

Assignment Project Exam Help

Algorithms

https://powcoder.com

Aleks Ignjatović

Add We Chat powcoder

University of New South Wales

5. THE FAST FOURIER TRANSFORM (not examinable material)

4□ > 4□ > 4□ > 4□ > 4□ > 900

• Given two polynomials of degree at most n,

Assignment Project Exam Help

https://powcoder.com

2 multiply them point by point using 2n + 1 multiplications:

$\left\{ (x_0, \underbrace{PAdd_0WeChat}_{P_C(x_1)}powcoder_n) \right\}$

6 Convert such value representation of $P_C(x)$ to its coefficient form

$$P_C(x) = C_{2n}x^{2n} + C_{2n-1}x^{2n-1} + \dots + C_1x + C_0;$$

4□ > 4□ > 4□ > 4□ > 4□ > 4□ > 4□

• Given two polynomials of degree at most n,

Assignment Project Exam Help

https://powcoder.com(x2n))}

2 multiply them point by point using 2n + 1 multiplications:

$\{(x_0, PAdd_0)WeChat powcoder_n)\}$

3 Convert such value representation of $P_C(x)$ to its coefficient form

$$P_C(x) = C_{2n}x^{2n} + C_{2n-1}x^{2n-1} + \ldots + C_1x + C_0;$$

• Given two polynomials of degree at most n,

Assignment Project Example 19
$$x_0, x_1, \dots, x_{2n}$$
:
 $P_B(x) = B_n x^n + \dots + B_0$
 $E_B(x) = B_n x^n +$

$$\mathbf{htt}_{B}^{P}(\mathbf{x}) \overset{\leftarrow}{\smile} ((\mathbf{p}, \mathbf{p}_{B}^{A}(\mathbf{x}_{0})) \overset{\leftarrow}{\smile} (\mathbf{p}, \mathbf{p}_{B}^{A}(\mathbf{x}_$$

{ Add WeChat powcoder }

$$P_C(x) = C_{2n}x^{2n} + C_{2n-1}x^{2n-1} + \ldots + C_1x + C_0;$$

COMP3121/9101

• Given two polynomials of degree at most n,

Assignment Project Exame Help
$$x_0, x_1, \dots, x_{2n}$$
:

$$\mathbf{htt}_{B}^{P}(\mathbf{x}) \overset{\leftrightarrow}{\smile} \underbrace{\{(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0}))\}}_{\mathbf{x}_{0}} \underbrace{\{(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0}))\}}_{\mathbf{x}_{0}}$$

2 multiply them point by point using 2n + 1 multiplications:

$$\left\{(x_0, \underbrace{PAQQ_0}_{P_C(x_0)} \underbrace{WxeQ(xbrat_1)}_{P_C(x_1)} \underbrace{POWeQ(x_1)}_{P_C(x_{2n})})\right\}$$

 \odot Convert such value representation of $P_C(x)$ to its coefficient form

$$P_C(x) = C_{2n}x^{2n} + C_{2n-1}x^{2n-1} + \dots + C_1x + C_0;$$

• Given two polynomials of degree at most n,

Assignment Project Example 19
$$x_0, x_1, \dots, x_{2n}$$
:
 $P_B(x) = B_n x^n + \dots + B_0$
 $E_B(x) = B_n x^n +$

$$\mathbf{htt}_{B}^{P}(\mathbf{x}) \overset{\leftrightarrow}{\smile} \underbrace{\{(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0}))\}}_{\mathbf{x}_{0}} \underbrace{\{(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0})),(\mathbf{x}_{0},P_{A}(x_{0}))\}}_{\mathbf{x}_{0}}$$

2 multiply them point by point using 2n + 1 multiplications:

$$\left\{(x_0, \underbrace{PAC(x_0)}_{P_C(x_0)} \underbrace{WxeC(x)Pat_1}_{P_C(x_1)} \underbrace{POW, \underline{CaOPet_{2n}}}_{P_C(x_{2n})})\right\}$$

3 Convert such value representation of $P_C(x)$ to its coefficient form

$$P_C(x) = C_{2n}x^{2n} + C_{2n-1}x^{2n-1} + \dots + C_1x + C_0;$$

• So, we need 2n + 1 values of $P_A(x_i)$ and $P_B(x_i)$, $0 \le i \le 2n$.

Assignment Project Exam Help

 $-n, -(n-1), \ldots, -1, 0, 1, \ldots, n-1, r$

https://powcoder.com

- We saw that the trouble is that, as the degree n of the polynomials $P_A(x)$ and $P_B(x)$ increases the value of n^n increases very fast and causes rapid increase of the Add on W increases $P_A(x)$ that is powcoder $P_A(x)$ increases $P_A(x)$ increa
- **Key Question:** What values should we take for x_0, \ldots, x_{2n} to avoid "explosion" of size when we evaluate x_i^n while computing $P_A(x_i) = A_0 + A_1x + \ldots + A_nx_i^n$?



• So, we need 2n + 1 values of $P_A(x_i)$ and $P_B(x_i)$, $0 \le i \le 2n$.

Assignment Project Exam Help

 $-n, -(n-1), \ldots, -1, 0, 1, \ldots, n-1, n$

- We saw that the trouble is that, as the degree n of the polynomials $P_A(x)$ and $P_B(x)$ increases the value of n^n increases very fast and causes rapid increase of the Add on W increases Add the Add Add the Add Add
- **Key Question:** What values should we take for $x_0, ..., x_{2n}$ to avoid "explosion" of size when we evaluate x_i^n while computing $P_A(x_i) = A_0 + A_1x + ... + A_nx_i^n$?



• So, we need 2n + 1 values of $P_A(x_i)$ and $P_B(x_i)$, $0 \le i \le 2n$.

Assignment Project Exam Help

 $-n, -(n-1), \ldots, -1, 0, 1, \ldots, n-1, n$

- We saw that the trouble is that, as the degree n of the polynomials $P_A(x)$ and $P_B(x)$ increases, the value of n^n increases very fast and causes rapid increase of the computational complexity of the elegation for polynomial multiplication which we used in the generalised Katatsuba algorithm.
- **Key Question:** What values should we take for $x_0, ..., x_{2n}$ to avoid "explosion" of size when we evaluate x_i^n while computing $P_A(x_i) = A_0 + A_1x + ... + A_nx_i^n$?



• So, we need 2n + 1 values of $P_A(x_i)$ and $P_B(x_i)$, $0 \le i \le 2n$.

Assignment Project Exam Help

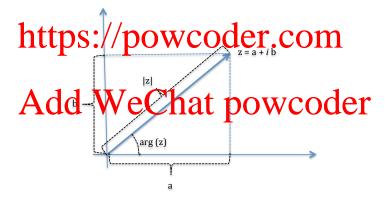
$$-n, -(n-1), \ldots, -1, 0, 1, \ldots, n-1, n$$

- We saw that the trouble is that, as the degree n of the polynomials $P_A(x)$ and $P_B(x)$ increases, the value of n^n increases very fast and causes rapid increase of the computational complexity of the elegation for polynomial multiplication which we used in the generalised Katatsuba algorithm.
- **Key Question:** What values should we take for $x_0, ..., x_{2n}$ to avoid "explosion" of size when we evaluate x_i^n while computing $P_A(x_i) = A_0 + A_1x + ... + A_nx_i^n$?



Complex numbers revisited

Complex numbers z=a+ib can be represented using their modulus $|z|=\sqrt{a^2+b^2}$ and their argument, $\arg(z)$, which is an angle taking values in $(-\pi,\pi]$ and satisfying:



Complex numbers revisited

Recall that

Assignment Project Exama Help see the figure.

https://powcoder.com

Add Wechat powcoder

- Roots of unity of order n are complex numbers which satisfy $z^n = 1$.
- If $z^n = |z|^n(\cos(n\arg(z)) + i\sin(n\arg(z))) = 1$ then

Assignment Project Exam Help

- Thus, $n \arg(z) = 2\pi k$, i.e., $\arg(z) = \frac{2\pi k}{n}$
- We that $\omega_n = e^{i 2\pi n}$, such ω_n is called a rimitive root of unity of order n. https://powcoder.com $\omega_n = e^{i 2\pi / n} \qquad (\omega_{16})^5 \qquad (\omega_{16})^4 (\omega_{16})^3$ $\omega_n = e^{i 2\pi / n} \qquad (\omega_{16})^5 \qquad (\omega_{16})^2$ $\omega_n = e^{i 2\pi / n} \qquad (\omega_{16})^5 \qquad (\omega_{16})^2$ $\omega_n = e^{i 2\pi / n} \qquad (\omega_{16})^5 \qquad (\omega_{16})^3$ $\omega_n = e^{i 2\pi / n} \qquad (\omega_{16})^5 \qquad (\omega_{16})^3$

• A root of unity ω of order n is *primitive* if all other roots of unity of the same order can be obtained as its powers ω^k .

Roots of unity of order 16

- Roots of unity of order n are complex numbers which satisfy $z^n = 1$.
- If $z^n = |z|^n (\cos(n \arg(z)) + i \sin(n \arg(z))) = 1$ then

Assignment Project Exam Help

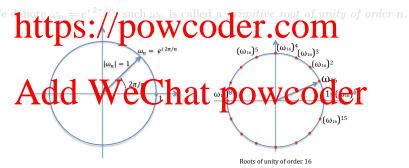
- Thus, $n \arg(z) = 2\pi k$, i.e., $\arg(z) = \frac{2\pi k}{m}$
- We that $\omega_n = e^{i 2\pi/n}$ such ω_n is called a trimitive root of unity of order n. https://powcoder.com $(\omega_{16})^5 \qquad (\omega_{16})^3 \qquad (\omega_{16})^2$ Add We'Chat ω_{16} powcoder. $(\omega_{16})^5 \qquad (\omega_{16})^3 \qquad (\omega_{16})^2$ $(\omega_{16})^{15}$
- A root of unity ω of order n is *primitive* if all other roots of unity of the same order can be obtained as its powers ω^k .

Roots of unity of order 16

- Roots of unity of order n are complex numbers which satisfy $z^n = 1$.
- If $z^n = |z|^n (\cos(n \arg(z)) + i \sin(n \arg(z))) = 1$ then

Assignment Project Exam Help

• Thus, $n \arg(z) = 2\pi k$, i.e., $\arg(z) = \frac{2\pi k}{n}$

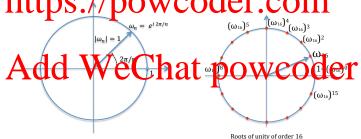


• A root of unity ω of order n is *primitive* if all other roots of unity of the same order can be obtained as its powers ω^k .

- Roots of unity of order n are complex numbers which satisfy $z^n = 1$.
- If $z^n = |z|^n (\cos(n \arg(z)) + \underline{i} \sin(n \arg(z))) = 1$ then

Assignment Project Exam Help

- Thus, $n \arg(z) = 2\pi k$, i.e., $\arg(z) = \frac{2\pi k}{n}$
- We denote $\omega_n = e^{i 2\pi/n}$, such ω_n is called a *primitive root of unity of order n*. **NUTS:**//**DOWCOGET.COM**

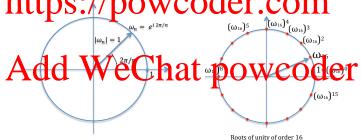


• A root of unity ω of order n is *primitive* if all other roots of unity of the same order can be obtained as its powers ω^k .

- Roots of unity of order n are complex numbers which satisfy $z^n = 1$.
- If $z^n = |z|^n (\cos(n \arg(z)) + \underline{i} \sin(n \arg(z))) = 1$ then

Assignment Project Exam Help

- Thus, $n \arg(z) = 2\pi k$, i.e., $\arg(z) = \frac{2\pi k}{n}$
- We denote $\omega_n = e^{i 2\pi/n}$, such ω_n is called a *primitive root of unity of order n*.



