Insegnamento di algoritmi avanzati

Paradigma *divide et impera*

Roberto Posenato

ver. 0.5, 01/10/2009

Sommario

Introduzione

In questa lezione si richiamano i concetti fondamentali del paradigma *divide et impera*, già studiato e usato nel corso di Algoritmi e strutture dati.

Idea generale

Il paradigma *divide et impera* prevede di risolvere un problema eseguendo i seguenti passi:

1. Suddividere l'istanza in sottoistanze del medesimo problema di dimensioni inferiori;
2. Risolvere ricorsivamente queste istanze;
3. Combinare i risultati ottenuti in modo appropriato per determinare la soluzione;

Aspetti cruciali per una buona applicazione:

- Suddivisione: deve determinare sottoistanze il più possibile disgiunte;
- Ricorsione: deve terminare quando la dimensione permette una risoluzione diretta;
- Combinazione: non deve richiedere più tempo rispetto al calcolo soluzioni sottoistanze.

Dettagli

Pseudocodifica

Il paradigma *divide et impera* può essere così schematizzato:

Procedura `DI`(x)

if $(|x| \leq n_0)$ **then return** `adhoc`(x); // n_0 =costante dell'algoritmo!
dividi x in sottoistanze più piccole $x_1, x_2, \ldots x_a$;
for $(i = 1; \quad i \leq a; \quad i{+}{+})$ **do**
    y_i =`DI`(x_i);
endfor
ricombina gli y_i in una soluzione y;
return y

Dettagli

Considerazioni

- La funzione `adhoc(`x`)`:
  - calcola la soluzione in modo diretto;
  - complessità $\approx$ ordine lineare o costante.
- dividi *dovrebbe*:
  - suddividere l'istanza in b>1 sottoistanze "disgiunte" di dimensione n/b ciascuna.
  - avere complessità $\approx$ lineare.
- Il numero di sottoistanze a:
  - solitamente piccolo: $[2, \ldots, 4]$ e diverso da b.
  - $a = 1 \Rightarrow$ si chiama "semplificazione".
- ricombina ha solitamente di complessità $O(n^d)$ per un d costante.

Dettagli

Costo computazionale

Costo algoritmo *divide et impera* = soluzione di un'equazione di ricorrenza:

- Sia tempo(`adhoc(x)`) $= g(|x|)$;
- Sia tempo(dividi(x)) $+$ tempo(combina($y_1, \ldots, y_a$)) $= f(|x|)$;
- $T(n) =$ tempo(`DI(x)`). Posto $n = |x|$,

$$T(n) = \begin{cases} g(n) & n \leq n_0, \\ aT(n/b) + f(n) & \text{altrimenti.} \end{cases}$$

- Per la risoluzione di questa equazione si ricorre ai soliti metodi: *per sostituzione*, *albero di ricorsione* o *metodo dell'esperto*.

Dettagli

Costo computazionale: calcolo mediante teorema dell'esperto

Teorema 1 (Teorema dell'esperto)

Date le costanti $a \geq 1$, $b > 1$ *e la funzione* $f(n)$, *la ricorrenza*

$$T(n) = aT(n/b) + f(n)$$

può essere limitata asintoticamente come:

1. $f(n) = O(n^{\log_b a - \varepsilon})$ *per almeno un* $\varepsilon > 0 \Rightarrow \; T(n) = \Theta(n^{\log_b a})$.
2. $f(n) = \Theta(n^{\log_b a}) \Rightarrow \; T(n) = \Theta(n^{\log_b a} \log_b n)$.
3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ *per almeno un* $\varepsilon > 0$ *e se* $af(n/b) \leq cf(n)$ *per almeno un* $c < 1$ *e per ogni* n *sufficientemente grande* $\Rightarrow \; T(n) = \Theta(f(n))$.

Nota!

$n^{\log_b a} = a^{\log_b n}$ è il numero di foglie dell'albero di ricorsione!
Il teorema assume che il costo di calcolo di un'istanza-foglia $= O(1)$.

Dettagli

Costo computazionale

È possibile sostituire la ricorsione con un ciclo iterativo.

Vantaggi: risparmio

- di un fattore moltiplicativo in tempo.
- di un fattore $[\Omega(\log n), \ldots, \Omega(n)]$ in spazio.

Svantaggi: l'algoritmo diviene un po' più complicato da scrivere.

