Berkcan EKİCİ
171044015

1) I use basic algorithm.

$T(n) = T(n/2) + O(1)$

Since the Master Theorem works with recurrences of the form:

$T(n) = aT(n/b) + n^c$

$a = 1, b = 2, c = 0$

$\Rightarrow \Theta(n^c \log n) = \Theta(n^0 \log n) = \underline{\Theta(\log n)}$

Berkcan EKİCİ
171044015

2) My algorithm is to sort the array with merge sort.
Because we need an ordered array and we have to do it with divide and conquer algorithm.

Analysis:

- The merge sort function is breaking the problem size of n into two subproblems of size n/2 each. Also, we deduced that the merge function is $\Theta(n)$.

$$T(n) = 2T(n/2) + \Theta(n) + \Theta(1)$$

It's following Master's Theorem equation format.

Hence $a=2, b=2$ and $f(n) = \Theta(n)$

$$n^{\log_a b} = n^{\log_2 2} = n$$

Thus above recurrence is matching case 2 of Master's Theorem.

Time complexity is $O(n \log n)$

Berkcan EKİCİ
171044015

4) My algorithm:

- Divide : seperate list into two halves [low, mid] and [mid+1, high]
- Conquer : recursively count inversions in each list
- Combine ; count inversions
- Return sum of three counts.

Analyse:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n=1 \\ T(n/2) + T(n/2) + \Theta(n) & \text{if } n>1 \end{cases}$$

It's following Master's Theorem equation format.

Hence $a=2, b=2$ and $f(n)=\Theta(n)$

$n^{\log_a b} = n^{\log_2 2} = n$

Thus above recurrence is matching case 2 of Master's Theorem.

Thus time complexity is $O(n \log n)$.

Berkcan EKİCİ

5) Brute force algorithm:

$$T(n) = \sum_{i=0}^{n} 1 \qquad T(n) = n = O(n)$$

Divide and Conquer algorithm:

$$T(n) = T(n/2) + 1$$

After k iterations,

$$T(n) = T(n^k) + k$$

Binary tree created by binary search can have maximum height $\log_2 n$

So, $k = \log_2 n \Rightarrow n = 2^k$

$$T(n) = T(2^k/2^k) + 1$$

$$= T(1) + k$$

From base case of recurrence,

$$T(n) = 1 + k = 1 + \log_2 n$$

$$T(n) = O(\log_2 n)$$