• A root of unity ω of order n is *primitive* if all other roots of unity of the same order can be obtained as its powers ω^k .

• For $\omega_n = e^{i 2\pi/n}$ and for all k such that $0 \le k \le n-1$,

- Thus, $\omega_n^k = (\omega_n)^k$ is also a root of unity (and it can be shown that it is primhttps://powcoder.com
- Since ω_n are roots of timey for $k=0,1,\ldots,n-1$ and there are at most n distinct roots of unity of order n (i.e., solutions to the equation $x^n-1=0$) we conclude that every root of unity of order n must be of the form ω_n^k .
- * Add WeChat powcoder
- If $k+m \ge n$ then k+m=n+l for $l=(k+m) \bmod n$ and we have $\omega_n^k \omega_n^m = \omega_n^{k+m} = \omega_n^{n+l} = \omega_n^n \omega_n^l = 1 \cdot \omega_n^l = \omega_n^l$ where $0 \le l < n$.
- Thus, the product of any two roots of unity of the same order is just another root of unity of the same order.



• For $\omega_n = e^{i 2\pi/n}$ and for all k such that $0 \le k \le n-1$,

- Thus, $\omega_n^k = (\omega_n)^k$ is also a root of unity (and it can be shown that it is primitive just in case/k/is relatively prime with n). Since ω_n and ω_n is also a root of unity (and it can be shown that it is primitive just in case/k/is relatively prime with n).
- Since ω_n and roots of an y for $k=0,1,\ldots,n-1$ and there are at most n distinct roots of unity of order n (i.e., solutions to the equation $x^n-1=0$) we conclude that every root of unity of order n must be of the form ω_n^k .
- * Add WeChat powcoder
- If $k+m \ge n$ then k+m=n+l for $l=(k+m) \bmod n$ and we have $\omega_n^k \omega_n^m = \omega_n^{k+m} = \omega_n^{n+l} = \omega_n^n \omega_n^l = 1 \cdot \omega_n^l = \omega_n^l$ where $0 \le l < n$.
- Thus, the product of any two roots of unity of the same order is just another root of unity of the same order.



• For $\omega_n = e^{i 2\pi/n}$ and for all k such that $0 \le k \le n-1$,

- Thus, ω_n^k = (ω_n)^k is also a root of unity (and it can be shown that it is primitive just in case/k/is relatively prime with n
 Since ω_n^k are roots of unity for k = 0,1,...,n-1 and there are at most n
- Since ω_n^k are roots of unity for $k = 0, 1, \dots, n-1$ and there are at most n distinct roots of unity of order n (i.e., solutions to the equation $x^n 1 = 0$) we conclude that every root of unity of order n must be of the form ω_n^k .
- * Add WeChat powcoder
- If $k+m \ge n$ then k+m=n+l for $l=(k+m) \bmod n$ and we have $\omega_n^k \omega_n^m = \omega_n^{k+m} = \omega_n^{n+l} = \omega_n^n \omega_n^l = 1 \cdot \omega_n^l = \omega_n^l$ where $0 \le l < n$.
- Thus, the product of any two roots of unity of the same order is just another root of unity of the same order.



• For $\omega_n = e^{i 2\pi/n}$ and for all k such that $0 \le k \le n-1$,

- Thus, $\omega_n^k = (\omega_n)^k$ is also a root of unity (and it can be shown that it is primitive just in case k is relatively prime with n).
- Since ω_n^k are roots of unity for $k=0,1,\ldots,n-1$ and there are at most n distinct roots of unity of order n (i.e., solutions to the equation $x^n-1=0$) we conclude that every root of unity of order n must be of the form ω_n^k .
- For the product of the vector of unity ω^k and ω^m of the same order we have ω^m of the same order we
- If $k+m \ge n$ then k+m=n+l for $l=(k+m) \bmod n$ and we have $\omega_n^k \omega_n^m = \omega_n^{k+m} = \omega_n^{n+l} = \omega_n^n \omega_n^l = 1 \cdot \omega_n^l = \omega_n^l$ where $0 \le l < n$.
- Thus, the product of any two roots of unity of the same order is just another root of unity of the same order.



• For $\omega_n = e^{i 2\pi/n}$ and for all k such that $0 \le k \le n-1$,

- Thus, $\omega_n^k = (\omega_n)^k$ is also a root of unity (and it can be shown that it is primitive just in case/k/is relatively prime with n).
- Since ω_n^k are roots of unity for $k=0,1,\ldots,n-1$ and there are at most n distinct roots of unity of order n (i.e., solutions to the equation $x^n-1=0$) we conclude that every root of unity of order n must be of the form ω_n^k .
- For the product of the we posts of unity ω^k and ω^m of the same order we have e_n of the same order we
- If $k+m \ge n$ then k+m=n+l for $l=(k+m) \bmod n$ and we have $\omega_n^k \omega_n^m = \omega_n^{k+m} = \omega_n^{n+l} = \omega_n^n \omega_n^l = 1 \cdot \omega_n^l = \omega_n^l$ where $0 \le l < n$.
- Thus, the product of any two roots of unity of the same order is just another root of unity of the same order.



• For $\omega_n = e^{i 2\pi/n}$ and for all k such that $0 \le k \le n-1$,

- Thus, $\omega_n^k = (\omega_n)^k$ is also a root of unity (and it can be shown that it is primitive just in case k is relatively prime with n).
- Since ω_n^k are roots of unity for $k=0,1,\ldots,n-1$ and there are at most n distinct roots of unity of order n (i.e., solutions to the equation $x^n-1=0$) we conclude that every root of unity of order n must be of the form ω_n^k .
- For the product of the we roots of unity ω^k and ω^m of the same order we have $c_n u = C_k^{k+m}$
- If $k+m \ge n$ then k+m=n+l for $l=(k+m) \bmod n$ and we have $\omega_n^k \omega_n^m = \omega_n^{k+m} = \omega_n^{n+l} = \omega_n^n \omega_n^l = 1 \cdot \omega_n^l = \omega_n^l$ where $0 \le l < n$.
- Thus, the product of any two roots of unity of the same order is just another root of unity of the same order.



• So in the set of all roots of unity of order n, i.e., $\{1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}\}$ we can multiply any two elements or raise an element to any power without going out of this set.

Assignment Project Exam Help

A most important property of the roots of unity is:

The https://powcoder.com

Add WeChat powcoder

- Thus, in particular, $(\omega_{2n}^k)^2 = \omega_{2n}^{2k} = (\omega_{2n}^2)^k = \omega_n^k$.
- So the squares of the roots of unity of order 2n are just the roots of unity of order n.

• So in the set of all roots of unity of order n, i.e., $\{1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}\}$ we can multiply any two elements or raise an element to any power without going out of this set.

Significant facilities since and of intest p

https://powcoder.com

Proof

Add WeChat powcoder

- Thus, in particular, $(\omega_{2n}^k)^2 = \omega_{2n}^{2k} = (\omega_{2n}^2)^k = \omega_n^k$.
- So the squares of the roots of unity of order 2n are just the roots of unity of order n.

• So in the set of all roots of unity of order n, i.e., $\{1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}\}$ we can multiply any two elements or raise an element to any power without going out of this set.

SSIeganlish ethte for adding ecte sExtraint of untest p

• A most important property of the roots of unity is:

The latter is Lempa we code to common the latter is the latter is the latter in the latter in the latter is the latter in the latter is the latter in the lat

Proof

Add WeChat powcoder

- Thus, in particular, $(\omega_{2n}^k)^2 = \omega_{2n}^{2k} = (\omega_{2n}^2)^k = \omega_n^k$.
- So the squares of the roots of unity of order 2n are just the roots of unity of order n.

• So in the set of all roots of unity of order n, i.e., $\{1, \omega_n, \omega_n^2, \ldots, \omega_n^{n-1}\}$ we can multiply any two elements or raise an element to any power without going out of this set.

Solegan list et the for adding e.G. s. Extraint of untest p

• A most important property of the roots of unity is:

The latter is Lemma We Code Tege Comm

Proof:

Addwwweenhat*poweodern

- Thus, in particular, $(\omega_{2n}^k)^2 = \omega_{2n}^{2k} = (\omega_{2n}^2)^k = \omega_n^k$.
- So the squares of the roots of unity of order 2n are just the roots of unity of order n.

• So in the set of all roots of unity of order n, i.e., $\{1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}\}$ we can multiply any two elements or raise an element to any power without going out of this set.

Solganistic that fraction extra sum of unterportation of unity!

• A most important property of the roots of unity is:

The latters Lendow Coder Com

Proof:

Add white the hat power of the final and the second series and the second series are the

- Thus, in particular, $(\omega_{2n}^k)^2 = \omega_{2n}^{2k} = (\omega_{2n}^2)^k = \omega_n^k$.
- So the squares of the roots of unity of order 2n are just the roots of unity of order n.



• So in the set of all roots of unity of order n, i.e., $\{1, \omega_n, \omega_n^2, \dots, \omega_n^{n-1}\}$ we can multiply any two elements or raise an element to any power without going out of this set.

SSIGNIFICATION OF THE PARTITION OF THE SPECIAL STREET OF THE SPECI

• A most important property of the roots of unity is:

The attps://powcoder.com

Proof:

$A^{k} d d \omega_{kn} W = C^{\frac{2\pi}{kn}} hat^{\frac{2\pi km}{k}} po^{\frac{i2\pi m}{k}} coder^n$

- Thus, in particular, $(\omega_{2n}^k)^2 = \omega_{2n}^{2k} = (\omega_{2n}^2)^k = \omega_n^k$.
- So the squares of the roots of unity of order 2n are just the roots of unity of order n.



• Let $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$ be a sequence of n real or complex numbers.

- We can evaluate it at all complex roots of unity of order n, i.e., we compute $P_A(\omega_n^k)$ for all $0 \le k \le n-1$.
- The https://powcoder.com scalled the Discrete Fourier Transform (DFT) of the sequence $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$.
- The Add We Chat powcoder $\widehat{A} = \langle \widehat{A}_0, \widehat{A}_1, \dots, \widehat{A}_{n-1} \rangle$.
- The DFT \widehat{A} of a sequence A can be computed VERY FAST using a divide-and-conquer algorithm called the **Fast Fourier Transform**.



• Let $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$ be a sequence of n real or complex numbers.

- We can evaluate it at all complex roots of unity of order n, i.e., we compute $P_A(\omega_n^k)$ for all $0 \le k \le n-1$.
- The https://powcoder.com's called the Discrete Fourier Transform (DFT) of the sequence $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$.
- The Add We Chat powcoder $\widehat{A} = \langle \widehat{A}_0, \widehat{A}_1, \dots, \widehat{A}_{n-1} \rangle$.
- The DFT \widehat{A} of a sequence A can be computed VERY FAST using a divide-and-conquer algorithm called the **Fast Fourier Transform**.



• Let $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$ be a sequence of n real or complex numbers.

- We can evaluate it at all complex roots of unity of order n, i.e., we compute $P_A(\omega_n^k)$ for all $0 \le k \le n-1$.
- The https://powcoder.com scalled the Discrete Fourier Transform (DFT) of the sequence $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$.
- The Add We Chat powcoder $\widehat{A} = \langle \widehat{A}_0, \widehat{A}_1, \dots, \widehat{A}_{n-1} \rangle$.
- The DFT \widehat{A} of a sequence A can be computed VERY FAST using a divide-and-conquer algorithm called the **Fast Fourier Transform**.



• Let $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$ be a sequence of n real or complex numbers.

