As pointed out by Emmanuel Thomé, the proof of the $O(n \log n \log\log n)$ complexity of the Schönhage-Strassen algorithm in *Modern Computer Arithmetic* (Theorem 2.3) is wrong. Indeed, the proof assumes that the recursive calls in step 8 cost $O(n' \log n' \log\log n')$ each, but the "constant" hidden behind the $O()$ might grow throughout the algorithm, and might not be constant at the end.

A correct proof can be found in the article *Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients*, by Arnold Schönhage, in the proceedings of Computer Algebra, EUROCAM'82, LNCS 144, pages 3-15. We reproduce it here with the notations of Algorithm 2.4 in the book *Modern Computer Arithmetic*.

Consider n is of the form $\nu 2^t$, with $t-1 \leq \nu \leq 2t-1$ (Schönhage calls such a number a *suitable number*). Such numbers are divisible by a large power of two, with only about $\log n$ non-zero high significant bits. In Algorithm 2.4, we take $K = 2^k$ with $k = \lfloor t/2 \rfloor + 1$. (More or less, we have $K \leq \sqrt{n}/\log n$.) We can check that for $t \geq 1$, K divides n. We then take $n' = \mu 2^m$ with $\mu = \lceil (\nu+1)/2 \rceil$ and $m = \lceil t/2 \rceil + 1$. We can check that n' is a suitable number too, i.e., $m - 1 \leq \mu \leq 2m - 1$. Indeed:

$$\mu = \left\lceil \frac{\nu+1}{2} \right\rceil \geq \frac{\nu+1}{2} \geq \frac{t}{2} \geq m-1,$$

and since ν is an integer:

$$\mu = \left\lceil \frac{\nu+1}{2} \right\rceil \leq \left\lceil \frac{2t}{2} \right\rceil \leq t \leq 2m-2,$$

since $m \geq t/2 + 1$.

We also have to check that this value of n' satisfies the condition of step 3 of Algorithm 2.4, i.e., $n' \geq 2n/K + k$, and n' multiple of K.[1] Since $n' = \mu 2^m$ with $m \geq k$, n' is clearly a multiple of $K = 2^k$. From the definitions of m and k, it follows $m + k = t + 2$, thus the bound $n' \geq 2n/K + k$ translates into $n' \geq \nu 2^{m-1} + k$. Now $k = \lfloor t/2 \rfloor + 1 \leq 2^{\lceil t/2 \rceil} = 2^{m-1}$, so that it suffices to prove $n' \geq \nu 2^{m-1} + 2^{m-1}$. Since $n' = \mu 2^m$, this is equivalent to $2\mu \geq \nu + 1$, which clearly holds by the definition of μ.

Note that the value of n' chosen by Schönhage's method is not the smallest multiple of K satisfying $n' \geq 2n/K + k$. Take for example $n = 768 = 6 \cdot 2^7$, with $\nu = 6$ and $t = 7$. It yields $k = 4$, thus we should have $n' \geq 100$. The smallest multiple of $K = 16$ is 112, whereas Schönhage's method chooses $n' = 128$ instead.

For the detailed complexity analysis, we refer the reader to the above mentioned article. What is missing however in Schönhage's article is the justification that after $O(\log\log n)$ steps we have sufficiently small numbers to multiply. Indeed, at each step the total size is multiplied by $n'/n = 4\mu/\nu$. Let us write $\nu_0 = \nu, \nu_1 = \mu, ...$ the sequence of values in the recursive tree. The worst case arises when all values are even, since with $\nu_{i+1} = \lceil (\nu_i + 1)/2 \rceil$ we then have $\nu_{i+1} = \nu_i/2 + 1$. This case happens when $\nu_0 = 2^k + 2$, then $\nu_1 = 2^{k-1} + 2$, ... If we go down to $\nu_d = 3$ at the bottom of the tree, the total size of all numbers in the tree at depth d can be up to $5 \cdot 2^d$. If $d = O(\log\log n)$, then the numbers to be multiplied at

[1] Schönhage requires $n' \geq 2n/K + k + 1$, where the extra $+1$ makes it easier the reconsctruction after the inverse FFT, but $n' \geq 2n/K + k$ works too.

the bottom of the tree have $O(\log\log n)$ bits, and we have $O(n)$ of them, thus using a naive algorithm costs $O(n\log^2\log n)$.p

The key ingredient is to have K not too large, i.e., at most about $\sqrt{n}/\log n$, to ensure the rounding of n' up to a multiple of K does not increase too much the transform size.

We can also choose k as the largest integer such that $k2^{2k} \le n$. Then the cost of the FFT transforms is $O(k2^k n')$, thus if we divide by $n\log n$, since $2^k n' \le 2n + 2^{2k} \le n(2 + 1/k)$, we get:

$$\frac{k2^k n'}{n\log n} \le \frac{2k+1}{\log n} \le 1,$$

since $\log n \ge 2k + \log k$ — with a base-2 logarithm — and we can assume $k \ge 2$. This is more or less equivalent to Schönhage's method, except it gives more *suitable numbers* at the top of the recursion tree. Indeed, n can have its top half bits set, against only a logarithmic part with Schönhage's method.