Back to multiplication of polynomials:
an optimal algorithm
Alternatively, the coefficients of the product of two polynomials of degree $n - 1$ can be computed as follows.

1. Fix any $2n-1$ pairwise distinct field elements (points) $t_0, t_1, \ldots, t_{2n-2}$ and evaluate $x(\lambda)$ and $y(\lambda)$ at these points.

2. Compute $z(t_k) = x(t_k)y(t_k)$, $k = 0, 2, \ldots, 2n - 2$.

3. Interpolate the coefficients of $z(\lambda)$.

This algorithm requires only $2n - 1$ multiplications.
Let $\bar{x} = (x_0, x_1, \ldots, x_{n-1}, 0, 0, \ldots, 0)$ and $\bar{y} = (y_0, y_1, \ldots, y_{n-1}, 0, 0, \ldots, 0)$, and let

$$W = W(t_0, t_1, \ldots, t_{2n-2}) = \begin{pmatrix} 1 & t_0 & \cdots & t_{2n-2}^2 \\ 1 & t_1 & \cdots & t_{2n-2}^2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_{2n-2} & \cdots & t_{2n-2}^2 \end{pmatrix}$$

Then

$$W\bar{x}^T = (x(t_0), x(t_1), \ldots, x(t_{2n-2}))^T,$$

$$W\bar{y}^T = (y(t_0), y(t_1), \ldots, y(t_{2n-2}))^T,$$

and

$$Wz^T = (z(t_0), z(t_1), \ldots, z(t_{2n-2}))^T,$$

implying

$$z^T = (z_0, z_1, \ldots, z_{2n-2})^T = W^{-1}((W\bar{x}^T) \cdot (W\bar{y}^T)).$$
\[ x(\lambda) = \sum_{i=0}^{n-1} x_i \lambda^i = \lambda^{n-1} \sum_{i=0}^{n-1} x_{n-1-i} \lambda^{-i} = \lambda^{n-1} \hat{x}(\lambda^{-1}) \]

\[ y(\lambda) = \sum_{j=0}^{n-1} y_j \lambda^j = \lambda^{n-1} \sum_{j=0}^{n-1} y_{n-1-j} \lambda^{-j} = \lambda^{n-1} \hat{y}(\lambda^{-1}) \]

\[ z(\lambda) = \lambda^{2n-2} \sum_{k=0}^{2n-2} z_k \lambda^{k-2n+2} = \lambda^{2n-2} \sum_{k=0}^{2n-2} z_{2n-2-k} \lambda^{-k} = \lambda^{2n-2} \hat{z}(\lambda^{-1}) \]

where \( z_k = \sum_{i+j=k} x_i y_j, \ k = 0, 1, \ldots, 2n - 2. \)
Substituting $\beta$ for $\lambda^{-1}$, we see that the set of coefficients of $x(\lambda)y(\lambda)$ is equal to set of coefficients of $\tilde{x}(\beta)\tilde{y}(\beta)$, where

$$\tilde{x}(\beta) = \sum_{i=0}^{n-1} x_{n-1-i}\beta^i \quad \tilde{y}(\beta) = \sum_{j=0}^{n-1} y_{n-1-j}\beta^j$$

Thus, evaluation of $x(\lambda)$ and $y(\lambda)$ at a point $t$ corresponds to evaluation of $\tilde{x} (\lambda)$ and $\tilde{y} (\lambda)$ at $t^{-1}$, and vice versa.

In particular, evaluation of $x(\lambda)$ and $y(\lambda)$ at 0 corresponds to evaluation of $\tilde{x} (\lambda)$ and $\tilde{y} (\lambda)$ at $\infty$, and vice versa: we just define $x(\infty) = x_{n-1}$, $y(\infty) = y_{n-1}$, and $z(\infty) = z_{2n-2}$.