- We can evaluate it at all complex roots of unity of order n, i.e., we compute $P_A(\omega_n^k)$ for all $0 \le k \le n-1$.
- The latter Salves (P.O., W.C.), P.A. e. T. C.O. is called the Discrete Fourier Transform (DFT) of the sequence $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$.
- The Add We Chat powcoder $\widehat{A} = \langle \widehat{A}_0, \widehat{A}_1, \dots, \widehat{A}_{n-1} \rangle$.
- The DFT \widehat{A} of a sequence A can be computed VERY FAST using a divide-and-conquer algorithm called the **Fast Fourier Transform**.



• Let $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$ be a sequence of n real or complex numbers.

- We can evaluate it at all complex roots of unity of order n, i.e., we compute $P_A(\omega_n^k)$ for all $0 \le k \le n-1$.
- The Latte Salves (P.O., W.C., O.d., et al. (O.T.) is called the Discrete Fourier Transform (DFT) of the sequence $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$.
- The value $A(a_n^k)$ is usually denoted by \widehat{A}_k and herefore \widehat{A}_k and here \widehat{A}_k and here \widehat{A}_k is usually denoted by $\widehat{A} = \langle \widehat{A}_0, \widehat{A}_1, \dots, \widehat{A}_{n-1} \rangle$.
- The DFT \widehat{A} of a sequence A can be computed VERY FAST using a divide-and-conquer algorithm called the **Fast Fourier Transform**.



• Let $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$ be a sequence of n real or complex numbers.

- We can evaluate it at all complex roots of unity of order n, i.e., we compute $P_A(\omega_n^k)$ for all $0 \le k \le n-1$.
- The Latte Salves (P.O., W.C., P.C., P.T., C.O.) is called the Discrete Fourier Transform (DFT) of the sequence $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$.
- The value (A_{n}^{k}) is the denoted of (A_{n}^{k}) is usually denoted by $\widehat{A} = \langle \widehat{A}_{0}, \widehat{A}_{1}, \dots, \widehat{A}_{n-1} \rangle$.
- The DFT \widehat{A} of a sequence A can be computed VERY FAST using a divide-and-conquer algorithm called the **Fast Fourier Transform**.



New way for fast multiplication of polynomials

• If we multiply a polynomial

Assignment Pa(p)=A0+...+An Exam Help

$$P_B(x) = B_0 + \ldots + B_{m-1}x^{m-1}$$
 of denoted by Sweet Bowe Goder. Com
$$C(x) = P_A(x)P_B(x) = C_0 + \ldots + C_{m+n-2}x^{m+n-2}$$

of degree dd + We Chat bowcoder

- To uniquely determine such a polynomial C(x) of degree m+n-2 we need m+n-1 many values.
- Thus, we will evaluate both $P_A(x)$ and $P_B(x)$ at all the roots of unity of order n+m-1 (instead of at $-(n-1),\ldots,-1,0,1,\ldots,m-1$ as we would in Karatsuba's method!)

• If we multiply a polynomial

Assignment Pa(p)=A₀+...+A_n Eⁿ⁻¹ Exam Help

$$P_B(x) = B_0 + \ldots + B_{m-1}x^{m-1}$$
 of denoted by $P_B(x) = P_A(x)P_B(x) = C_0 + \ldots + C_{m+n-2}x^{m+n-2}$

of degree dd + We Chat bowcoder

- To uniquely determine such a polynomial C(x) of degree m+n-2 we need m+n-1 many values.
- Thus, we will evaluate both $P_A(x)$ and $P_B(x)$ at all the roots of unity of order n+m-1 (instead of at $-(n-1),\ldots,-1,0,1,\ldots,m-1$ as we would in Karatsuba's method!)

• If we multiply a polynomial

Assignment Pa(r) = A₀ + ... + A_n + r = Exam Help

$$P_B(x) = B_0 + \ldots + B_{m-1}x^{m-1}$$
 of denoted by $P_B(x) = P_A(x)P_B(x) = C_0 + \ldots + C_{m+n-2}x^{m+n-2}$

of degree dd + We Chat bowcoe die r

- To uniquely determine such a polynomial C(x) of degree m+n-2 we need m+n-1 many values.
- Thus, we will evaluate both $P_A(x)$ and $P_B(x)$ at all the roots of unity of order n+m-1 (instead of at $-(n-1),\ldots,-1,0,1,\ldots,m-1$ as we would in Karatsuba's method!)

A Sortegorian point of the DET of a sequence of length n as the values of the Sortegorian point of the Hoxs a inity of the point n as the values of the Loxs n inity of the point n in n of the n point n in n in

- So the DFT of a sequence A is another sequence \widehat{A} of exactly the same length; we do not have an operation which would evaluate a polynomial of degree
- For https://powcoder.com $_{P_B(x)}$, we pad A with m-1 zeros at the end, $(A_0,A_1,\ldots,A_{n-1},0,\ldots,0)$ to make it of
 - (Bo, Add WeChat powcoder
- Note that this does not change the associated polynomials because the added higher powers have the corresponding coefficients equal to zero.

Note that we defined the DET of a sequence of length n as the values of the S or equality partial of the length n as the values of the length n and n are values of n and n are v

- So the DFT of a sequence A is another sequence \widehat{A} of exactly the same length; we do not have an operation which would evaluate a polynomial of degree n-1 at m roots of white of order $m \neq n$
- For the proof of white of order $m \neq n$ derived a with m-1 zeros at the end, $(A_0,A_1,\ldots,A_{n-1},0,\ldots,0)$ to make it of
 - (Bo, Add 1, WeChat powcoder -
- Note that this does not change the associated polynomials because the added higher powers have the corresponding coefficients equal to zero.

A Sortemonia point of a sequence of length n as the values of the Sortemonia point of the length n as the values of the sortemonia point of the length n as the values of the length n and n as the values of the length n as the values of the length n and n as the values of the length n as the values of the length n as the values of the length n and n as the values of the length n and n as the values of the length n and n as the values of the length n and n and n are values of the length n and n are values of the length n and n are values of n and n are va

- So the DFT of a sequence A is another sequence \widehat{A} of exactly the same length; we do not have an operation which would evaluate a polynomial of degree n-1 at purposes of whity of order $m \neq n$
- For that reason, since we need n+n. We that of $P_{\bullet}(x)-P_{A}(x)P_{B}(x)$, we pad A with m-1 zeros at the end, $(A_{0},A_{1},\ldots,A_{n-1},\underbrace{0,\ldots,0}_{m-1})$ to make it of

 Note that this does not change the associated polynomials because the added higher powers have the corresponding coefficients equal to zero.

Note that we defined the DFT of a sequence of length n as the values of the Sortegiological points of the results of the re

- So the DFT of a sequence A is another sequence \widehat{A} of exactly the same length; we do not have an operation which would evaluate a polynomial of degree n-1 at m roots of white of order $m \neq n$
- For that reason since we need n+n-1 value of $P_{\bullet}(x)-P_{A}(x)P_{B}(x)$, we pad A with m-1 zeros at the end, $(A_{0},A_{1},\ldots,A_{n-1},\underbrace{0,\ldots,0}_{m-1})$ to make it of

```
length n+n-1, and similarly we pad B with n-1 zeros at the end, (B_0, H_1, \dots, H_{n-1}, \dots, H_{n-1}) of all obtain B with B and B and B with B and B with B and B are all B and B and B and B are all B and B and B and B are all B are all B and B are all B and B are all B and B are all B are all B and B are all B are all B and B are all B and
```

• Note that this does not change the associated polynomials because the added higher powers have the corresponding coefficients equal to zero.

• We can now compute the DFTs of the two (0 padded) sequences:

Assignment Project Exam Help

and

$$https://powcoder.com$$

• For each k we multiply the corresponding values $\widehat{A}_k = P_A(\omega_{n+m-1}^k)$ and $\widehat{B}_k = P_B(\omega_{n+m-1}^k)$, thus obtaining

Add WeChat powcoder

• We then use the inverse transformation for DFT, called IDFT, to recover the coefficients $\langle C_0, C_1, \ldots, C_{n+m-1} \rangle$ of the product polynomial $P_C(x)$ from the sequence $\langle \widehat{C}_0, \widehat{C}_1, \ldots, \widehat{C}_{n+m-1} \rangle$ of its values $C_k = P_C(\omega_{n+m-1}^k)$ at the roots of unity of order n+m-1.

• We can now compute the DFTs of the two (0 padded) sequences:

Assignment Project Exam Help

and

$$\begin{array}{l} \textit{DFT}(\langle B_0, B_1, \dots, B_{m-1}, \underbrace{0, \dots, 0} \rangle) = \langle \widehat{B}_0, \widehat{B}_1, \dots, \widehat{B}_{n+m-2}) \\ \textit{https://powcoder.com} \end{array}$$

• For each k we multiply the corresponding values $\widehat{A}_k = P_A(\omega_{n+m-1}^k)$ and $\widehat{B}_k = P_B(\omega_{n+m-1}^k)$, thus obtaining

Add a We Chat poweder

• We then use the inverse transformation for DFT, called IDFT, to recover the coefficients $\langle C_0, C_1, \ldots, C_{n+m-1} \rangle$ of the product polynomial $P_C(x)$ from the sequence $\langle \widehat{C}_0, \widehat{C}_1, \ldots, \widehat{C}_{n+m-1} \rangle$ of its values $C_k = P_C(\omega_{n+m-1}^k)$ at the roots of unity of order n+m-1.

• We can now compute the DFTs of the two (0 padded) sequences:

Assignment Project Exam Help

and

$$\begin{array}{l} \textit{DFT}(\langle B_0, B_1, \dots, B_{m-1}, \underbrace{0, \dots, 0} \rangle) = \langle \widehat{B}_0, \widehat{B}_1, \dots, \widehat{B}_{n+m-2}) \\ \textit{https://powcoder.com} \end{array}$$

• For each k we multiply the corresponding values $\widehat{A}_k = P_A(\omega_{n+m-1}^k)$ and $\widehat{B}_k = P_B(\omega_{n+m-1}^k)$, thus obtaining

Add AW-eChat poweoder

• We then use the inverse transformation for DFT, called IDFT, to recover the coefficients $\langle C_0, C_1, \ldots, C_{n+m-1} \rangle$ of the product polynomial $P_C(x)$ from the sequence $\langle \widehat{C}_0, \widehat{C}_1, \ldots, \widehat{C}_{n+m-1} \rangle$ of its values $C_k = P_C(\omega_{n+m-1}^k)$ at the roots of unity of order n+m-1.

$$P_A(x) = A_0 + \ldots + A_{n-1}x^{n-1} + 0 \cdot x^n + \ldots + 0 \cdot x^{n+m-2};$$

Assignment Project Exam Help

 $\underset{num}{\overset{P_{A}(\omega_{n+m-1}), \dots, P_{A}(\omega_{n+m-1}^{n+m-2})}{\text{nttps://powcoder.com}} } \underbrace{P_{B}(1), P_{B}(\omega_{n+m-1}), \dots, P_{B}(\omega_{n+m-1}^{n+m-2})}_{\text{multiplication}}$



 \Downarrow IDFT

$$P_C(x) = P_A(x) \cdot P_B(x) = \sum_{j=0}^{n+m-2} C_j x^j = \sum_{j=0}^{n+m-2} \left(\sum_{i=0}^{j} A_i B_{j-i} \right) x^j$$

• Crucial fact: the values $P_A(\omega_n^k)$ for all k such that $0 \le k < n$ can be computed in $\mathbf{O}(\mathbf{n} \log \mathbf{n})$ time!

