Análise de Algoritmos

Estes slides são adaptações de slides

do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Aula 3

Transformada rápida de Fourier

Secs 30.1 e 30.2 do CLRS e 5.6 do KT.

Produto de polinômios

Problema: Dados dois polinômios

$a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ e

$b(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}$,

calcular o polinômio $p(x) = a(x) \cdot b(x)$.

Produto de polinômios

Problema: Dados dois polinômios
$a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ e
$b(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}$,
calcular o polinômio $p(x) = a(x) \cdot b(x)$.

Lembre-se que $p(x) = c_0 + c_1 x + \cdots + c_{2n-2} x^{2n-2}$, onde

$$c_k = a_o b_k + a_1 b_{k-1} + \cdots + a_k b_0,$$

para $k = 0, 1, \ldots, 2n-2$.

O vetor c é a convolução de a e b.

Produto de polinômios

Problema: Dados dois polinômios
$a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ e
$b(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}$,
calcular o polinômio $p(x) = a(x) \cdot b(x)$.

Há um algoritmo $\mathrm{O}(n^2)$ óbvio para calcular $p(x)$,
ou seja, para calcular $c_0, c_1, \ldots, c_{2n-2}$.

Queremos um algoritmo $\mathrm{O}(n \lg n)$.

Produto de polinômios

Problema: Dados dois polinômios
$a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ e
$b(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}$,
calcular o polinômio $p(x) = a(x) \cdot b(x)$.

Há um algoritmo $\mathrm{O}(n^2)$ óbvio para calcular $p(x)$,
ou seja, para calcular $c_0, c_1, \ldots, c_{2n-2}$.

Queremos um algoritmo $\mathrm{O}(n \lg n)$.

Representações alternativas de polinômios de grau $n - 1$:

- seus n coeficientes, ou
- seu valor em n pontos distintos.

Ideia do algoritmo $O(n \lg n)$

Entrada: $a = (a_0, \ldots, a_{n-1})$ e $b = (b_0, \ldots, b_{n-1})$.

- Obter pares

$$(x_0, y_0^a), \ldots, (x_{2n-2}, y_{2n-2}^a) \text{ e } (x_0, y_0^b), \ldots, (x_{2n-2}, y_{2n-2}^b)$$

onde $x_i \neq x_j$ para $i \neq j$, e

$y_i^a = a(x_i)$ e $y_i^b = b(x_i)$ para $i = 0, \ldots, 2n-2$.

Ideia do algoritmo $O(n \lg n)$

Entrada: $a = (a_0, \ldots, a_{n-1})$ e $b = (b_0, \ldots, b_{n-1})$.

- Obter pares

  $$(x_0, y_0^a), \ldots, (x_{2n-2}, y_{2n-2}^a) \text{ e } (x_0, y_0^b), \ldots, (x_{2n-2}, y_{2n-2}^b)$$

  onde $x_i \neq x_j$ para $i \neq j$, e

  $$y_i^a = a(x_i) \text{ e } y_i^b = b(x_i) \text{ para } i = 0, \ldots, 2n-2.$$

- Obter pares $(x_0, q_0), \ldots, (x_{2n-2}, q_{2n-2})$
  onde $q_i = y_i^a \cdot y_i^b$ para $i = 0, \ldots, 2n-2$.

Ideia do algoritmo $O(n \lg n)$

Entrada: $a = (a_0, \ldots, a_{n-1})$ e $b = (b_0, \ldots, b_{n-1})$.

- Obter pares

  $$(x_0, y^a_0), \ldots, (x_{2n-2}, y^a_{2n-2}) \text{ e } (x_0, y^b_0), \ldots, (x_{2n-2}, y^b_{2n-2})$$

  onde $x_i \neq x_j$ para $i \neq j$, e

  $$y^a_i = a(x_i) \text{ e } y^b_i = b(x_i) \text{ para } i = 0, \ldots, 2n-2.$$

- Obter pares $(x_0, q_0), \ldots, (x_{2n-2}, q_{2n-2})$
  onde $q_i = y^a_i \cdot y^b_i$ para $i = 0, \ldots, 2n-2$.