That is, the row of $W$ corresponding to $\infty$ is $(0,0,\ldots,0,1)$. 
Fourier transform and convolution
\[ \hat{f}(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i \omega x} \, dx \]

\[ f(x) = \int_{-\infty}^{\infty} \hat{f}(\omega)e^{2\pi i x \omega} \, d\omega \]

\[ h(x) = (f \ast g)(x) = \int_{-\infty}^{\infty} f(y)g(x - y) \, dy \]

\[ \hat{h}(\omega) = \hat{f}(\omega)\hat{g}(\omega) \]
Let $f, g, h : \mathbb{Z} \to \mathbb{R}$ be such that

- $f(x) = g(x) = 0$ for $x < 0$ or $x > n - 1$, and
- $h(x) = 0$ for $x < 0$ or $x > 2n - 2$,

and let

$$[f](\lambda) = \sum_{x=0}^{n-1} f(x) \lambda^x \quad [g](\lambda) = \sum_{x=0}^{n-1} g(x) \lambda^x \quad [h](\lambda) = \sum_{x=0}^{2n-2} h(x) \lambda^x$$

$$[h](\lambda) = [f](\lambda)[g](\lambda) = \sum_{x=0}^{2n-2} h(x) \lambda^x = \sum_{x=0}^{2n-2} \sum_{y=0}^{2n-2} f(y) g(x - y) \lambda^x$$

$$\hat{f}(\omega) = \sum_{x=0}^{n-1} f(x) (e^{-2\pi i \omega})^x \quad \hat{g}(\omega) = \sum_{x=0}^{n-1} g(x) (e^{-2\pi i \omega})^x$$

$$\hat{h}(\omega) = \hat{f}(\omega)\hat{g}(\omega)$$
Discrete Fourier transform
Let \( \omega_n \) be a primitive \( n \)th root of unity. That is, \( \omega_n^n = 1 \), but for all \( i = 1, 2, \ldots, n - 1 \), \( \omega_n^i \neq 1 \).

**Remark** If \( m \mid n \), then \( \omega_n^{n/m} \) is a primitive \( m \)th root of unity (why?).

Let

\[
W(\omega_n) = W(1, \omega_n, \ldots, \omega_n^{n-1}).
\]

That is,

\[
W(\omega_n) = \begin{pmatrix}
1 & 1 & \cdots & 1 \\
1 & \omega_n & \cdots & \omega_n^{n-1} \\
\vdots & \vdots & \ddots & \vdots \\
1 & \omega_n^{n-1} & \cdots & \omega_n^{(n-1)(n-1)}
\end{pmatrix}
\]

The \( i, j \) entry of \( W(\omega_n) \) is \( \omega_n^{ij} \). It is more convenient to count the entries from 0 to \( n - 1 \).
Example $n = 4$, $\omega_4 = i$. Then

$$\omega_4^{-1} = \omega_4^3 = -i = \bar{i}.$$ 

$$W(i) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

and

$$W^{-1}(i) = \frac{W(-i)}{4} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}$$

Proposition $W^{-1}(\omega_n) = \frac{1}{n} W(\omega_n^{-1})$. 
Proof It suffices to show that $W(\omega_n)W(\omega_n^{-1}) = nI_n$. The $i,j$ entry of $W(\omega_n)W(\omega_n^{-1})$ is

\[
\begin{pmatrix}
1, \omega_n^i, \ldots, \omega_n^{i(n-1)}
\end{pmatrix}
\begin{pmatrix}
1 \\
\omega_n^{-j} \\
\vdots \\
\omega_n^{-j(n-1)}
\end{pmatrix}
= \sum_{k=0}^{n-1} \omega_n^{ik} \omega_n^{-jk}
= \sum_{k=0}^{n-1} (\omega_n^{i-j})^k.
\]

If $i = j$, then $(\omega_n^{i-j})^k = 1$, implying $\sum_{k=0}^{n-1} (\omega_n^{i-j})^k = n$. Otherwise, i.e., if $i \neq j$, $\omega_n^{i-j} \neq 1$, because $\omega_n$ is a primitive $n$th root of unity and $-n < i-j < n$. Therefore,

\[
\sum_{k=0}^{n-1} (\omega_n^{i-j})^k = \frac{(\omega_n^{i-j})^n - 1}{\omega_n^{i-j} - 1} = \frac{(\omega_n^n)^{i-j} - 1}{\omega_n^{i-j} - 1} = \frac{1 - 1}{\omega_n^{i-j} - 1} = 0.
\]

$\square$
Computing $W(\omega_n)\mathbf{x}^T$, where
\[\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})\] and
\[n = 2^k, \quad k = 1, 2, \ldots\]
Denote $\omega_n$ just by $\omega$. Then

$$W(\omega_n)x^T = (x(\omega^0), x(\omega^1), \ldots, x(\omega^{n-1}))^T,$$

where $x(\lambda) = \sum_{i=0}^{n-1} x_i \lambda^i$. That is, we have to compute $\{x(\omega^j)\}_{j=0,\ldots,n-1}$.

