MTH 4320 Homework 2

Yaohui Wu

February 14, 2024

1 Problem 1

Solution.

$$\begin{aligned} T(1) &= O(1) \\ T(n) &= 2 \cdot T(n-1) \\ &= 2(2 \cdot T(n-2)) \\ &= 2^n \cdot T(n-(n-1)) \\ &= 2^n \cdot T(1)) \\ &= O(2^n) \end{aligned}$$

The running time is

$$\begin{aligned} T(1) &= O(1) \\ T(n) &= O(2^n) \end{aligned}$$

■

2 Problem 2

Solution. We have

$$\begin{aligned} T(1) &= O(1) \\ T(n) &= n + 4 \cdot T\left(\frac{n}{2}\right) \\ &= n + 4\left[\frac{n}{2} + 4 \cdot T\left(\frac{n}{2^2}\right)\right] \\ &= n + 2n + 4^2 \cdot T\left(\frac{n}{2^2}\right) \\ &= n + 2n + \cdots + 2^{k-1}n + 4^k \cdot T\left(\frac{n}{2^k}\right) \end{aligned}$$

$T(n)$ converges when $T\left(\frac{n}{2^k}\right) = T(1)$ so $k = \log n$ then we have

$$\begin{aligned} T(n) &= n + 2n + 2^2 n + \cdots + 2^{\log(n)-1} n + 4^{\log n} T(1) \\ &= n + 2n + 2^2 n + \cdots + \frac{n^2}{2} + n^2 \cdot T(1) \\ &= O(n^2) \end{aligned}$$

The running time is

$$\begin{aligned} T(1) &= O(1) \\ T(n) &= O(n^2) \end{aligned}$$

■

3 Problem 3

Solution. The algorithm is

1. Sort A in ascending order using merge sort.
2. Find the set of all sums of pairs of elements from A.
3. Iterate over the pairs and use binary search to find if there is another pair that sums to 2000.

The running time is $O(n \log n) + O(n^2) + O(n^2 \log n) = O(n^2 \log n)$. ■

4 Problem 4

Solution. The algorithm is

1. Divide L into two intervals L and R of the same size.
2. Find the largest number in L and R using recursion.
3. Return the largest number in each interval then compare them to find the largest number in L.

The running time is $O(n)$. ■

5 Problem 5

Solution. The algorithm is

1. Divide A into two intervals L and R of equal size.
2. Sort the elements in each interval and iterate over the elements of L and R.

3. If the current element in L is greater than an element in R then we add the number of the remaining elements in L to the count of the flipped pairs.

4. If the current element in L is not greater than an element in R then we skip it and we make sure we only count every flipped pair exactly once.

5. Merge L and R then return the sum of the count of the flipped pairs using recursion.

The running time is $O(n \log n)$. ■