question 1 c)In the naive iterative method, we multiply 'a' by itself 'n' times

which means it has a O(n) complexity.

In the divide-and-conquer method, The recurrence relation is

T(n) = T(n/2) + O(1) similar to binary search

using recursive trees

T(n)

/ \

T(n/2) T(n/2)

/ \ / \

T(n/4) T(n/4) ...

The number of levels in the tree log₂(n)

because we keep dividing 'n' by 2 until it becomes 1.

At each level, we perform constant work, which is O(1). Therefore, the total

work done at each level is O(1).

Total Work = O(1) \* log₂(n) = O(log(n))

using master theorem

we need to compare f(n) to n^log\_b(a). In this case: a=1 b=2 f(n)= O(1)

n^log\_b(a) = n^log\_2(1) = n^0 = 1 so it falls

within Case 2 of the Master theorem as f(n)=n^log\_b(a) as O(1)=1 so since T(n) = Θ(n^log\_b(a)\*log(n))then

T(n)=O(1\*log(n))

T(n) = Θ(log n)

the experimental results meet the theortical results

//question 2 c)merge sort recurrence from lecture is 2T(n/2)+O(1)+O(n) and its time complexity using recursive tree is O(n\*log(n))

//binary search recurrence from lecture is T(n/2)+O(1)and its time complexity using recursive tree is O(log(n))

// since we are using binary search n times as binary searches is performed while looping through the array O(n)

//so the overall complexity is O(n)\*O(log(n))= O(n\*log(n)) since the merge sort and looping with binary search is O(n\*log(n))

//then overall complexity is O(n\*log(n))

//the experimental results meet the theortical results