Let

$$x(\lambda) = \sum_{i=0}^{\frac{n}{2}-1} x_{2i} \lambda^{2i} + \lambda \sum_{i=0}^{\frac{n}{2}-1} x_{2i+1} \lambda^{2i} = x_0(\lambda^2) + \lambda x_1(\lambda^2).$$

Then

$$x(\omega^j) = x_0(\omega^{2j}) + \omega^j x_1(\omega^{2j}).$$

Note that

$$\left(\omega^{\frac{n}{2}+j}\right)^2 = \omega^{n+2j} = \omega^n \omega^{2j} = \omega^{2j}.$$ 

In addition, $\left(\omega^{\frac{n}{2}}\right)^2 = 1$. Therefore, $\omega^{\frac{n}{2}} = -1$ (why?), implying

$$x(\omega^{\frac{n}{2}+j}) = x_0(\omega^{2j}) - \omega^j x_1(\omega^{2j}), \quad j = 0, 1, \ldots, \frac{n}{2} - 1.$$ 

Finally, $\omega^2$ is a primitive $\frac{n}{2}$th of unity.
Thus, we have

\[ x(\omega^j) = x_0(\omega^{2j}) + \omega^j x_1(\omega^{2j}), \quad j = 0, 1, \ldots, \frac{n}{2} - 1 \]

\[ x(\omega^{\frac{n}{2} + j}) = x_0(\omega^{2j}) - \omega^j x_1(\omega^{2j}), \quad j = 0, 1, \ldots, \frac{n}{2} - 1. \]

Let \( F(n) \) denote the number of operations (additions and multiplications by a constant) needed for computing the \( n \) point discrete Fourier transform by the above described algorithm. Then

\[ F(n) = 2F\left(\frac{n}{2}\right) + \left(\frac{n}{2} - 1\right) + n \]

(what are \( n/2 - 1 \) and \( n \)?) implies

\[ F(n) \leq \frac{3}{2} n \log_2 n. \]

The above algorithm is called \textit{Fast Fourier Transform} – FFT.
Remark Over $\mathbb{C}$, $\omega_n$ can be *recursively* computed as follows.

$$\omega_{2n} = \frac{\omega_n + 1}{|\omega_n + 1|}$$
Finally, if the underlying domain contains a primitive $2^k$th root of unity, we can compute the coefficients of the product of two degree-$(n - 1)$, $n = 2^{k-1}$, polynomials in $O(n \log n)$ operations, $2n - 1$ of which are multiplications and the rest are additions and multiplications by a constant.

$$z^T = \frac{1}{2n} W(\omega_{2n}^{-1}) \left( W(\omega_{2n}) \left( \begin{array}{c} x^T \\ 0^T_n \end{array} \right) \cdot W(\omega_{2n}) \left( \begin{array}{c} y^T \\ 0^T_n \end{array} \right) \right)$$

First we compute

$$W(\omega_{2n}^{-1}) \left( W(\omega_{2n}) \left( \begin{array}{c} x^T \\ 0^T_n \end{array} \right) \cdot W(\omega_{2n}) \left( \begin{array}{c} y^T \\ 0^T_n \end{array} \right) \right)$$

and then divide each component $z_k$, $k = 0, 1, \ldots, 2n - 2$, of $z$ by $2n$ (which is the multiplication by the constant $1/2n$).
**Example** (Computing the “basic” symmetric functions of $n$ variables).

The basic symmetric functions of $x_0, x_1, \ldots, x_{n-1}$ are the coefficients of

$$x = \prod_{i=0}^{n-1} (\lambda + x_i)$$

which can be computed as follows.

Let

$$x_{k,\ell} = \prod_{i=k}^{\ell-1} (\lambda + x_i).$$

Then $x = x_{0,n}$ and $x_{k,\ell}x_{\ell,m} = x_{k,m}$.

