back to while loop and if not than it check value of k is   equal to value of jif value of k is equal to value of j than it swap value of k with value of i and break the loop and increase i by 1 and decrease j by 1 and go back to while loop and otherwise it increase k with 1 and go back to same loop

2. Time complexity
   base case = O(n)
   In the best case the time complexity is O(n) because the list is already palindrome and increase i and decrease j than the i = j and while loop do nothing and return true

   Average case = $O(n^2)$
   Worst case = $O(n^2)$
   In the average and worst case in both case it is checking the condition and in the 2 nested while loop one will run n times and and second will run n*n times

```
def algo X(A):                                              1
    i=0                                                     1
    j = len(A) – 1                                          1
    while i < j:                                            n
        if A[i] != A[j]:                                  n-1
            k=i+1                                         n-1
            while k < j:                                  n*n
                if A[i] ==A[k]:                         n*n-1
                    A[k], A[j] = A[j], A[k]             n*n-1
                    break
                elif A[j] ==A[k]:                       n*n-1
                    A[k], A[i] = A[i], A[k]             n*n-1
                    break
                else:                                 n*n-1
                    k += 1                            n*n-1
            if k == j :                               n*n-1
                return False
        i += 1                                        n-1
        j –= 1                                        n-1
return True                                           1
```

3.
In the better algo X
best case O(nlogn)
Merge sort is a divide and conquer algorithm that has best case time **complexity** of O(nlogn).

average case O(nlogn)
worst case O(nlogn)

Merge sort is a divide and conquer algorithm that has best case time **complexity** of O(nlogn).it using first function that complexity is n and than it uses the merge sort that have complexity is nlogn and than for next function have also n complexity than we have

n+nlogn+n =O(nlogn)

PROBLEM 2

1.
In this function it takes two number and check which is bigger or which is smaller than it put number in increasing order and check the difference of this two number and subtract the difference from the first element and add in last number and check the whole list if we have number equal to

that number if we are not find any number than it takes next number than it take the difference and again subtract from first element and add in last element and if we find the value than add in list if element is equal to subtract the difference from first element than append that in first position if the value is equal to add the difference in last value than append that value in last in list and which have more value than return that list
for example we have list a = [1,3,6,8,44,55,66,56,77]
than it will return [44,55,66,77]

2.
The complexity is

Best case - O(n^3)
because the first loop will run n times and next nested for loop run n-1 times and the last loop also run n times because it have no condition everytime it will run n times than the best case is O(n^3)

Average case - O(n^3)
Worst case - O(n^3)
because in this both cases the first loop run n times and the second loop will run n-1 times and the third loop will run n times also

3.
The time complexity of better_algo_Y has

Best case - O(n^2)
Aveage case - O(n^2)
Worst case - O(n^2)

because the first loop will run n times and n times and next loop also run n*n times in all the cases