Assignment Project Exam Help

roots of unity of order n would take n^2 many multiplications, even if we precompute all powers ω_n^{km} , because we have to perform mulhttps://powcoder.com

We can assume that n is a power of 2 - otherwise we can pad $P_A(x)$ Add We chat powcoders equal to the nearest power of 2

- Exercise: Show that for every n which is not a power of two the smallest power of 2 larger or equal to n is smaller than 2n.
- *Hint:* consider n in binary. How many bits does the nearest power of two have?

• Crucial fact: the values $P_A(\omega_n^k)$ for all k such that $0 \le k < n$ can be computed in $\mathbf{O}(\mathbf{n} \log \mathbf{n})$ time!

Assignment a latin of ectronia X_n and X_n roots of unity of order n would take n^2 many multiplications, even if we precompute all powers $\omega_n^{k\,m}$, because we have to perform multiplications A_n/p^k words from the $0 \le k \le n$.

We can assume that n is a power of 2 - otherwise we can pad $P_A(x)$ Add \mathbf{We} to the nearest power of 2.

- Exercise: Show that for every n which is not a power of two the smallest power of 2 larger or equal to n is smaller than 2n.
- *Hint:* consider n in binary. How many bits does the nearest power of two have?

• Crucial fact: the values $P_A(\omega_n^k)$ for all k such that $0 \le k < n$ can be computed in $\mathbf{O}(\mathbf{n} \log \mathbf{n})$ time!

Assignment abuting ectorist agree Help roots of unity of order n would take n^2 many multiplications, even if we precompute all powers $\omega_n^{k\,m}$, because we have to perform multiplications $A/n/p^k$ words from the property of the p

- We can assume that n is a power of 2 otherwise we can pad $P_A(x)$ with the coefficient until the number of 2 otherwise we can pad $P_A(x)$ to the nearest power of 2.
- Exercise: Show that for every n which is not a power of two the smallest power of 2 larger or equal to n is smaller than 2n.
- *Hint:* consider n in binary. How many bits does the nearest power of two have?

• Crucial fact: the values $P_A(\omega_n^k)$ for all k such that $0 \le k < n$ can be computed in $\mathbf{O}(\mathbf{n} \log \mathbf{n})$ time!

Assignment abuting ectorist agree Help roots of unity of order n would take n^2 many multiplications, even if we precompute all powers $\omega_n^{k\,m}$, because we have to perform multiplications $A/n/p^k$ words from the property of the p

- We can assume that n is a power of 2 otherwise we can pad $P_A(x)$ with process of 2 otherwise we can pad $P_A(x)$ to the nearest power of 2.
- Exercise: Show that for every n which is not a power of two the smallest power of 2 larger or equal to n is smaller than 2n.
- *Hint:* consider n in binary. How many bits does the nearest power of two have?

• Crucial fact: the values $P_A(\omega_n^k)$ for all k such that $0 \le k < n$ can be computed in $\mathbf{O}(\mathbf{n} \log \mathbf{n})$ time!

Assignment a latit of ectronia and the proofs of unity of order n would take n^2 many multiplications, even if we precompute all powers $\omega_n^{k\,m}$, because we have to perform multiplications $A/n/p^k$ world from the proof of the proo

- We can assume that n is a power of 2 otherwise we can pad $P_A(x)$ with 200 coefficient until 15 1 miles of Wylfcenty begins equal to the nearest power of 2.
- Exercise: Show that for every n which is not a power of two the smallest power of 2 larger or equal to n is smaller than 2n.
- *Hint:* consider n in binary. How many bits does the nearest power of two have?

- **Problem:** Given a sequence $A = \langle A_0, A_1, \dots, A_n \rangle$ compute its DFT.
- This amounts to finding values of $P_A(x)$ for all $x = \omega_n^k$, $0 \le k \le n-1$.
- Assignment Project Exam Help

$$P_A(x) = (A_0 + A_2 x^2 + A_4 x^4 + \dots + A_{n-2} x^{n-2}) + (A_1 x + A_3 x^3 + \dots + A_{n-1} x^{n-1})$$

https://powcoder.com(122)71/2-1)

• Let us define $A^{[0]} = \langle A_0, A_2, A_4, \dots A_{n-2} \rangle$ and $A^{[1]} = \langle A_1, A_3, A_5, \dots A_{n-1} \rangle$; then

Add WeChat powcoder

$$P_{A[1]}(y) = A_1 + A_3 y + A_5 y^2 + \ldots + A_{n-1} y^{n/2-1}$$

$$P_A(x) = P_{A[0]}(x^2) + x P_{A[1]}(x^2)$$

• Note that the number of coefficients of the polynomials $P_{A^{[0]}}(y)$ and $P_{A^{[1]}}(y)$ is n/2 each, while the number of coefficients of the polynomial $P_A(x)$ is n.

- **Problem:** Given a sequence $A = \langle A_0, A_1, \dots, A_n \rangle$ compute its DFT.
- This amounts to finding values of $P_A(x)$ for all $x = \omega_n^k$, $0 \le k \le n 1$.
- Assignment Project Exam Help

 $P_A(x) = (A_0 + A_2 x^2 + A_4 x^4 + \dots + A_{n-2} x^{n-2}) + (A_1 x + A_3 x^3 + \dots + A_{n-1} x^{n-1})$ $= A_0 + A_2 x^2 + A_4 (x^2)^2 + \dots + A_{n-2} (x^2)^{n/2-1}$

https://powcoder.com(**)**/2-1)

• Let us define $A^{[0]} = \langle A_0, A_2, A_4, \dots A_{n-2} \rangle$ and $A^{[1]} = \langle A_1, A_3, A_5, \dots A_{n-1} \rangle$; then

Add WeChat powcoder

 $P_{A[1]}(y) = A_1 + A_3 y + A_5 y^2 + \ldots + A_{n-1} y^{n/2-1}$

$$P_A(x) = P_{A[0]}(x^2) + x P_{A[1]}(x^2)$$

• Note that the number of coefficients of the polynomials $P_{A^{[0]}}(y)$ and $P_{A^{[1]}}(y)$ is n/2 each, while the number of coefficients of the polynomial $P_A(x)$ is n.

- **Problem:** Given a sequence $A = \langle A_0, A_1, \dots, A_n \rangle$ compute its DFT.
- This amounts to finding values of $P_A(x)$ for all $x = \omega_n^k$, $0 \le k \le n-1$.
- The main idea of the FFT algorithm: divide-and conquer by splitting the Splinguin Fig. 2 and the splitting the spl

 $P_A(x) = (A_0 + A_2 x^2 + A_4 x^4 + \dots + A_{n-2} x^{n-2}) + (A_1 x + A_3 x^3 + \dots + A_{n-1} x^{n-1})$ $= A_0 + A_2 x^2 + A_4 (x^2)^2 + \dots + A_{n-2} (x^2)^{n/2-1}$

https://powcoder.com¹(x²)^{n/2-1})

• Let us define $A^{[0]} = \langle A_0, A_2, A_4, \dots A_{n-2} \rangle$ and $A^{[1]} = \langle A_1, A_3, A_5, \dots A_{n-1} \rangle$; then

Add WeChat powcoder

 $P_{A[1]}(y) = A_1 + A_3 y + A_5 y^2 + \ldots + A_{n-1} y^{n/2-1}$

$$P_A(x) = P_{A[0]}(x^2) + x P_{A[1]}(x^2)$$

Note that the number of coefficients of the polynomials $P_{A[0]}(y)$ and $P_{A[1]}(y)$ is n/2 each, while the number of coefficients of the polynomial $P_{A}(x)$ is n.

- **Problem:** Given a sequence $A = \langle A_0, A_1, \dots, A_n \rangle$ compute its DFT.
- This amounts to finding values of $P_A(x)$ for all $x = \omega_n^k$, $0 \le k \le n-1$.
- The main idea of the FFT algorithm: divide-and conquer by splitting the Signature of the FFT algorithm: divide-and conquer by splitting the PA(x) = $(A_0 + A_2x^2 + A_4x^4 + ... + A_{n-2}x^{n-2}) + (A_1x + A_3x^3 + ... + A_{n-1}x^{n-1})$

 $\frac{A_0 + A_2 x^2 + A_4 (x^2)^2 + \ldots + A_{n-2} (x^2)^{n/2-1}}{\text{https://powcodef.com}^2 \text{com}^2 \text{com}^{1}(x^2)^{n/2-1})}$

• Let us define $A^{[0]}=\langle A_0,A_2,A_4,\ldots A_{n-2}\rangle$ and $A^{[1]}=\langle A_1,A_3,A_5,\ldots A_{n-1}\rangle;$ then

Add We Chat powe oder

$$P_{A[1]}(y) = A_1 + A_3 y + A_5 y^2 + \dots + A_{n-1} y^{n/2-1}$$

 $P_A(x) = P_{A[0]}(x^2) + x P_{A[1]}(x^2)$

• Note that the number of coefficients of the polynomials $P_{A[0]}(y)$ and $P_{A[1]}(y)$ is n/2 each, while the number of coefficients of the polynomial $P_A(x)$ is n.

- **Problem:** Given a sequence $A = \langle A_0, A_1, \dots, A_n \rangle$ compute its DFT.
- This amounts to finding values of $P_A(x)$ for all $x = \omega_n^k$, $0 \le k \le n-1$.
- The main idea of the FFT algorithm: divide-and conquer by splitting the Splinghi Figure Power and Conquer by splitting the Pa(x) = $(A_0 + A_2x^2 + A_4x^4 + ... + A_{n-2}x^{n-2}) + (A_1x + A_3x^3 + ... + A_{n-1}x^{n-1})$ = $A_0 + A_2x^2 + A_4(x^2)^2 + ... + A_{n-2}(x^2)^{n/2-1}$

 $https://powco²der².com¹(x²)^{n/2-1})$

• Let us define $A^{[0]} = \langle A_0, A_2, A_4, \dots A_{n-2} \rangle$ and $A^{[1]} = \langle A_1, A_3, A_5, \dots A_{n-1} \rangle$; then

Add We Chat powe oder

$$P_{A[1]}(y) = A_1 + A_3 y + A_5 y^2 + \dots + A_{n-1} y^{n/2-1}$$

 $P_{A}(x) = P_{A[0]}(x^2) + x P_{A[1]}(x^2)$

• Note that the number of coefficients of the polynomials $P_{A^{[0]}}(y)$ and $P_{A^{[1]}}(y)$ is n/2 each, while the number of coefficients of the polynomial $P_A(x)$ is n.

 Problem of size n: Evaluate a polynomial with n coefficients at n many roots of unity.

Assignment Project Exam Help

• We reduced evaluation of our polynomial $P_A(x)$ with n coefficients at inputs $x = \omega_n^0$, $x = \omega_n^1$, $x = \omega_n^2$, ..., $x = \omega_n^{n-1}$ to evaluation of two polynomials $P_{A[0]}(y)$ and $P_{A[1]}$ https://powcoder.com

• However, as x ranges through values $\{\omega_n^0, \omega_n^1, \omega_n^2, \dots, \omega_n^{n-1}\}$, the value of $y = x^2$ ranges through $\{\omega_{n/2}^0, \omega_{n/2}^1, \omega_{n/2}^2, \dots, \omega_{n/2}^{n-1}\}$, and there are only n/2 distinct such values.

Once Add nWe Chat pow.coder multiplications with numbers ω_n^k to obtain the values of

$$\begin{split} P_A(\omega_n^k) &= P_{A^{[0]}}((\omega_n^k)^2) + \omega_n^k \cdot P_{A^{[1]}}((\omega_n^k)^2) \\ &= P_{A^{[0]}}(\omega_{n/2}^k) + \omega_n^k \cdot P_{A^{[1]}}(\omega_{n/2}^k). \end{split}$$

● Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.

Problem of size n:
 Evaluate a polynomial with n coefficients at n many roots of unity.