As always we assume that $n$ is a power of 2. Then the computation of $x$ consists of $k$ stages ($k = \log_2 n$), from 1 to $k$, where at stage $\ell$ we compute all the products $x_{(j-1)2^\ell, j2^\ell}, j = 1, 2, \ldots, 2^{\log_2 n-\ell}$. Each product is computed from the products computed at the previous stage (or already existing at stage 0):

$$x_{(j-1)2^\ell, j2^\ell} = x_{(2j-2)2^{\ell-1}, (2j-1)2^{\ell-1}} x_{(2j-1)2^{\ell-1}, 2j2^{\ell-1}}.$$
\[ x_{0,8} = x \]

\[ x_{0,4} \]

\[ x_{0,2} \]
\[ x_{0,1} \]
\[ \lambda + x_0 \]

\[ x_{1,2} \]
\[ \lambda + x_1 \]

\[ x_{2,3} \]
\[ \lambda + x_2 \]

\[ x_{3,4} \]
\[ \lambda + x_3 \]

\[ x_{4,5} \]
\[ \lambda + x_4 \]

\[ x_{5,6} \]
\[ \lambda + x_5 \]

\[ x_{6,7} \]
\[ \lambda + x_6 \]

\[ x_{7,8} \]
\[ \lambda + x_7 \]
Bookkeeping

- \( \text{deg } x_{(j-1)2^\ell,j2^\ell} = 2^\ell, \ell = 0, 1, \ldots, k - 1 \) and \( j = 1, 2, \ldots, 2^{\log_2 n - \ell} \).

- Therefore, computing \( x_{(j-1)2^\ell,j2^\ell} \) from \( x_{(2j-2)2^{\ell-1},(2j-1)2^{\ell-1}} \) and \( x_{(2j-1)2^{\ell-1},2j2^{\ell-1}} \) can be performed in \( c2^\ell \log_2 2^\ell = c\ell 2^\ell \) operations (what is the constant \( c \)), \( 2^\ell - 1 \) of which are multiplications.

- Thus, the entire stage \( \ell, \ell = 1, 2, \ldots, k \) can be performed in \( c\ell 2^\ell 2^{k-\ell} = c\ell n \) (why?) operations, \( 2^k - 2^{k-\ell} = n - n/2^\ell \) of which are multiplications.

- Finally, the total number of operations is

\[
\sum_{\ell=1}^{\log_2 n - 1} c\ell n = cn \frac{(\log_2 n - 1) \log_2 n}{2} = O(n \log^2 n),
\]

whereas the number of multiplications is

\[
\sum_{\ell=1}^{\log_2 n - 1} \left( n - n/2^\ell \right) < n \log_2 n - n.
\]
Back to multiplication of integers
As usual, \( n = 2^\ell \). Let \( m = \lceil n/\ell \rceil \) and let

\[
x = \sum_{i=0}^{m-1} x_i 2^{i\ell} \quad y = \sum_{j=0}^{m-1} y_j 2^{j\ell},
\]

where \( x_i, y_j < 2^\ell = n \), \( i, j = 0, 1, \ldots, m - 1 \). That is,

\[
x_i = \sum_{t=0}^{\ell-1} a_{i\ell+t} 2^t \quad y_j = \sum_{t=0}^{\ell-1} b_{j\ell+t} 2^t
\]

Then

\[
xy = \sum_{k=0}^{2m-2} \left( \sum_{i+j=k} x_i y_j \right) 2^{k\ell}.
\]
Let
\[ z_k = \sum_{i+j=k} x_i y_j, \quad k = 0, 1, \ldots, 2m - 2. \]

That is, \( z_k \) is the coefficient of \( \lambda^k \) in the polynomial product \( z(\lambda) = x(\lambda)y(\lambda) \), where
\[
\begin{align*}
x(\lambda) &= \sum_{i=0}^{m-1} x_i \lambda^i \\
y(\lambda) &= \sum_{j=0}^{m-1} y_j \lambda^j.
\end{align*}
\]

Then the integer product \( xy \) can be computed from \( \{z_k\}_{k=0,1,\ldots,2m-2} \) in \( 2m - 2 \) additions of shifted copies of \( z_k \)'s as depicted in the next slide.
These additions can be performed in $O(n)$ bit operations.
Now we apply the above multiplication algorithm (over $\mathbb{C}$) to $x(\lambda)$ and $y(\lambda)$, but truncate the roots of 1 at bit-$t$ after the dot.

