Recursive binary search:

**Big O:** O(Log n): This algorithm has a ‘divide and conqueror’ tactic, which reduces the number of runs the programs has to make. Each divide breaks the array into smaller sub arrays, which results in a time complexity of O(Log n).

**Tree:**

A drawing on a paper

Description automatically generated with low confidence

**Example Run:**

**Text

Description automatically generated**

Recursive Factorial:

**Big O:** O(n): Since the recurrence relation for T(n) = T(n-1) + c, which results in complexity O(n).

**Tree:**

A piece of paper with writing on it

Description automatically generated with medium confidence

**Example Run:**

**Text

Description automatically generated**

Recursive Fibonacci Sequence:

**Big O:** O(2^n): The recurrence relation is T(n) = T(n-1) + T(n-2), which is exponential in nature, which results in a time complexity of O(2^n).

**Tree:**

A picture containing text, whiteboard

Description automatically generated

**Example Run:** Text

Description automatically generated