Assignment Project Examy Help

• We reduced evaluation of our polynomial $P_A(x)$ with n coefficients at inputs $x = \omega_n^0$, $x = \omega_n^1$, $x = \omega_n^2$, ..., $x = \omega_n^{n-1}$ to evaluation of two polynomials $P_{A^{[0]}}(y)$ and $P_{A^{[0]}}$ https://powcoder.com

• However, as x ranges through values $\{\omega_n^0, \omega_n^1, \omega_n^2, \dots, \omega_n^{n-1}\}$, the value of $y = x^2$ ranges through $\{\omega_{n/2}^0, \omega_{n/2}^1, \omega_{n/2}^2, \dots, \omega_{n/2}^{n-1}\}$, and there are only n/2 distinct such values.

Once Add nWe Chat pow.coder
multiplications with numbers ω_n^k to obtain the values of

$$\begin{split} P_{A}(\omega_{n}^{k}) &= P_{A^{[0]}}((\omega_{n}^{k})^{2}) + \omega_{n}^{k} \cdot P_{A^{[1]}}((\omega_{n}^{k})^{2}) \\ &= P_{A^{[0]}}(\omega_{n/2}^{k}) + \omega_{n}^{k} \cdot P_{A^{[1]}}(\omega_{n/2}^{k}). \end{split}$$

● Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.

• Problem of size n: Evaluate a polynomial with n coefficients at n many roots of unity.

Assignment Project Examy Help

- We reduced evaluation of our polynomial $P_A(x)$ with n coefficients at inputs $x = \omega_n^0, \ x = \omega_n^1, \ x = \omega_n^2, \dots, x = \omega_n^{n-1}$ to evaluation of two polynomials $P_{A[0]}(y)$ and $P_{A[1]}(y)$ at the with n/2 coefficients, at points $y = x^2$ for the same values of inputs x.
- However, as x ranges through values $\{\omega_n^0, \omega_n^1, \omega_n^2, \dots, \omega_n^{n-1}\}$, the value of $y = x^2$ ranges through $\{\omega_{n/2}^0, \omega_{n/2}^1, \omega_{n/2}^2, \dots, \omega_{n/2}^{n-1}\}$, and there are only n/2 distinct such values.
- Once Add nWeChat powcoder multiplications with numbers ω_n^k to obtain the values of

$$\begin{split} P_A(\omega_n^k) &= P_{A^{[0]}}((\omega_n^k)^2) + \omega_n^k \cdot P_{A^{[1]}}((\omega_n^k)^2) \\ &= P_{A^{[0]}}(\omega_{n/2}^k) + \omega_n^k \cdot P_{A^{[1]}}(\omega_{n/2}^k). \end{split}$$

• Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.

Problem of size n:
 Evaluate a polynomial with n coefficients at n many roots of unity.

Assignment Project Examy Help

- We reduced evaluation of our polynomial $P_A(x)$ with n coefficients at inputs $x = \omega_n^0, \ x = \omega_n^1, \ x = \omega_n^2, \dots, x = \omega_n^{n-1}$ to evaluation of two polynomials $P_{A[0]}(y)$ and $P_{A[1]}(y)$ tack with n/2 coefficients, at points $y = x^2$ for the same values of inputs x.
- However, as x ranges through values $\{\omega_n^0, \omega_n^1, \omega_n^2, \dots, \omega_n^{n-1}\}$, the value of $y = x^2$ ranges through $\{\omega_{n/2}^0, \omega_{n/2}^1, \omega_{n/2}^2, \dots, \omega_{n/2}^{n-1}\}$, and there are only n/2 distinct such values.
- Once Add n We Chat powcoder multiplications with numbers ω_n^k to obtain the values of

$$\begin{split} P_A(\omega_n^k) &= P_{A^{[0]}}((\omega_n^k)^2) + \omega_n^k \cdot P_{A^{[1]}}((\omega_n^k)^2) \\ &= P_{A^{[0]}}(\omega_{n/2}^k) + \omega_n^k \cdot P_{A^{[1]}}(\omega_{n/2}^k). \end{split}$$

● Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.

Problem of size n:
 Evaluate a polynomial with n coefficients at n many roots of unity.

Assignment Project Examy Help

- We reduced evaluation of our polynomial $P_A(x)$ with n coefficients at inputs $x = \omega_n^0, \ x = \omega_n^1, \ x = \omega_n^2, \dots, x = \omega_n^{n-1}$ to evaluation of two polynomials $P_{A[0]}(y)$ and $P_{A[1]}(y)$ tack with n/2 deficients, at points $y = x^2$ for the same values of inputs x.
- However, as x ranges through values $\{\omega_n^0, \omega_n^1, \omega_n^2, \dots, \omega_n^{n-1}\}$, the value of $y=x^2$ ranges through $\{\omega_{n/2}^0, \omega_{n/2}^1, \omega_{n/2}^2, \dots, \omega_{n/2}^{n-1}\}$, and there are only n/2 distinct such values.
- Once we said n/Value of 1 to obtain the values of

$$\begin{split} P_A(\omega_n^k) &= P_{A^{[0]}}((\omega_n^k)^2) + \omega_n^k \cdot P_{A^{[1]}}((\omega_n^k)^2) \\ &= P_{A^{[0]}}(\omega_{n/2}^k) + \omega_n^k \cdot P_{A^{[1]}}(\omega_{n/2}^k). \end{split}$$

• Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.

Problem of size n:
 Evaluate a polynomial with n coefficients at n many roots of unity.

Assignment Project Examy Help

- We reduced evaluation of our polynomial $P_A(x)$ with n coefficients at inputs $x = \omega_n^0, \ x = \omega_n^1, \ x = \omega_n^2, \dots, x = \omega_n^{n-1}$ to evaluation of two polynomials $P_{A[0]}(y)$ and $P_{A[1]}(y)$ and with with n/2 coefficients, at points $y = x^2$ for the same values of inputs x.
- However, as x ranges through values $\{\omega_n^0, \omega_n^1, \omega_n^2, \dots, \omega_n^{n-1}\}$, the value of $y = x^2$ ranges through $\{\omega_{n/2}^0, \omega_{n/2}^1, \omega_{n/2}^2, \dots, \omega_{n/2}^{n-1}\}$, and there are only n/2 distinct such values.
- Once we see that n/N value of n/N value of n/N value of n/N values of n/N values

$$\begin{split} P_A(\omega_n^k) &= P_{A^{[0]}}((\omega_n^k)^2) + \omega_n^k \cdot P_{A^{[1]}}((\omega_n^k)^2) \\ &= P_{A^{[0]}}(\omega_{n/2}^k) + \omega_n^k \cdot P_{A^{[1]}}(\omega_{n/2}^k). \end{split}$$

• Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.

- Note that by the Cancelation Lemma $\omega_n^{n/2} = \omega_{2n/2}^{n/2} = \omega_2 = -1$.
- Thus,

Assignment Project Exam Help

• We can now simplify evaluation of

https://powcoder.com

for $n/2 \le k < n$ as follows: let k = n/2 + m where $0 \le m < n/2$; the

Add WeChat powcoder

$$\begin{split} &= P_{A[0]}(\omega_{n/2}^{n/2}\omega_{n/2}^m) + \omega_n^{n/2}\omega_n^m P_{A[1]}(\omega_{n/2}^{n/2}\omega_{n/2}^m) \\ &= P_{A[0]}(\omega_{n/2}^m) - \omega_n^m P_{A[1]}(\omega_{n/2}^m) \end{split}$$

 $\bullet \ \text{Compare this with} \quad P_A(\omega_n^m) = P_{A^{[0]}}(\omega_{n/2}^m) + \omega_n^m P_{A^{[1]}}(\omega_{n/2}^m) \quad \text{for } 0 \leq m < n/2.$

- Note that by the Cancelation Lemma $\omega_n^{n/2} = \omega_{2n/2}^{n/2} = \omega_2 = -1$.
- Thus,

Assignment Project Exam Help

• We can now simplify evaluation of

https://powcoder.com

for $n/2 \le k < n$ as follows: let k = n/2 + m where $0 \le m < n/2$; then

Add WeChat powcoder

$$\begin{split} &= P_{A[0]}(\omega_{n/2}^{n/2}\omega_{n/2}^m) + \omega_n^{n/2}\omega_n^m P_{A[1]}(\omega_{n/2}^{n/2}\omega_{n/2}^m) \\ &= P_{A[0]}(\omega_{n/2}^m) - \omega_n^m P_{A[1]}(\omega_{n/2}^m) \end{split}$$

• Compare this with $P_A(\omega_n^m) = P_{A^{[0]}}(\omega_{n/2}^m) + \omega_n^m P_{A^{[1]}}(\omega_{n/2}^m)$ for $0 \le m < n/2$.

- Note that by the Cancelation Lemma $\omega_n^{n/2} = \omega_{2n/2}^{n/2} = \omega_2 = -1$.
- Thus,

Assignment Project Exam Help

We can now simplify evaluation of

https://powcoder.com

for $n/2 \le k < n$ as follows: let k = n/2 + m where $0 \le m < n/2$; then

$$\begin{array}{c} \mathbf{Add}^{n/\text{W}} = \mathbf{C}_{0}(\omega_{n/2}^{n/2}\omega_{n/2}^{\text{m}} + \mathbf{p}^{n/2})^{2+\text{m}} \mathbf{W} \mathbf{C}_{0}^{n/2} \mathbf{C} \mathbf{C} \mathbf{C} \\ = P_{A^{[0]}}(\omega_{n/2}^{n/2}\omega_{n/2}^{\text{m}}) + \omega_{n}^{n/2}\omega_{n}^{m}P_{A^{[1]}}(\omega_{n/2}^{n/2}\omega_{n/2}^{\text{m}}) \\ = P_{A^{[0]}}(\omega_{n/2}^{\text{m}}) - \omega_{n}^{m}P_{A^{[1]}}(\omega_{n/2}^{\text{m}}) \end{array}$$

• Compare this with $P_A(\omega_n^m) = P_{A^{[0]}}(\omega_{n/2}^m) + \omega_n^m P_{A^{[1]}}(\omega_{n/2}^m)$ for $0 \le m < n/2$.

