0.1 A possible step further: Iterative Fast Fourier Transform Algorithm

If we could arrange from the start the input components in a order that resembles the one reached by the deepest level of recursion (that is, when $N = 1$), we could create a non-recursive approach for the FFT algorithm that works "bottom up". The idea behind this algorithm is the following:

1. we store in an array $A[0, \ldots, n-1]$ in the order in which they appear in the leaves of the recursion tree;
2. we take the elements in pairs, applying the butterfly operation that multiplies/adds two inputs with the twiddle factor to obtain $X_k = \text{evenFFT}_k + e^{-i\frac{2\pi}{N}k} \cdot \text{oddFFT}_k$ and $X_{k+\frac{N}{2}} = \text{evenFFT}_k - e^{-i\frac{2\pi}{N}k} \cdot \text{oddFFT}_k$. This will provide us with $N/2$ size 2 DFTs of each pair;
3. we replace the pairs in the array with their DFTs;
4. we take these $N/2$ DFTs (of size 2 each) in pair and we compute the DFT of the $N/4$ groups of 4 input each that sit at the upper level in the recursion tree. This operation will apply two butterfly operations (since we are dealing with inputs of length 4);
5. we write these $N/4$ size 4 DFTs in the array;
6. we iterate the procedure untile the array holds 2 size $N/2$ DFTs, which we can combine using $N/2$ butterfly operations to obtain out final N size DFT.

To do this, we must firstly fill an array wtih the input component in the particular order that they appear in the leaves of the recursion graph. Let's examine the ordere in which the input components end up.

Original index	0	1	2	3	4	4	6	7
Final index	0	4	2	6	1	5	2	7
Original index (binary)	000	001	010	011	100	101	110	111
Final index (binary)	000	100	010	110	001	101	110	111