- Determinar $q(x)$ tal que $q(x_i) = q_i$ para $i = 0, \ldots, 2n-2$.

Ideia do algoritmo $\mathrm{O}(n \lg n)$

Entrada: $a = (a_0, \ldots, a_{n-1})$ e $b = (b_0, \ldots, b_{n-1})$.

- Obter pares

  $$(x_0, y_0^a), \ldots, (x_{2n-2}, y_{2n-2}^a) \text{ e } (x_0, y_0^b), \ldots, (x_{2n-2}, y_{2n-2}^b)$$

  onde $x_i \neq x_j$ para $i \neq j$, e

  $$y_i^a = a(x_i) \text{ e } y_i^b = b(x_i) \text{ para } i = 0, \ldots, 2n-2.$$

- Obter pares $(x_0, q_0), \ldots, (x_{2n-2}, q_{2n-2})$
  onde $q_i = y_i^a \cdot y_i^b$ para $i = 0, \ldots, 2n-2$.

- Determinar $q(x)$ tal que $q(x_i) = q_i$ para $i = 0, \ldots, 2n-2$.

Vimos como fazer o primeiro passo em tempo $\mathrm{O}(n \lg n)$.

Ideia do algoritmo $\mathrm{O}(n \lg n)$

Entrada: $a = (a_0, \ldots, a_{n-1})$ e $b = (b_0, \ldots, b_{n-1})$.

- Obter pares

  $$(x_0, y^a_0), \ldots, (x_{2n-2}, y^a_{2n-2}) \text{ e } (x_0, y^b_0), \ldots, (x_{2n-2}, y^b_{2n-2})$$

  onde $x_i \neq x_j$ para $i \neq j$, e

  $$y^a_i = a(x_i) \text{ e } y^b_i = b(x_i) \text{ para } i = 0, \ldots, 2n-2.$$

- Obter pares $(x_0, q_0), \ldots, (x_{2n-2}, q_{2n-2})$
  onde $q_i = y^a_i \cdot y^b_i$ para $i = 0, \ldots, 2n-2$.

- Determinar $q(x)$ tal que $q(x_i) = q_i$ para $i = 0, \ldots, 2n-2$.

Vimos como fazer o primeiro passo em tempo $\mathrm{O}(n \lg n)$.
O passo do meio consome tempo $\mathrm{O}(n)$ trivialmente.

Raízes da unidade

São definidas para cada n.

Raízes da unidade

São definidas para cada n.

Seja $\omega_n = e^{2\pi i/n}$.

Raízes n-ésimas da unidade: para $k = 0, 1, \ldots, n-1$,

$$\omega_n^k = e^{2\pi ki/n}.$$

Raízes da unidade

São definidas para cada n.

Seja $\omega_n = e^{2\pi i/n}$.

Raízes n-ésimas da unidade: para $k = 0, 1, \ldots, n-1$,

$$\omega_n^k = e^{2\pi k i/n}.$$

Lembre-se que $e^{\theta\, i} = \cos(\theta) + i\,\mathrm{sen}(\theta)$.

Raízes da unidade

São definidas para cada n.

Seja $\omega_n = e^{2\pi i/n}$.

Raízes n-ésimas da unidade: para $k = 0, 1, \ldots, n-1$,

$$\omega_n^k = e^{2\pi k i/n}.$$

Lembre-se que $e^{\theta\, i} = \cos(\theta) + i\,\mathrm{sen}(\theta)$.

Notação conveniente pois o produto dos números complexos $\cos(\theta_1) + i\,\mathrm{sen}(\theta_1)$ e $\cos(\theta_2) + i\,\mathrm{sen}(\theta_2)$ é

$$\begin{aligned}(\cos(\theta_1)\cos(\theta_2) - \mathrm{sen}(\theta_1)\mathrm{sen}(\theta_2)) &+ i\,(\cos(\theta_1)\mathrm{sen}(\theta_2) + \cos(\theta_2)\mathrm{sen}(\theta_1)) \\ = \cos(\theta_1+\theta_2) &+ i\,\mathrm{sen}(\theta_1+\theta_2).\end{aligned}$$

Raízes da unidade

São definidas para cada n.

