**Identify the Big-O Notation of the following Blocks of Codes**: (Identify and Discuss the Block of Codes' Structure which complies to the identified Notation)

const products = [50, 20, 60, 15, 30];

const sortByPrice = (arr) => {

return arr.sort((a,b) => a-b)

}

console.log(sortByPrice(products));

module.exports = {sortByPrice};

The provided block of code sorts an array of product prices in ascending order using the sort method. The **Big-O Notation** for this code is **O(n log n)**. This is because the sort method in JavaScript uses an efficient sorting algorithm, typically a variant of TimSort, which has a time complexity of **O(n log n)**.

1. **Array Declaration**: The array products is declared with five initial elements. This operation is **O(1)** since it's just initializing a fixed number of elements.
2. **Sort Method**:
   * The sort method is called on the array. The JavaScript sort method utilizes an efficient sorting algorithm which typically performs **O(n log n)** operations where n is the number of elements in the array.
   * The function (a, b) => a - b is a simple comparison function which determines the order of the elements. Each comparison is constant time **O(1)**, but since the entire sort operation follows the **O(n log n)** complexity, the comparison function does not change this overall complexity.
3. **Console Log**: The console log statement prints the sorted array. This operation is **O(1)** as displaying results is not dependent on the size of the array.

const sortByPrice = (arr) => {

return arr.sort((a, b) => a - b);

}

In summary, the sortByPrice function sorts the products array in ascending order based on their prices, and this sorting operation has a time complexity of **O(n log n)**, making it efficient for larger arrays as well.