- Note that by the Cancelation Lemma $\omega_n^{n/2} = \omega_{2n/2}^{n/2} = \omega_2 = -1$.
- Thus,

Assignment Project Exam Help

We can now simplify evaluation of

https://powcoder.com

for $n/2 \le k < n$ as follows: let k = n/2 + m where $0 \le m < n/2$; then

$$\begin{array}{c} \mathbf{Add}^{n/\mathbf{W}} = \mathbf{C}_{[0]}(\mathbf{A}_{n/2}^{(2+m)} + \mathbf{P}_{n/2}^{(2+m)} \mathbf{W} \cdot \mathbf{C}_{0}^{n/2} + \mathbf{C}_{0}^{m} \mathbf{C}_{1}^{(2+m)} \mathbf{C}_{0}^{m} \mathbf{C}_{1}^{m/2} \mathbf{C}_{1}^{m/2} \\ &= P_{A^{[0]}}(\omega_{n/2}^{m/2} \omega_{n/2}^{m}) + \omega_{n}^{m} P_{A^{[1]}}(\omega_{n/2}^{m/2} \omega_{n/2}^{m}) \\ &= P_{A^{[0]}}(\omega_{n/2}^{m}) - \omega_{n}^{m} P_{A^{[1]}}(\omega_{n/2}^{m}) \end{array}$$

 $\bullet \ \text{Compare this with} \quad P_A(\omega_n^m) = P_{A^{[0]}}(\omega_{n/2}^m) + \omega_n^m P_{A^{[1]}}(\omega_{n/2}^m) \quad \text{for } 0 \leq m < n/2.$

| ←□ → ←□ → ← = → へ = → へ へ へ |

 $Assignment \stackrel{\text{So we can replace evaluations of}}{\text{Assignment}} \stackrel{\text{No like}}{\text{Partial Suppose}} = r_{A^{[0]}} \stackrel{\text{like}}{\text{like}} = r_{A^{[1]}} \stackrel{\text{Like}}{\text{Like}} = r_{A$

for
$$k = 0$$
 to $k = n - 1$
with https://powteoder.com
and just let for $k = 0$ to $k = n/2 - 1$

$$Add \ \, \mathbf{W}^{P_{A}(\omega^{k})} = \mathbf{L}^{P_{A[0]}(\omega^{k}_{n/2}) + \omega^{k}_{n}P_{A[1]}(\omega^{k}_{n/2})} \underbrace{\mathbf{C}^{P_{A[1]}(\omega^{k}_{n/2})}_{P_{A}[1]} \underbrace{\mathbf{C}^{k}_{n/2}}_{P_{A}[1]} \underbrace{\mathbf{C}^{k}_{n/2}}_{P_{A}$$

• We can now write a pseudo-code for our FFT algorithm

 $Assignment \stackrel{\text{So we can replace evaluations of}}{\text{Assignment}} \stackrel{\text{No like}}{\text{Partial Suppose}} = r_{A^{[0]}} \stackrel{\text{like}}{\text{like}} = r_{A^{[1]}} \stackrel{\text{Like}}{\text{Like}} = r_{A$

for
$$k = 0$$
 to $k = n - 1$
with integrals on power oder. Com
and just let for $k = 0$ to $k = n/2 - 1$

$$Add \ \, \mathbf{W}^{P_{A}(\omega^{k})} = \mathbf{P}^{P_{A}[0]}(\omega^{k}_{n/2}) + \omega^{k}_{n}P_{A}[1](\omega^{k}_{n/2}) \\ = \mathbf{Q}^{P_{A}[1]}(\omega^{k}_{n/2}) + \omega^{k}_{n/2}(\omega^{k}_{n/2}) + \omega^{k}_{n/2}(\omega^{k}_{n/2}) \\ = \mathbf{Q}^{P_{A}[1]}(\omega^{k}_{n/2}) + \omega^{k}_{n/2}(\omega^{k}_{n/2}) + \omega^{k}_{n/2}(\omega^{k}_{n/2}) \\ = \mathbf{Q}^{P_{$$

• We can now write a pseudo-code for our FFT algorithm:

FFT algorithm

```
function FFT(A)
        n \leftarrow \operatorname{length}[A]
        if n = 1 then return A
                                     t Project Exam Help
                  \leftarrow (A_1, A_3, \dots A_{n-1});
            y^{[0]} \leftarrow FFT(A^{[0]});
8:
                                         {f powcoder.com}_{\% 	ext{ a variable to hold powers of } \omega_n}
9:
10:
                                                        \% P_A(\omega_n^k) = P_{A[0]}(\omega_{n/2}^k) + \omega_n^k P_{A[1]}(\omega_{n/2}^k)
11:
             for k = 0 to k = n/2 - 1 do:
              A<sup>y</sup>dd WeChat pow
12:
13:
14:
                 \omega \leftarrow \omega \cdot \omega_n;
                                                           y_{n/2+k}
15:
             end for
16:
             return y
17:
         end if
18: end function
```

How fast is the Fast Fourier Transform?

• We have recursively reduced evaluation of a polynomial $P_A(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $P_{A[0]}(y)$ and $P_{A[1]}(y)$, each with n/2 coefficients, at n/2 many roots of unity of

Assignment Project Exam Help

$$\underline{P_A(\omega_n^k)} = P_{A^{[0]}}(\omega_{n/2}^k) + \omega_n^k P_{A^{[1]}}(\omega_{n/2}^k)$$
(1)

https://powcoder.com

and

$$\underline{P_A(\omega_n^{n/2+k})} = \underline{A^{[0]}(\omega_{n/2}^k)} - \omega_n^k \underline{A^{[1]}(\omega_{n/2}^k)}$$
(2)

Add WeChat powcoder

- Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2T(n/2) + cn$$

• The Master Theorem gives $T(n) = \Theta(n \log n)$.



How fast is the Fast Fourier Transform?

• We have recursively reduced evaluation of a polynomial $P_A(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $P_{A^{[0]}}(y)$ and $P_{A^{[1]}}(y)$, each with n/2 coefficients, at n/2 many roots of unity of

Assignment Project Exam Help

$$\underbrace{P_A(\omega_n^k)}_{} = \underbrace{P_{A[0]}(\omega_{n/2}^k)}_{} + \omega_n^k \underbrace{P_{A[1]}(\omega_{n/2}^k)}_{} \tag{1}$$

https://powcoder.com

$$\underbrace{P_A(\omega_n^{n/2+k})}_{n} = \underbrace{A^{[0]}(\omega_{n/2}^k)}_{-\omega_n} - \omega_n^k \underbrace{A^{[1]}(\omega_{n/2}^k)}_{-\omega_n^k}$$
(2)

Add WeChat powcoder

- Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2T(n/2) + cn$$

• The Master Theorem gives $T(n) = \Theta(n \log n)$.

How fast is the Fast Fourier Transform?

• We have recursively reduced evaluation of a polynomial $P_A(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $P_{A^{[0]}}(y)$ and $P_{A^{[1]}}(y)$, each with n/2 coefficients, at n/2 many roots of unity of

order n/2. Sile selections to obtain the values of the term of the control of the

$$https://pow/coder/com$$
(1)

$$\underbrace{P_A(\omega_n^{n/2+k})}_{n} = \underbrace{A^{[0]}(\omega_{n/2}^k)}_{n/2} - \omega_n^k \underbrace{A^{[1]}(\omega_{n/2}^k)}_{n/2} \tag{2}$$

for all dd /2. We Chat pow coder

- Thus, we have reduced a problem of size n to two such problems of size n/2,
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2T(n/2) + cn$$

• The Master Theorem gives $T(n) = \Theta(n \log n)$.

COMP3121/9101

How fast is the Fast Fourier Transform?

• We have recursively reduced evaluation of a polynomial $P_A(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $P_{A^{[0]}}(y)$ and $P_{A^{[1]}}(y)$, each with n/2 coefficients, at n/2 many roots of unity of order n/2

Assie some n/2 and the project PAE/X am/2 and publications to obtain the values

$$https://pow/coder/com$$
(1)

and

$$\underbrace{P_A(\omega_n^{n/2+k})}_{n} = \underbrace{A^{[0]}(\omega_{n/2}^k)}_{n/2} - \omega_n^k \underbrace{A^{[1]}(\omega_{n/2}^k)}_{n/2} \tag{2}$$

for all 0 dd/2. We'chat powcoder

- Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2T(n/2) + cn$$

• The Master Theorem gives $T(n) = \Theta(n \log n)$.

How fast is the Fast Fourier Transform?

• We have recursively reduced evaluation of a polynomial $P_A(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $P_{A^{[0]}}(y)$ and $P_{A^{[1]}}(y)$, each with n/2 coefficients, at n/2 many roots of unity of order n/2

Assie some n/2 and the project PAE/X am/2 and publications to obtain the values

$$\frac{P_A(\omega_n^k)}{\text{https://pow/coder.icom}} = P_{A^{[0]}}(\omega_{n/2}^k) + \omega_n^k P_{A^{[1]}}(\omega_{n/2}^k) \tag{1}$$

and

$$\underbrace{P_A(\omega_n^{n/2+k})}_{n} = \underbrace{A^{[0]}(\omega_{n/2}^k)}_{n/2} - \omega_n^k \underbrace{A^{[1]}(\omega_{n/2}^k)}_{n/2} \tag{2}$$

for all ded_/2. We'chat powcoder

- Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2T(n/2) + cn$$

• The Master Theorem gives $T(n) = \Theta(n \log n)$.

How fast is the Fast Fourier Transform?

• We have recursively reduced evaluation of a polynomial $P_A(x)$ with n coefficients at n roots of unity of order n to evaluations of two polynomials $P_{A^{[0]}}(y)$ and $P_{A^{[1]}}(y)$, each with n/2 coefficients, at n/2 many roots of unity of order n/2

Assie Septimental Project PAEX am/2 at the publications to obtain the values

$$\frac{P_A(\omega_n^k)}{\text{https://pow/coder.icom}} = P_{A^{[0]}}(\omega_{n/2}^k) + \omega_n^k P_{A^{[1]}}(\omega_{n/2}^k) \tag{1}$$

and

$$\underbrace{P_A(\omega_n^{n/2+k})}_{n} = \underbrace{A^{[0]}(\omega_{n/2}^k)}_{n-k} - \omega_n^k \underbrace{A^{[1]}(\omega_{n/2}^k)}_{n-k}$$
(2)

for all ded_/2. We'chat powcoder

- Thus, we have reduced a problem of size n to two such problems of size n/2, plus a linear overhead.
- Consequently, our algorithm's run time satisfies the recurrence

$$T(n) = 2T(n/2) + cn$$

• The Master Theorem gives $T(n) = \Theta(n \log n)$.

GON (Poster /oster

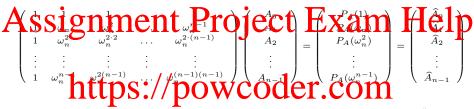
• Evaluation of a polynomial $P_A(x) = A_0 + A_1 x + \ldots + A_{n-1} x^{n-1}$ at roots of unity ω_n^k of order n can be represented in the matrix form as follows:

Assignment Project Exam Help

https://powcoder.com

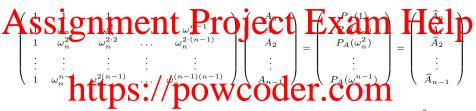
- The FFT is just a method of replacing this matrix-vector multiplication taking n^2 many multiplications with an $n \log n$ procedure.
- Add WeChat powcoder

• Evaluation of a polynomial $P_A(x) = A_0 + A_1 x + \ldots + A_{n-1} x^{n-1}$ at roots of unity ω_n^k of order n can be represented in the matrix form as follows:



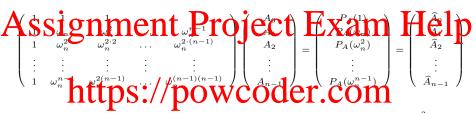
• The FFT is just a method of replacing this matrix-vector multiplication taking n^2 many multiplications with an $n \log n$ procedure.

• Evaluation of a polynomial $P_A(x) = A_0 + A_1 x + \ldots + A_{n-1} x^{n-1}$ at roots of unity ω_n^k of order n can be represented in the matrix form as follows:



ullet The FFT is just a method of replacing this matrix-vector multiplication taking n^2 many multiplications with an $n \log n$ procedure.

• Evaluation of a polynomial $P_A(x) = A_0 + A_1 x + \ldots + A_{n-1} x^{n-1}$ at roots of unity ω_n^k of order n can be represented in the matrix form as follows:



• The FFT is just a method of replacing this matrix-vector multiplication taking n^2 many multiplications with an $n \log n$ procedure.