Let

$$\tilde{z} = (\tilde{z}_0, \tilde{z}_1, \ldots, \tilde{z}_{2m-2})$$

be the approximation of $z$ obtained by the above algorithm. We shall prove that for $t = 7 \log_2 n + 1$,

$$|\tilde{z}_i - z_i| < \frac{1}{2}, \quad i = 0, 1, \ldots, 2m - 2.$$

Thus, we can recover $z$ form $\tilde{z}$.

We postpone for a moment the proof that the above $t$ (what is it?) indeed provides the desired inequality (which one?) and compute the bit complexity of the “truncated” algorithm.
Algorithm 1 for integer multiplication

Substitute the truncated roots of 1 for their counterparts to compute the \( \tilde{z}_i \)s, \( i = 0, 1, \ldots, m - 1 \), by applying the same sequence of computational steps. The number of arithmetical operations is

\[
O(m \log m) = O\left(\frac{n}{\log_2 n} \log \left(\frac{n}{\log_2 n}\right)\right) = O(n).
\]

Each arithmetical operation (addition/subtraction, multiplication by a truncated root of 1, and multiplication of the results of the approximate Fourier transforms) can be performed in \( O(\log^2 n) \) bit operations. Therefore the total number of bit operations is \( O(n \log^2 n) \) (including \( O(n) \) bit operations needed to compute \( xy \) from the \( z_i \)s, \( i = 0, 1, \ldots, m - 1 \)).

Remark Note that the \( \tilde{\omega}_{2^i}, i = 1, 2, \ldots, \log_2 m + 1 \), can be precomputed in \( O(n \log^2 n) \) bit operations (how?).

Algorithm $i + 1$ for integer multiplication

As above, just use Algorithm $i$ for integer multiplication. Then the total number of bit operations is

$$O(n \log n \log \log n \cdots \log \cdots \log n (\log \cdots \log n)^2).$$
Searching for $t$
We shall need the following notation.

- \( \varepsilon = 2^{-t} \).
- \( \tilde{\omega}_2^i \) and \( \tilde{\omega}_2^{-1} \) denote \( \omega_2^i \) and \( \omega_2^{-1} \) truncated at bit-\( t \), respectively, \( i = 1, 2, \ldots, \log_2 m + 1 \).

Thus, \( |\tilde{\omega}_2^i - \omega_2^i| \leq \varepsilon \) and \( |\tilde{\omega}_2^{-1} - \omega_2^{-1}| \leq \varepsilon \), \( i = 0, 1, \ldots, \log_2 m + 1 \).

- \( \hat{x}^T = (\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_{2m-2})^T = W(\omega_{2m-1})x^T \) and \( \tilde{x}^T = (\tilde{x}_0, \tilde{x}_1, \ldots, \tilde{x}_{2m-2})^T \) is the result of the “truncated” transformation of \( x \).

- \( \hat{y}^T = (\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{2m-2})^T = W(\omega_{2m-1})y^T \) and \( \tilde{y}^T = (\tilde{y}_0, \tilde{y}_1, \ldots, \tilde{y}_{2m-2})^T \) is the result of the “truncated” transformation of \( y \).

- \( \hat{z} = (\hat{z}_0, \hat{z}_1, \ldots, \hat{z}_{2m-2}) = \hat{x} \cdot \hat{y} \) and \( \tilde{z} = (\tilde{z}_0, \tilde{z}_1, \ldots, \tilde{z}_{2m-2}) = \tilde{x} \cdot \tilde{y} \).

- \( \tilde{z} = (\tilde{z}_0, \tilde{z}_1, \ldots, \tilde{z}_{2m-2}) \) is the result of the “reversal truncated” transformation of \( \tilde{z} \).