Seja $\omega_n = e^{2\pi i/n}$.

Raízes n-ésimas da unidade: para $k = 0, 1, \ldots, n-1$,

$$\omega_n^k = e^{2\pi k i/n}.$$

Lembre-se que $e^{\theta\, i} = \cos(\theta) + i\,\mathrm{sen}(\theta)$.

Notação conveniente pois o produto dos números complexos $\cos(\theta_1) + i\,\mathrm{sen}(\theta_1)$ e $\cos(\theta_2) + i\,\mathrm{sen}(\theta_2)$ é

$$\begin{aligned}(\cos(\theta_1)\cos(\theta_2) - \mathrm{sen}(\theta_1)\mathrm{sen}(\theta_2)) &+ i\,(\cos(\theta_1)\mathrm{sen}(\theta_2) + \cos(\theta_2)\mathrm{sen}(\theta_1)) \\ = \cos(\theta_1+\theta_2) &+ i\,\mathrm{sen}(\theta_1+\theta_2).\end{aligned}$$

Ou seja, $e^{\theta_1 i} e^{\theta_2 i} = e^{(\theta_1+\theta_2)i}$.

Transformada discreta de Fourier

Raízes n-ésimas da unidade: para $k = 0, 1, \ldots, n-1$,

$$\omega_n^k = e^{2\pi k i/n}.$$

Dado um vetor $a = (a_0, a_1, \ldots, a_{n-1})$, representando os coeficientes de um polinômio que denotamos por $a(x)$, a Transformada Discreta de Fourier (DFT) de ordem n de a é o vetor $y = (y_0, y_1, \ldots, y_{n-1})$ onde $y_k = a(\omega_n^k)$ para $k = 0, 1, \ldots, n-1$.

Transformada discreta de Fourier

Raízes n-ésimas da unidade: para $k = 0, 1, \ldots, n-1$,

$$\omega_n^k = e^{2\pi k i/n}.$$

Dado um vetor $a = (a_0, a_1, \ldots, a_{n-1})$, representando os coeficientes de um polinômio que denotamos por $a(x)$, a Transformada Discreta de Fourier (DFT) de ordem n de a é o vetor $y = (y_0, y_1, \ldots, y_{n-1})$ onde $y_k = a(\omega_n^k)$ para $k = 0, 1, \ldots, n-1$.

Objetivo: programa que, dado um vetor $a = (a_0, \ldots, a_{n-1})$, determina a sua DFT de ordem n em tempo $\Theta(n \lg n)$.

Transformada discreta de Fourier

Raízes n-ésimas da unidade: para $k = 0, 1, \ldots, n-1$,

$$\omega_n^k = e^{2\pi ki/n}.$$

Dado um vetor $a = (a_0, a_1, \ldots, a_{n-1})$, representando os coeficientes de um polinômio que denotamos por $a(x)$, a Transformada Discreta de Fourier (DFT) de ordem n de a é o vetor $y = (y_0, y_1, \ldots, y_{n-1})$ onde $y_k = a(\omega_n^k)$ para $k = 0, 1, \ldots, n-1$.

Objetivo: programa que, dado um vetor $a = (a_0, \ldots, a_{n-1})$, determina a sua DFT de ordem n em tempo $\Theta(n \lg n)$.

Essa é a chamada Transformada Rápida de Fourier (FFT).

Transformada Rápida de Fourier

```
FFT (a, n)                                          ▷ n é uma potência de 2
 1   se n = 1 então devolva a
 2   a^0 ← (a_0, a_2, ..., a_{n-2})
 3   a^1 ← (a_1, a_3, ..., a_{n-1})
 4   y^0 ← FFT (a^0, n/2)
 5   y^1 ← FFT (a^1, n/2)
 6   ω_n ← e^{2πi/n}
 7   ω ← 1
 8   para k ← 0 até n/2 − 1 faça
 9         y_k ← y^0_k + ω y^1_k
10         y_{k+n/2} ← y^0_k − ω y^1_k
11         ω ← ω ω_n
12   devolva y
```

Consumo de tempo: $T(n) = 2T(n/2) + \Theta(n) = \Theta(n \lg n)$.