From $P_{\mathbf{A}}(1) = P_{\mathbf{A}}(\omega_n^0 \mathbf{X}) P_{\mathbf{A}}(\omega_n) P_{\mathbf{A}}(\omega_n^0 \mathbf{X}) P_{\mathbf{A}}(\omega_n^0 \mathbf{X}) \dots P_{\mathbf{A}}(\omega_n^{n-1})$, we get the coefficients from $\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ \vdots \\ A_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & P_{\mathbf{A}}(\omega_n^{n-1}) \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^{2\cdot 2} & \dots & \omega_n^{2\cdot (n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} \widehat{A}_1 \\ \widehat{A}_2 \\ \vdots \\ \widehat{A}_{n-1} \end{pmatrix}$ (3)

Another remarkable feature of the roots of unity:

To obtain the inverse of the above matrix, all we have to do is just change the signs of Assignment Project Exam Help

$$\begin{array}{c} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ \text{https://powcoder-com} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix}^{-1} \\ \text{Add} \ \ \mathbf{W} \stackrel{1}{\text{ecc}} \stackrel{1}{\text{chart powcoder}} \stackrel{1}{\text{chart powcoder}} \\ & \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n^{(n-1)(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_n^{-(n-1)} & \omega_n^{-2(n-1)} & \dots & \omega_n^{-(n-1)(n-1)} \end{pmatrix}$$

To see this, note that if we compute the product

$$A_{ssign}^{1 \atop 1} \underbrace{ A_{n}^{1} \atop \omega_{n}^{2}}_{1 \atop \dots \atop \omega_{n}^{n-1} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \dots \atop \dots \atop \omega_{n}^{(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{1} \atop \omega_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{1} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{2} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{2} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2(n-1)(n-1)}}^{1 \atop n} \underbrace{ J_{n}^{2} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \underbrace{ J_{n}^{2} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \underbrace{ J_{n}^{2} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \underbrace{ J_{n}^{2} \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \atop \dots \atop \omega_{n}^{2} \underbrace{ J_{n}^{2} \atop \omega_{n}^{2} \underbrace{ J_{n}^{2} \atop \omega_{n}^{2} \atop \omega_{n}^{2} \underbrace{ J_{n}^{2} \atop \omega_{n}^{2} \underbrace{$$

the (i,j) httips properties and j^{th} column:

$$(1 \omega_n^i \Delta_n^i dd \omega_n^i)^{-1} \mathbf{Ch}_{at}^{\omega_n^{-j}} \mathbf{D}_{k=0}^{\mathbf{U}_n^i} \mathbf{C}_n^{\mathbf{U}_n^i} \mathbf{C}_n^$$

◆ロト ◆御ト ◆恵ト ◆恵ト 恵 めなべ

We now have two possibilities:

$$\mathbf{0}$$
 $i=j$: then

$$\sum_{k=0}^{n-1} \omega_n^{(i-j)k} = \sum_{k=0}^{n-1} \omega_n^0 = \sum_{k=0}^{n-1} 1 = n;$$

Assignment Project Exam Help

https://powcoder.com

So

Add WeChat powcoder

$$\begin{pmatrix} 1 & \omega_n^i & \omega_n^{2 \cdot i} & \dots & \omega_n^{i \cdot (n-1)} \end{pmatrix} \begin{pmatrix} \omega_n^j \\ \omega_n^{-2j} \\ \vdots \\ \omega_n^{-(n-1)j} \end{pmatrix} = \sum_{k=0}^{n-1} \omega_n^{(i-j)k} = \begin{cases} n & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

(4)

We now have two possibilities:

 $\mathbf{0}$ i=j: then

$$\sum_{k=0}^{n-1} \omega_n^{(i-j)k} = \sum_{k=0}^{n-1} \omega_n^0 = \sum_{k=0}^{n-1} 1 = n;$$

Assignment) k PerceitectofExamportelp with the ratio ω_n^{i-j} and thus

$$h \sum_{k=0}^{n-1} \frac{1}{s^{j+k}} \frac{1 - \omega_n^{(i-j)n}}{p \omega_n^{n}} c \frac{1 - \omega_n^{n}}{q \omega_n^{n}} \frac{1 - \omega_n^{n-j}}{q \omega_n^{n-j}} = 0$$

So,

Add WeChat powcoder

$$\begin{pmatrix} 1 & \omega_n^i & \omega_n^{2 \cdot i} & \dots & \omega_n^{i \cdot (n-1)} \end{pmatrix} \begin{pmatrix} \omega_n^j \\ \omega_n^{-2j} \\ \vdots \\ \omega_n^{-(n-1)j} \end{pmatrix} = \sum_{k=0}^{n-1} \omega_n^{(i-j)k} = \begin{cases} n & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

(4)

<□▶ <圖▶ <필▶ <臺▶ 臺 ◆

We now have two possibilities:

 $\mathbf{0}$ i=j: then

$$\sum_{k=0}^{n-1} \omega_n^{(i-j)k} = \sum_{k=0}^{n-1} \omega_n^0 = \sum_{k=0}^{n-1} 1 = n;$$

Assignment Project Exam Help with the ratio ω_n^{i-j} and thus

$$h \sum_{k=0}^{n-1} \frac{1}{s^{j+k}} \frac{1 - \omega_n^{(i-j)n}}{p \omega_n^{n}} c \frac{1 - \omega_n^{n}}{q \omega_n^{n}} \frac{1 - \omega_n^{n-j}}{q \omega_n^{n-j}} = 0$$

So,

$$\begin{array}{cccc} & \textbf{Add WeChat} \\ \left(1 & \omega_n^i & \omega_n^{2 \cdot i} & \dots & \omega_n^{i \cdot (n-1)}\right) & & & \\ & \vdots & & \\ & \omega_n^{-(n-1)j} & & & \\ & \vdots & & \\ & \omega_n^{-(n-1)j} & & \\ \end{array} \right) = \sum_{k=0}^{n-1} \omega_n^{(i-j)k} = \begin{cases} n & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

So we get:

i.e.,

Add WeChat powcoder

$$\begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^{2\cdot 2} & \dots & \omega_n^{2\cdot (n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2}(n-1) & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} = \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n^{-1} & \omega_n^{-2} & \dots & \omega_n^{-(n-1)} \\ 1 & \omega_n^{-2} & \omega_n^{-2\cdot 2} & \dots & \omega_n^{-2\cdot (n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_n^{-(n-1)} & \omega_n^{-2(n-1)} & \dots & \omega_n^{-(n-1)(n-1)} \end{pmatrix}$$

◆ロト ◆問 ト ◆恵 ト ◆恵 ト ・ 恵 ・ 夕久で

• This means that to covert from the values

Add We Chat powcoder

$$P_A(x) = A_0 + A_1 x + A_2 x^2 + A_{n-1} x^{n-1}$$

- ① the root of unity ω_n is replaced by $\overline{\omega_n} = e^{-i\frac{2\pi}{n}}$,

$$\begin{array}{c} A_{0} \\ A_{1} \\ A_{2} \\ \vdots \\ \end{array} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_{n} & \omega_{n}^{2} & \dots & \omega_{n}^{n-1} \\ 1 & \omega_{n}^{2} & \omega_{n}^{2\cdot 2} & \dots & \omega_{n}^{2\cdot (n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \end{array} = \begin{pmatrix} P_{A}(1) \\ P_{A}(\omega_{n}) \\ P_{A}(\omega_{n}^{2}) \\ \vdots & \vdots & \vdots \\ P_{A}(1) \\ \end{array} = \frac{1}{n} \begin{pmatrix} 1 & \omega_{n}^{-1} & \omega_{n}^{-2} & \dots & \omega_{n}^{-(n-1)} \\ 1 & \omega_{n}^{-2} & \omega_{n}^{-2\cdot 2} & \dots & \omega_{n}^{-(n-1)} \\ 1 & \omega_{n}^{-2\cdot 2} & \omega_{n}^{-2\cdot 2\cdot 2} & \dots & \omega_{n}^{-2\cdot (n-1)} \\ \end{pmatrix} \begin{pmatrix} P_{A}(\omega_{n}) \\ P_{A}(\omega_{n}) \\ P_{A}(\omega_{n}^{2}) \\ \vdots \\ P_{A}(\omega_{n}^{2}) \\ \end{pmatrix} = \begin{pmatrix} 1 & \omega_{n}^{-1} & \omega_{n}^{-2} & \dots & \omega_{n}^{-(n-1)} \\ 1 & \omega_{n}^{-2} & \omega_{n}^{-2\cdot 2} & \dots & \omega_{n}^{-2\cdot (n-1)} \\ \end{pmatrix} \begin{pmatrix} P_{A}(\omega_{n}) \\ P_{A}(\omega_{n}^{2}) \\ \vdots \\ P_{A}(\omega_{n}^{2}) \\ \end{pmatrix}$$

• This means that to covert from the values

$$P_A(x) = A_0 + A_1 x + A_2 x^2 + A_{n-1} x^{n-1}$$

we can use **the same** FFT algorithm with the only change that:

- ① the root of unity ω_n is replaced by $\overline{\omega_n} = e^{-i\frac{2\pi}{n}}$,

• This means that to covert from the values

 $\underset{\text{which we denoted by } \langle A_0, A_1, A_2, \dots, \widehat{A_{n-1}} \rangle}{\text{Add}} \underbrace{ \underset{\text{which we denoted by } \langle A_0, A_1, A_2, \dots, \widehat{A_{n-1}} \rangle}{\text{pro}} \underbrace{ \underset{\text{back to the coefficient form}}{\text{pro}} }_{\text{back to the coefficient form}}$

$$P_A(x) = A_0 + A_1 x + A_2 x^2 + A_{n-1} x^{n-1}$$

we can use **the same** FFT algorithm with the only change that:

- the root of unity ω_n is replaced by $\overline{\omega_n} = e^{-i\frac{2\pi}{n}}$,
- 2 the resulting output values are divided by n.

$$Assignment^{1} Project^{1} Exam^{P_{A}(1)}_{P_{A}(\omega_{n})} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_{n} & \omega_{n}^{2} & \dots & \omega_{n}^{n-1} \\ 1 & \omega_{n}^{2} & \omega_{n}^{2\cdot 2} & \dots & \omega_{n}^{2\cdot (n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_{n}^{-1} & Project^{1} Exam^{P_{A}(1)}_{P_{A}(\omega_{n}^{2})} = \frac{1}{n} \begin{bmatrix} 1 & \omega_{n}^{-1} & \omega_{n}^{-2} & \dots & \omega_{n}^{-(n-1)} \\ 1 & \omega_{n}^{-2} & \omega_{n}^{-2\cdot 2} & \dots & \omega_{n}^{-(n-1)} \\ 1 & \omega_{n}^{-2} & \omega_{n}^{-2\cdot 2} & \dots & \omega_{n}^{-(n-1)} \end{bmatrix} P_{A}(\omega_{n}) P_{A}(\omega_{n}^{2}) P_{A}(\omega_{$$

• This means that to covert from the values

 $\underset{\text{which we denoted by } \langle A_0, A_1, A_2, \dots, \widehat{A_{n-1}} \rangle }{\text{Add}} \underbrace{ \underset{\text{product}}{\text{product}} \underset{\text{back to the coefficient form}}{\text{product}} }_{\text{product}}$

$$P_A(x) = A_0 + A_1 x + A_2 x^2 + A_{n-1} x^{n-1}$$

we can use **the same** FFT algorithm with the only change that:

- the root of unity ω_n is replaced by $\overline{\omega_n} = e^{-i\frac{2\pi^2}{n}}$,
- ② the resulting output values are divided by n.

4□ > 4□ > 4Ē > 4Ē > Ē < 50<

Inverse Fast Fourier Transform (IFFT):

```
1: function IFFT*(\widehat{A})
        n \leftarrow \operatorname{length}(\widehat{A})
        if n=1 then return \widehat{A}
                        ment-Project Exam Help
        else
            y^{[0]} \leftarrow \widehat{IFFT}^*(\widehat{A}^{[0]});
            y^{[1]} \leftarrow IFFT^*(\widehat{A}^{[1]});
            \underset{\text{for } k = 0 \text{ to } k = n/2}{\text{https://powcoder.com}}
9:
10:
11:
                y_k \leftarrow y_i^{[0]} + \omega \cdot y_i^{[1]}:
12:
              And derive Chat powcoder
13:
14:
15:
16:
             return v:
17:
         end if
18: end function
 1: function IFFT(\widehat{A})

    ← different from FFT

        return IFFT^*(\widehat{A})/\mathrm{length}(\widehat{A})
3: end function
```

Computer science books take the forward DFT operation to be the evaluation of the corresponding polynomial at all roots of unity $\omega_n^k = \cos \frac{2\pi k}{n} + i \sin \frac{2\pi k}{n}$ and the InverseDFT to be the evaluation of the polynomial at the complex conjugates of the roots of unity, i.e.,

A'ssignment Project Exam Help

polynomial at ω_n^{-k} and the InverseDFT at ω_n^k !