Esempi di applicazione

Prodotto

PROBLEMA PRODOTTO NUMERI

DESCRIZIONE: Due interi x, y di n bit ciascuno.

QUESITO: Determinare il loro prodotto.

- Il costo del prodotto tradizionale $= O(n^2)$;

Nota!

Il costo è riferito quando si devono moltiplicare due numeri con $n >> 64$ bit e si usa il metodo tradizionale!
Se $n \leq 64$, i moderni calcolatori permettono il calcolo diretto!

Esempi di applicazione
Prodotto

Si può fare meglio?

- Assumiamo, per ora, che $n = 2^k$.
- Si suddivida $x = \boxed{x_L}\boxed{x_R} = 2^{\frac{n}{2}}x_L + x_R$.
- Lo stesso per $y = 2^{\frac{n}{2}}y_L + y_R$;
- Il prodotto

$$\begin{aligned} xy &= (2^{\frac{n}{2}}x_L + x_R) \times (2^{\frac{n}{2}}y_L + y_R) \\ &= 2^n x_L \times y_L + 2^{\frac{n}{2}}(x_L \times y_R + x_R \times y_L) + x_R \times y_R \end{aligned}$$

  richiede 4 moltiplicazioni (su fattori di dimensione $O(\frac{n}{2})$) + 3 addizioni (su addendi di dimensione lineare) + 2 shift (su termini di dimensione lineare).
- Complessità in tempo $T(n) = 4T(\frac{n}{2}) + O(n) = O(n^2) \Leftarrow$ Tutto inutile?

Esempi di applicazione
Prodotto

Gauss (1777-1855) ha osservato che il prodotto di 2 numeri complessi

$$(a + bi) \times (c + di) = a \times c - b \times d + (b \times c + a \times d)i$$

può essere eseguito con 3 moltiplicazioni anziché 4:

$$a \times c, b \times d, (a + b) \times (c + d)$$

dato che $b \times c + a \times d = (a + b) \times (c + d) - a \times c - b \times d$.

Applicando lo stesso principio al prodotto di xy, si ottiene che è sufficiente calcolare

$$x_L \times y_L, \quad x_R \times y_R, \quad (x_L + x_R) \times (y_L + y_R)$$

Esempi di applicazione

Algoritmo per il prodotto

Procedura `prodotto`(x, y)

if *x è di 1 bit* **then return** xy; // 1 è per comodità...
$x_L = \lceil \frac{n}{2} \rceil$ bit più a sx, $x_R = \lceil \frac{n}{2} \rceil$ bit più a dx di x;
$y_L = \lceil \frac{n}{2} \rceil$ bit più a sx, $y_R = \lceil \frac{n}{2} \rceil$ bit più a dx di y;
$P_1 = \texttt{prodotto}(x_L, y_L)$;
$P_2 = \texttt{prodotto}(x_R, y_R)$;
$P_3 = \texttt{prodotto}(x_L + x_R, y_L + y_R)$;
return $(P_1 << n) + ((P_3 - P_1 - P_2) << \frac{n}{2}) + P_2$

Complessità

$$\begin{aligned} T(n) &\leq 3T(\frac{n}{2} + 1) + 2n + 6n + \frac{n}{2} \\ &\approx 3T(\frac{n}{2}) + O(n) = O(n^{\log 3}) = O(n^{1.59}). \end{aligned}$$

Esempi di applicazione

Prodotto di matrici

PROBLEMA PRODOTTO DI MATRICI

DESCRIZIONE: Due matrici X, Y di dimensioni $n \times n$.

QUESITO: Determinare il prodotto $Z = XY$.

- Considerato $z_{i,j} = \sum_{k=1}^{n} x_{i,k} y_{k,j}$, il costo in tempo è $O(n^3)$.
- Nel 1969 Strassen ha proposto un algoritmo più efficiente basato sul paradigma *divide et impera*.

