Algorithm

Quiz #1

April 8, 2024

Please answer the following questions. You are asking to select only ONE correct answer for each question.

1. What is the asymptotic tight bound of $T(n) = 2T(\sqrt(n)) + lgn$?

   A) $\Theta(nlgn)$
   B) $\Theta(nlglgn)$
   C) $\Theta(lgnlglgn)$
   D) $\Theta(lgnlglglgn)$

2. Which of the following recurrence relations describes the time complexity of the binary search algorithm on a sorted array?

   A) $T(n) = 2T(n/2) + O(1)$
   B) $T(n) = T(n-1) + O(1)$
   C) $T(n) = T(n/2) + O(1)$
   D) $T(n) = T(n^2) + O(1)$

3. What is the time complexity of $T(n) = T(n-2) + 2lgn$?

   A) $\Theta(n \lg n)$
   B) $\Theta(\sqrt{n})$
   C) $\Theta(n^2)$
   D) $\Theta(2^n)$

4. In a distant kingdom, there was an ancient library that housed millions of books. These books recorded the history, knowledge, and myths of the kingdom, sorted according to their importance and content codes, from the least to the most important. One day, the king decided to find a book that contained an ancient prophecy to resolve the crisis facing the nation. This book was very special, said to guide the kingdom toward prosperity. However, the library was too large and had too many books for a simple search to be feasible.

   So, the king summoned his sage, a wise man known as the "Book Seeker." The Book Seeker proposed a clever plan: he would not check each book from beginning to end. Instead, he would start with the middle book, and if that book's importance index was lower than the book he was looking for, he knew he should search among the books on the right side; if higher, then he would search on the left side. In this way, the Book Seeker could eliminate half of the books with each search, significantly speeding up the process.

   After several such searches, the Book Seeker finally found the precious book of prophecy, and the king and his kingdom were thus guided toward a path of prosperity.

   Now, the question is: Assuming there are n books in the library, and using the method described by the Book Seeker to find a specific book, analyze its time complexity.

   A) $O(n \lg n)$
   B) $O(\sqrt{n})$
   C) $O(n^2)$

   D) $O(lgn)$

5. Once upon a time, in a kingdom of intricate data structures and perplexing problems, there lived a diligent algorithm. This algorithm was known far and wide for its remarkable ability to sort items swiftly and efficiently. Let's journey into the heart of this enchanting tale and uncover the secrets of its prowess.

   In a bustling marketplace where numbers danced like merchants vying for attention, there was chaos. Every number sought its rightful place in the grand order of things, yet confusion reigned supreme. Amidst this turmoil, our valiant algorithm stepped forth, armed with its trusty divide-and-conquer strategy.

   With each wave of its algorithmic wand, the list of numbers was divided into smaller, more manageable segments. Like a skilled conductor orchestrating a symphony, it compared neighboring elements, swapping them if they were out of order. Through this intricate dance of comparison and rearrangement, harmony began to emerge from the chaos.

   As the algorithm continued its tireless work, the segments grew increasingly sorted, until finally, the entire list stood in perfect order. The marketplace erupted into cheers and applause as the algorithm bowed gracefully, its mission accomplished.

   But amidst the celebration, a wise sage approached with a curious question, "Oh noble algorithm, how might thy splendid performance be measured in terms of time complexity?"

   A) $O(n \lg n)$
   B) $O(\sqrt{n})$
   C) $O(n^2)$
   D) $O(lgn)$

6. Once upon a time, in the land of Algorithmia, there lived a diligent gardener named Master Tree. Master Tree tended to a magical garden filled with all sorts of enchanting plants and flowers. But among all the flora in his garden, there was one extraordinary plant known as the Blossom Tree.

   The Blossom Tree was unlike any other in Algorithmia. Its branches intertwined in intricate patterns, and its leaves shimmered with an otherworldly glow. But what truly set the Blossom Tree apart was its ability to organize itself in a most peculiar manner.

   You see, whenever a new blossom bloomed on the tree, it would arrange itself meticulously, ensuring that every blossom on the left branch was smaller than those on the right. This made the Blossom Tree not only a sight to behold but also an incredibly efficient way to search for flowers of varying sizes.

   As the villagers marveled at the beauty of the Blossom Tree, they often wondered: "How does the Blossom Tree manage to maintain such order and balance?"

   And so, one day, Master Tree gathered the villagers beneath the boughs of the Blossom Tree and shared its secret. He explained that the tree followed a special rule: each time a new blossom appeared, it would compare itself to the existing blossoms and find its rightful place in the hierarchy.

   The villagers were fascinated by this revelation and bombarded Master Tree with questions. "But how long does it take for the Blossom Tree to rearrange itself when a new blossom appears?" they asked eagerly.

   Master Tree smiled knowingly and replied, "Ah, the time it takes for the Blossom Tree to reorganize itself depends on the number of blossoms already present. The larger the tree grows, the longer it takes to maintain its order."

   And with that, the villagers departed, their minds buzzing with thoughts of time and complexity.

   Now, dear reader, can you guess which algorithm the Blossom Tree represents? And what might be the BEST time complexity of this algorithm as the number of elements grows?

   A) $O(n)$
   B) $O(\sqrt{n}lgn)$
   C) $O(nlgn)$
   D) $O(lgn)$