While for the transport of multiper collections are could good, the choice made by the transport of the collection of th

we did here only multiplication of polynomials, and did not apply it to multiplication of large integer. That does we careful because road of unity are represented Add polynomials to be careful because road of unity are represented Add polynomials to be careful because road of unity are represented Add polynomials. The hasto be careful because road of unity are represented Add polynomials, and did not apply it to multiplication of large integer.

Computer science books take the forward DFT operation to be the evaluation of the corresponding polynomial at all roots of unity $\omega_n^k = \cos \frac{2\pi k}{n} + i \sin \frac{2\pi k}{n}$ and the InverseDFT to be the evaluation of the polynomial at the complex conjugates of the roots of unity, i.e.,

A \$\sigma_{\text{sign}}^{\text{cos}} \frac{2\pi\k}{\text{pinent}} \frac{1}{\text{Project Exam Help}} \frac{\text{However, Excital engineering books do it just opposite, the direct DFT evaluates the}}{\text{Project Exam Help}}

However, Electrical engineering books do it just opposite, the direct DFT evaluates the polynomial at ω_n^{-k} and the InverseDFT at ω_n^{k} !

While for the proper of multiper extraornials bot debaies are couple good, the choice made by the third power of the couple good, the choice made by the third power of the couple good in the Advanced Aprithms 41 P when we do the JPEG.

large integer That dos We Chat powcoder in sufficient precision you can round off the results and beauty powcoder where the powcoder is sufficient precision you can round off the results and beauty correct integer values, but all of this is tricky.

Computer science books take the forward DFT operation to be the evaluation of the corresponding polynomial at all roots of unity $\omega_n^k = \cos \frac{2\pi k}{n} + i \sin \frac{2\pi k}{n}$ and the InverseDFT to be the evaluation of the polynomial at the complex conjugates of the roots of unity, i.e.,

Assignment Project Exam Help

However, Electrical engineering books do it just opposite, the direct DFT evaluates the polynomial at ω_n^{-k} and the InverseDFT at ω_n^k !

While for the purpose of multiplying polynomials both choices are equally good, the choice made by the licerital engineers is hull with a reliable purposes W vill explain this in the Advanced Algorithms 412, when we do the JPEG.

we did here only multiplication of polynomials, and did not apply it to multiplication of large integer. The solve of the has to be careful because roof of unity are represented Add polynome Chat the powcode in sufficient precision you can round off the results and brain correct integer values, but all of this is tricky.

Computer science books take the forward DFT operation to be the evaluation of the corresponding polynomial at all roots of unity $\omega_n^k = \cos \frac{2\pi k}{n} + i \sin \frac{2\pi k}{n}$ and the InverseDFT to be the evaluation of the polynomial at the complex conjugates of the roots of unity, i.e.,

Assignment Project Exam Help

However, Electrical engineering books do it just opposite, the direct DFT evaluates the polynomial at ω_n^{-k} and the InverseDFT at ω_n^k !

While for the purpose of multiplying polynomials both choices are equally good, the choice made by the licerial engineers is the hypertrum of the purposes W vill explain this in the Advanced Algorithms 412, when we do the JPEG.

We did here only multiplication of polynomials, and did not apply it to multiplication of large integers. This is possible to do purche has to be careful because roots of unity are represented by filed lig point in both so to the test with the sufficient precision you can round off the results and obtain correct integer values, but all of this is tricky.

Computer science books take the forward DFT operation to be the evaluation of the corresponding polynomial at all roots of unity $\omega_n^k = \cos \frac{2\pi k}{n} + i \sin \frac{2\pi k}{n}$ and the InverseDFT to be the evaluation of the polynomial at the complex conjugates of the roots of unity, i.e.,

Assignment Project Exam Hel

However, Electrical engineering books do it just opposite, the direct DFT evaluates the polynomial at ω_n^{-k} and the InverseDFT at ω_n^k !

While for the purpose of multiplying polynomials both choices are equally good, the choice made by the alcrital engineers is an howeter brief purposes W vill explain this in the Advanced Algorithms 4121 when we do the JPEG.

We did here only multiplication of polynomials, and did not apply it to multiplication of large integers. This is boss by to do but the has to be careful because roofs of unity are represented by floating point numbers so to hive to slow that if you de HIT with sufficient precision you can round off the results and obtain correct integer values, but all of this is tricky.

Back to fast multiplication of polynomials

$$P_A(x) = A_0 + A_1 x + \ldots + A_{n-1} x^{n-1}$$
 $P_B(x) = B_0 + B_1 x + \ldots + B_{n-1} x^{n-1}$

 $\underset{\{P_{A}(1),\,P_{A}(\omega_{2}^{2})_{1},\,P_{A}(\omega_{2n-1}^{2})_{1},\,\dots,\,P_{A}(\omega_{2n-1}^{2n-2})_{1}\}}{\text{Assign}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1}^{2})_{1},\,P_{B}(\omega_{2n-1}^{2})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1}^{2})_{1},\,P_{B}(\omega_{2n-1}^{2})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1}^{2})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1}^{2})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}}\underset{\{P_{B}(1),\,P_{B}(\omega_{2n-1})_{1},\,P_{B}(\omega_{2n-1})_{1}\}}{\text{Exam}$

 $\underset{\{P_A(1)P_B(1), P_A(\omega_{2n-1})P_B(\omega_{2n-1}), \dots, P_A(\omega_{2n-1}^{2n-2})P_B(\omega_{2n-1}^{2n-2})\}}{\text{multiplication } O(n)}$

Add We Chat powcoder $P_C(x) = \sum_{j=0}^{2n-2} \left(\sum_{i=0}^{j} A_i B_{j-i} \right) x^j = \sum_{j=0}^{2n-2} C_j x^j = P_A(x) \cdot P_B(x)$

$$P_C(x) = \sum_{j=0}^{2n-2} \left(\sum_{i=0}^{j} A_i B_{j-i} \right) x^j = \sum_{j=0}^{2n-2} C_j x^j = P_A(x) \cdot P_B(x)$$

Back to fast multiplication of polynomials

$$P_A(x) = A_0 + A_1 x + \ldots + A_{n-1} x^{n-1}$$
 $P_B(x) = B_0 + B_1 x + \ldots + B_{n-1} x^{n-1}$

 $\underbrace{ \text{Assignment}}_{\{P_A(1), P_A(\omega_2^2 1, 1), \dots, P_A(\omega_{2n-1}^{2n-2})\};} \underbrace{ \text{Pert}}_{\{P_B(1), P_B(\omega_{2n-1}), P_B(\omega_{2n-1}^{2n})\}} \underbrace{ \text{Pert}}_{O(n \log n)} \underbrace{ \text{Help}}_{O(n \log n)} \underbrace{ \text$

 $\underset{\{P_A(1)P_B(1), P_A(\omega_{2n-1})P_B(\omega_{2n-1}), \dots, P_A(\omega_{2n-1}^{2n-2})P_B(\omega_{2n-1}^{2n-2})\}}{\text{multiplication } O(n)}$

Add We Chat powcoder $P_C(x) = \sum_{j=0}^{2n-2} \left(\sum_{i=0}^{j} A_i B_{j-i} \right) x^j = \sum_{j=0}^{2n-2} C_j x^j = P_A(x) \cdot P_B(x)$

$$P_C(x) = \sum_{j=0}^{2n-2} \left(\sum_{i=0}^{j} A_i B_{j-i} \right) x^j = \sum_{j=0}^{2n-2} C_j x^j = P_A(x) \cdot P_B(x)$$

Thus, the product $P_C(x) = P_A(x) P_B(x)$ of two polynomials $P_A(x)$ and $P_B(x)$ can be computed in time $O(n \log n)$.

Computing the convolution C = A * B

 $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$

$$A^{A}Ssi_{\mathcal{C}_{n-1}}^{\bullet} \underbrace{ Pro_{n-1}^{\bullet} Pro_{j}^{\bullet} e^{-1} Pro_{j}^{\bullet} e^{-1} Pro_{j}^{\bullet} e^{-1} Pro_{j}^{\bullet} e^{-1} Pro_{n-1}^{\bullet} e^{-1} Pro_{n-1}^{\bullet}$$

 $B = \langle B_0, B_1, \dots, B_{n-1} \rangle$

{Phttps://powcoder.com

$$C = \left\langle \sum_{i=0}^{j} A_i B_{j-i} \right\rangle_{j=0}^{j=2n-2}$$

Convolution C = A * B of sequences A and B is computed in time $O(n \log n)$.



Computing the convolution C = A * B

 $A = \langle A_0, A_1, \dots, A_{n-1} \rangle$

 $B = \langle B_0, B_1, \dots, B_{n-1} \rangle$

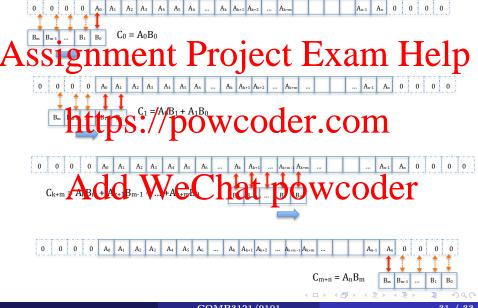
{Phttps://powcoder.com

$$C = \left\langle \sum_{i=0}^{j} A_i B_{j-i} \right\rangle_{j=0}^{j=2n-2}$$

Convolution C = A * B of sequences A and B is computed in time $O(n \log n)$.



Visualizing Convolution C = A * B



An Exercise

A0 A1

A2 A3 A4 ...

Assume you are given a map of a straight sea shore of length 100n meters as a sequence on 100n numbers such that A_i is the number of fish between ith meter of the shore and (i + 1)th meter, 0 ≤ i ≤ 100n − 1.
You also have a net of length noneters but unfortunately it has loles in bach a net stees or bed as a sequence N on none and zeros, where D 0's denote where the holes are. If you throw such a net starting at meter

0's denote where the holes are. If you throw such a net starting at meter k and ending at meter k+n, then you will catch only the fish in one meter stretches of the shore where the corresponding bit of the net is 1; see the figure $\frac{k}{5}$ DOWCOCET. Capital n

... A_{k+m-2} A_{k+m-1} ... A_{n-2} A_{n-1} 0

Add Wellhattpowcoder

 $A_k A_{k+1} A_{k+2} A_{k+3} A_{k+4}$

Find the spot where you should place the left end of your net in order to catch the largest possible number of fish using an algorithm which runs in time $O(n \log n)$.

Hint: Let N' be the net sequence N in the reverse order; Compute A * B' and look for the peak of that sequence.

PUZZLE!!

Assignment Project Exam Help On a circular highway there are n petrol stations, unevenly spaced, each

containing a different quantity of petrol. It is known that the total quantity of petrol on all stations is enough to go around the highway once and that the tank of your car can hold enough fuel to make a trip around the highway. Prove that there are exists a station among all of the stations on the highway, such that if you take it as a starting point and take the fuel from that station, you can continue to make a complete round trip around the highway, never emptying your tank before reaching the dext without or relief powcoder