Pseudocode for algorithm

create an array myArray of size n

create an integer parsing variable for parsing the array

create an integer partition variable that marks the end of the white disk partition

create a counting variable that counts the number of times the while loop is used

while count variable is less than n, do the following:

    if myArray[partition] equals black AND myArray[parsing] equals white
        Swap the strings at these two locations
        Increment parsing and partition
    if myArray[partition] equals white
        The partition already holds a white value, meaning the swap is unnecessary
        Increment parsing and partition
    if neither of these scenarios are true, meaning they are both black,
        Increment Parsing, but keep partition where it is, as it needs to stay there to swap in a white disk.

    increment the counting variable

Display the sorted array.

Efficiency Class and Number of Steps

The Efficiency class of this algorithm is **O(n)**, meaning that the efficiency is dependent on the size of the array, and that there are no nested loops. The while loop runs n times with 11 steps, the first for loop runs n times with 4 steps, and the send for loop runs n times with 1 step. Outside of the loops there are 18 steps. So the number of steps is equal to:

11n + 4n + n + 18, or

16n + 18

Statement on Possible Improvement

This algorithm could be improved by making the code more efficient and decreasing the number of steps needed, but the efficiency class of **O(n)** is pretty efficient as it is.