Voltando ao produto de polinômios...

Queremos...

- Para $x_0, \ldots, x_{2n-2}$ distintos, calcular
  $y_i^a = a(x_i)$ e $y_i^b = b(x_i)$ para $i = 0, \ldots, 2n-2$.
- Calcular $q_i = y_i^a \cdot y_i^b$ para $i = 0, \ldots, 2n-2$.
- Determinar $q(x)$ tal que $q(x_i) = q_i$ para $i = 0, \ldots, 2n-2$.

Voltando ao produto de polinômios...

Queremos...

- Para $x_0, \ldots, x_{2n-2}$ distintos, calcular
  $y_i^a = a(x_i)$ e $y_i^b = b(x_i)$ para $i = 0, \ldots, 2n-2$.
- Calcular $q_i = y_i^a \cdot y_i^b$ para $i = 0, \ldots, 2n-2$.
- Determinar $q(x)$ tal que $q(x_i) = q_i$ para $i = 0, \ldots, 2n-2$.

Primeira etapa:
dados vetores $a = (a_0, \ldots, a_{n-1})$ e $b = (b_0, \ldots, b_{n-1})$,
estendemos tais vetores adicionando n zeros em cada um,
obtendo $a = (a_0, \ldots, a_{2n-1})$ e $b = (b_0, \ldots, b_{2n-1})$,
e calculamos $y^a = \mathrm{FFT}_{2n}(a)$ e $y^b = \mathrm{FFT}_{2n}(b)$.

Voltando ao produto de polinômios...

Queremos...

- Para $x_0, \ldots, x_{2n-2}$ distintos, calcular
  $y_i^a = a(x_i)$ e $y_i^b = b(x_i)$ para $i = 0, \ldots, 2n-2$.
- Calcular $q_i = y_i^a \cdot y_i^b$ para $i = 0, \ldots, 2n-2$.
- Determinar $q(x)$ tal que $q(x_i) = q_i$ para $i = 0, \ldots, 2n-2$.

Primeira etapa:
dados vetores $a = (a_0, \ldots, a_{n-1})$ e $b = (b_0, \ldots, b_{n-1})$,
estendemos tais vetores adicionando n zeros em cada um,
obtendo $a = (a_0, \ldots, a_{2n-1})$ e $b = (b_0, \ldots, b_{2n-1})$,
e calculamos $y^a = \mathrm{FFT}_{2n}(a)$ e $y^b = \mathrm{FFT}_{2n}(b)$.

Segunda etapa: óbvia...

Voltando ao produto de polinômios...

Queremos...

- Para $x_0, \ldots, x_{2n-2}$ distintos, calcular
  $y_i^a = a(x_i)$ e $y_i^b = b(x_i)$ para $i = 0, \ldots, 2n-2$.
- Calcular $q_i = y_i^a \cdot y_i^b$ para $i = 0, \ldots, 2n-2$.
- Determinar $q(x)$ tal que $q(x_i) = q_i$ para $i = 0, \ldots, 2n-2$.

Primeira etapa:
dados vetores $a = (a_0, \ldots, a_{n-1})$ e $b = (b_0, \ldots, b_{n-1})$,
estendemos tais vetores adicionando n zeros em cada um,
obtendo $a = (a_0, \ldots, a_{2n-1})$ e $b = (b_0, \ldots, b_{2n-1})$,
e calculamos $y^a = \mathrm{FFT}_{2n}(a)$ e $y^b = \mathrm{FFT}_{2n}(b)$.

Segunda etapa: óbvia...

Terceira etapa: interpolação.

Interpolação

Primeira etapa: dado um vetor $a = (a_0, \ldots, a_{n-1})$, calcular $y = (y_0, \ldots, y_{n-1})$ tal que $y = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}) \cdot a$.

Interpolação

Primeira etapa: dado um vetor $a = (a_0, \ldots, a_{n-1})$, calcular $y = (y_0, \ldots, y_{n-1})$ tal que $y = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}) \cdot a$.

A matriz $V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})$ é a matriz de Vandermond para $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$.

