The time complexity of the Fibonacci series with recursion is indeed O(2^n), while the time complexity of merge sort is O(n log n). This difference arises from the different structures of the recursive calls in each algorithm.  
In the Fibonacci series, the recursive calls are nested, meaning that one call to the Fibonacci function requires two additional calls to itself. This leads to an exponential growth in the number of recursive calls as the input value (n) increases. Hence, the time complexity is O(2^n).

On the other hand, the recursive calls in merge sort are not nested. Instead, they are parallel, meaning that each call to the merge sort function divides the input array into two halves and makes two recursive calls to sort each half. This leads to a logarithmic growth in the number of recursive calls as the input size (n) increases. Hence, the time complexity of merge sort is O(n log n).