# Q1. Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in 8n 2 steps, while merge sort runs in 64nlg n steps. For which values of n does insertion sort beat merge sort? (Exercise 1.2-2)

To compare implementations of insertion sort and merge sort on the same machine., we need to compare the time complexities of insertion and merge sortText, letter

Description automatically generated

# Q2. We can express insertion sort as a recursive procedure as follows. In order to sort A [1 .. n], we recursively sort A [1 .. n-1] and then insert A [n] into the sorted array A [1 .. n-1]. Write a recurrence T(n) for the running time of this recursive version of insertion sort. (Exercise 2.3-4)

Text, letter

Description automatically generated

# Q3. Describe a Θ (n lg n) - time algorithm that, given a set S of n integers and another integer x, determines whether there exist two elements in S whose sum is exactly x. (Exercise 2.3-7)

we can sort the array with merge sort Θ (n log n) and then for each element (say y) in the array, we can do a binary search for (x – y) on the sorted array. So, the algorithm will run in Θ(nlog n).

Text, letter

Description automatically generated

# Q4. Sort all the functions below in increasing order of asymptotic (big-O) growth. If some have the same asymptotic growth, then be sure to indicate that. As usual, lg means base 2.

# 1. 5n

# 2. n4

# 3. 4 lg n

# 4. nn/4

# 5. n1/2log4n

Answer: 4lgn < 5n < n4  < n1/2log4n  < nn/4

Text, letter

Description automatically generated

# Q5. Prove that 2 n+1 = O(2n ). (Exercise 3.1-4) (Hint: Try to satisfy the definition of O-notation with some constants c, n0 > 0)

Text, letter

Description automatically generated