Recall that we need an \( \varepsilon \) such that \( |\tilde{z}_i - z_i| < \frac{1}{2} \).
Lemma  Let $M \geq 1$, $f = \sum_{i=0}^{2^k-1} f_i \lambda^i$, $|f_i| \leq M$, $i = 0, 1, \ldots, 2^k - 1$, $|\omega_{2i} - \omega_2| \leq \varepsilon$, $i = 1, 2, \ldots, k$, and let $(f(\omega_{2^k}^0), f(\omega_{2^k}^1), \ldots, f(\omega_{2^k}^{k-1}))$ be the result of the “truncated” transformation of $f$. Then, for all $j = 0, 1, \ldots, 2^k - 1$,

$$\left| f\left(\omega_{2^k}^j\right) - f\left(\omega_{2^k}^j\right) \right| \leq 4^k M \varepsilon.$$  

\(^1\)Note that $f(\omega_{2^k}^j)$ is not the value of $f$ at $\omega_{2^k}^j$ (why?), but is just a notation.
Proof The proof is by induction on $k$. The basis $k = 0$ is immediate (why), and for the induction step, assume that the inequality holds for $k$. Then

$$\left| f \left( \omega_{2k+1}^j \right) - f \left( \omega_{2k+1}^j \right) \right|$$

$$\leq \left| f_0 \left( \omega_{2k}^j \right) - f_0 \left( \omega_{2k}^j \right) \right| + \left| \omega_{2k} f_1 \left( \omega_{2k}^j \right) - \omega_{2k} f_1 \left( \omega_{2k}^j \right) \right|$$

$$\leq 4^k M \varepsilon + \left| \omega_{2k} f_1 \left( \omega_{2k}^j \right) - \omega_{2k} f_1 \left( \omega_{2k}^j \right) \right| + \left| \omega_{2k} f_1 \left( \omega_{2k}^j \right) - \omega_{2k} f_1 \left( \omega_{2k}^j \right) \right|$$

$$\leq 4^k M \varepsilon + (1 + \varepsilon) 4^k M \varepsilon + 4^k M \varepsilon$$

$$\leq (4^k + 2 \cdot 4^k + 2^k) M \varepsilon$$

$$\leq 4^{k+1} M \varepsilon.$$
Next, by the lemma with $f$ being $x$ and $y$ (in both cases, $k = \log_2 m+1$),

$$\left| \tilde{x}_i \tilde{y}_i - \hat{x}_i \hat{y}_i \right| \leq \left| \tilde{x}_i \tilde{y}_i - \hat{x}_i \hat{y}_i \right| + \left| \tilde{x}_i \hat{y}_i - \hat{x}_i \hat{y}_i \right|$$

$$\leq \left| \tilde{x}_i \right| \left| \tilde{y}_i - \hat{y}_i \right| + \left| \hat{y}_i \right| \left| \tilde{x}_i - \hat{x}_i \right|$$

$$\leq (2mM + 4^{\log_2 m+1} M \varepsilon) \left( 4^{\log_2 m+1} M \varepsilon \right) + (2mM) \left( 4^{\log_2 m+1} M \varepsilon \right)$$

$$= (2mM + 4m^2 M \varepsilon) \left( 4m^2 M \varepsilon \right) + (2mM) \left( 4m^2 M \varepsilon \right)$$

$$\leq 40m^4 M^2 \varepsilon.$$
Finally,

\[ |\tilde{z}_i - z_i| = \left| \tilde{z} \left( \omega_{2m}^i \right) - \tilde{z} \left( \omega_{2m}^i \right) \right| \]
\[ \leq \left| \tilde{z} \left( \omega_{2m}^i \right) - \tilde{z} \left( \omega_{2m}^i \right) \right| + \left| \tilde{z} \left( \omega_{2m}^i \right) - \tilde{z} \left( \omega_{2m}^i \right) \right| \]
\[ \leq \frac{1}{2m} 4m^2 \left( m \left( M^2 + 40m^4 M^2 \varepsilon \right) \right) \varepsilon + \frac{1}{2m} 2m \cdot 40m^4 M^2 \varepsilon \]
\[ \leq 46m^4 M^2 \varepsilon \]
\[ \leq n^7 \varepsilon , \]

because, in our case \( M = n \) (why?).

Therefore, \( \varepsilon \leq \frac{1}{2n^7} \) would suffice. That is, we may truncate at the \( 7 \log_2 n \) bit.