1. **Question 1:**

int [] beautiful(int [] A) {

for (int i = 0; i < A.length; i++) {

A[i] = A[i] \* 2;

}

return A;

}

* Best-case running time: The algorithm iterates through all n elements of the array exactly once, so the best-case running time is (O(n)).
* Worst-case running time: Similarly, the algorithm iterates through all n elements of the array exactly once, so the worst-case running time is also (O(n)).

1. **Question 2:**

2^n , 2^(n + 1), 2^(2n), 2^( 2^n )

1. **Question 3:**

* O(1) - Constant Time Complexity

Algorithm: Accessing an element in an array by index.

Example: arr[i]

* O(log n) - Logarithmic Time Complexity

Algorithm: Binary Search.

Example: Finding an element in a sorted array.

* O(n) - Linear Time Complexity

Algorithm: Linear Search.

Example: Searching for an element in an unsorted array.

* O(n log n) - Linearithmic Time Complexity

Algorithm: Merge Sort.

Example: Sorting an array using the merge sort algorithm.

* O(n^2) - Quadratic Time Complexity

Algorithm: Bubble Sort.

Example: Sorting an array using the bubble sort algorithm.

* O(2^n) - Exponential Time Complexity

Algorithm: Problem using brute force.

Example: Finding the shortest path that visits all cities and returns to the starting point.

1. **Question 4:**

In lesson 2, fib(n) has the linear time complexity O(n) because we use iterative to solve the problem. So it can not apply Master Theorem (The Master Theorem just apply for recursive or divide algorithm (like Binary Search)