ASSIGNMENT 1

Q1.3.3, Q1.3.4, Q1.3.5, Q1.3.6

Question 3: Calculate the runtime of the Iterative Version of the Binary Search Algorithm experimentally (Give an answer in terms of seconds).

Answer:

RECURSIVE:

Array Size	Time in ns	Time in seconds
1	1190	0.0000011900
10	928	0.0000009280
50	829	0.0000008290
200	955	0.0000009550
500	1065	0.0000010650
1000	1113	0.0000011130
Average time taken	1013.3	0.0000010133

Question 4: Calculate the runtime of the Recursive Version of the Binary Search Algorithm experimentally (Give an answer in terms of seconds).

Answer:

ITERATIVE:

Array Size	Time in ns	Time in seconds
1	1432	0.0000014320
10	846	0.0000008460
50	713	0.0000007130
200	822	0.0000008220
500	819	0.0000008190
1000	871	0.0000008710
Average time taken	917.2	0.0000009172

Question 5: Justify the difference or similarity between the run-times of the Iterative and Recursive versions of the Binary Search Algorithm.

Answer:

Theoretical justification for the similarity in runtimes of iterative and recursive binary search focuses on their time complexity, which is **O(log n)** for both, indicating that the number of steps required grows logarithmically with the size of the input array. However, in practice, the recursive version may have slightly longer runtimes due to the overhead associated with recursive calls and managing the call stack, especially for larger input sizes or deeper recursion depths. The iterative version, on the other hand, uses constant space and may be slightly more efficient in terms of space complexity and reducing overhead.

Question 6: Prove the correctness of the Binary Search Algorithm using Mathematical Induction.

Answer:

1. **Base Case**: Show that the algorithm works for a simple case, such as an array of size 1 or 2. For a single-element array, the middle is the only element, and if it matches the target, the algorithm correctly identifies the target.

2. **Induction Hypothesis**: Assume that the algorithm works correctly for an array of size k.

3. **Induction Step**: Prove that if the algorithm works for an array of size k, it also works for an array of size k+1. Consider the step where the array is divided into two halves. By the hypothesis, binary search correctly determines if the target is in either half of an array of size k. For k+1, the extra element doesn't alter the algorithm's logic, as it either falls into one of the two halves or is the middle element compared directly.

By proving these steps, we show that binary search will correctly determine whether a target is present in a sorted array, regardless of the array's size, thus proving its correctness through mathematical induction.