Interpolação

Primeira etapa: dado um vetor $a = (a_0, \ldots, a_{n-1})$, calcular $y = (y_0, \ldots, y_{n-1})$ tal que $y = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}) \cdot a$.

A matriz $V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})$ é a matriz de Vandermond para $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$.

Terceira etapa: dado um vetor $y = (y_0, \ldots, y_{n-1})$, calcular $q = (q_0, \ldots, q_{n-1})$ tal que $q = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})^{-1} \cdot y$.

Interpolação

Primeira etapa: dado um vetor $a = (a_0, \ldots, a_{n-1})$, calcular $y = (y_0, \ldots, y_{n-1})$ tal que $y = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}) \cdot a$.

A matriz $V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})$ é a matriz de Vandermond para $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$.

Terceira etapa: dado um vetor $y = (y_0, \ldots, y_{n-1})$, calcular $q = (q_0, \ldots, q_{n-1})$ tal que $q = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})^{-1} \cdot y$.

Como calcular q em tempo $\mathrm{O}(n \lg n)$?

Interpolação

Primeira etapa: dado um vetor $a = (a_0, \ldots, a_{n-1})$, calcular $y = (y_0, \ldots, y_{n-1})$ tal que $y = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}) \cdot a$.

A matriz $V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})$ é a matriz de Vandermond para $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$.

Terceira etapa: dado um vetor $y = (y_0, \ldots, y_{n-1})$, calcular $q = (q_0, \ldots, q_{n-1})$ tal que $q = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})^{-1} \cdot y$.

Como calcular q em tempo $\mathrm{O}(n \lg n)$?

Teorema: A inversa da matriz de Vandermond $V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})$ tem na posição (j, k) o valor ω_n^{-jk}/n para $j, k = 0, 1, \ldots, n-1$

Prova feita na aula.

Interpolação

Teorema: A inversa da matriz de Vandermond $V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})$ tem na posição (j,k) o valor ω_n^{-jk}/n para j, $k = 0, 1, \ldots, n-1$

Ou seja,

$$V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})^{-1} = \frac{1}{n} V(\omega_n^0, \omega_n^{-1}, \ldots, \omega_n^{-(n-1)}).$$

Interpolação

Teorema: A inversa da matriz de Vandermond $V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})$ tem na posição (j,k) o valor ω_n^{-jk}/n para j, $k = 0, 1, \ldots, n-1$

Ou seja,

$$V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})^{-1} = \frac{1}{n} V(\omega_n^0, \omega_n^{-1}, \ldots, \omega_n^{-(n-1)}).$$

Exercício:
Modifique o algoritmo FFT para fazer a interpolação.

Inversa da DFT

Como bem notado por alguns alunos na aula,

$$V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}) = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & & & & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{pmatrix}$$

é uma matriz simétrica e, como o conjugado complexo de $e^{\theta i}$ é exatamente $e^{-\theta i}$, temos que

$$V(\omega_n^0, \omega_n^{-1}, \ldots, \omega_n^{-(n-1)}) = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})^*$$

Inversa da DFT

Como $V(\omega_n^0, \omega_n^{-1}, \ldots, \omega_n^{-(n-1)}) = V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})^*$, então podemos calcular

$$q \;=\; \frac{1}{n}\, V(\omega_n^0, \omega_n^{-1}, \ldots, \omega_n^{-(n-1)})\, y$$

calculando

$$q^* \;=\; y^*\, \frac{1}{n}\, V(\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1})$$

onde y' é o vetor $(y^*)^T$ (o vetor y com cada coordenada trocada pelo seu conjugado), e o vetor q é obtido de q' analogamente.

Ou seja, podemos usar a própria função FFT para calcular q a partir de y.

Aplicações de FFT

- Processamento de imagens
  - suavização da imagem
  - remoção de ruído
  - destaque de contornos

Aplicações de FFT

- Processamento de imagens
  - suavização da imagem
  - remoção de ruído
  - destaque de contornos
- Música
  - equalizadores
  - reconhecimento de notas

Aplicações de FFT

- Processamento de imagens
  - suavização da imagem
  - remoção de ruído
  - destaque de contornos
- Música
  - equalizadores
  - reconhecimento de notas
- Combinação de histogramas