Esempi di applicazione

Prodotto di matrici

- Si suddividano le due matrici in 4 blocchi ciascuna di dimensione $\frac{n}{2} \times \frac{n}{2}$:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}.$$

- Il prodotto diviene quindi:

$$XY = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}.$$

- Ancora una volta... se si applicasse il *divide et impera* usando questa decomposizione... non si otterrebbe nulla:

$$T(n) = 8T(\frac{n}{2}) + O(n^2) = O(n^{\log 8}) = O(n^3)$$

Esempi di applicazione
Prodotto di matrici

Metodo di Strassen:

- Senza entrare nei dettagli, Strassen ha osservato che

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}.$$

dove

$$\begin{aligned} P_1 &= A(F - H) & P_5 &= (A + D)(E + H) \\ P_2 &= (A + B)H & P_6 &= (B - D)(G + H) \\ P_3 &= (C + D)E & P_7 &= (A - C)(E + F) \\ P_4 &= D(G - E) \end{aligned}$$

- Ci sono 7 sottoproblemi di dimensione $\frac{n}{2}$ + 18 addizioni di matrici $\frac{n}{2} \times \frac{n}{2}$ (costo $O(n^2)$):

$$T(n) = 7T(\frac{n}{2}) + O(n^2) = O(n^{\log 7}) \approx O(n^{2.81})$$

Esempi di applicazione

Metodo di Strassen: critica

Da un punto di vista pratico, l'algoritmi di Strassen non è il metodo migliore:

- Fattore moltiplicativo nascosto di $O(n^{2.81})$ è decisamente più grande del fattore dell'algoritmo semplice;
- La stabilità numerica dell'algoritmo di Strassen è inferiore a quella dell'algoritmo semplice in alcune applicazioni;

Nella pratica, gli algoritmi veloci per moltiplicare matrici

- Usano Strassen solo per matrici dense e quando la loro dimensione supera un valore di transizione (*transition point*)
- Sotto tale valore, usano il metodo tradizionale;
- Il livello di transizione dipende dal sistema; sembra però che 20 sia un buon valore;

Esempi di applicazione

Metodo di Strassen: critica

- Miglior algoritmo per la moltiplicazioni di matrici è di Coppersmith e Winograd (1990).
- Rielaborazione dell'algoritmi di Strassen. Complessità $O(n^{2.376})$.
- Il termine costante nascosto da $O()$ è molto più grande.
- Knuth sostiene che tale algoritmo è di fatto applicabile per matrici di dimensioni non attualmente gestibili da computer di medie dimensioni (1998).

Esempi di applicazione

Metodo di Strassen

Esercizio 1 (Cormen 28.2-3)

Qual è il più grande valore di k tale che, se è possibile moltiplicare delle matrici 3×3 effettuando k moltiplicazioni (senza considerare la commutatività della moltiplicazione), allora è possibile moltiplicare delle matrici $n \times n$ nel tempo $o(n^{\log_2 7})$?

Esempi di applicazione

Prodotto di polinomi

Dopo aver applicato il *divide et impera* su prodotto di interi e di matrici... consideriamo ora il calcolo del prodotto di polinomi.

- Dati
  $A(x) = a_0 + a_1 x + \cdots + a_d x^d$,
  $B(x) = b_0 + b_1 x + \cdots + b_d x^d$, il loro prodotto
  $C(x) = A(x)B(x) = c_0 + c_1 x + \cdots + c_{2d} x^{2d}$ ha coefficienti

  $$c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_k b_0 = \sum_{i=0}^{k} a_i b_{k-i} \quad k = 0, \ldots, 2d$$

  (per $i > d$, si ponga $a_i = b_i = 0$).
- Determinare i $2d + 1$ coefficienti costa $\sum_{k=0}^{2d} O(k) = O(d^2)$

Esempi di applicazione

Polinomi: rappresentazione

Rappresentazione per coefficienti

- Ogni polinomio $A(x)$ di grado d è rappresentato come $\sum_{j=0}^{d} a_j x^j$
- I coefficienti $(a_0, \ldots, a_d)$ identificano il polimonio;
- La valutazione di un polinomio in un dato punto x_0 richiede tempo $\Theta(d)$ (Regola di Horner):

$$A(x_0) = a_0 + x_0(a_1 + x_0(a_2 + \cdots + x_0(a_{d-1} + x_0(a_d)) \cdots))$$

- La somma di due polinomi di grado d richiede tempo $\Theta(d)$;
- La moltiplicazione di due polinomi di grado d, invece, richiede $\Theta(d^2)$ se si usa il metodo della pagina precedente.

Esempi di applicazione

Polinomi: rappresentazione

Rappresentazione per punti

- Ogni $A(x)$ di grado d è rappresentato come un vettore di $d+1$ coppie $\{(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)\}$, dove x_i sono distinti e $y_k = A(x_k)$ per $k = 0, 1, \ldots, d$ (vedi teorema pagina seguente);
- La valutazione di un polinomio in un dato punto x_0 richiede di passare per la rappresentazione per coefficienti.
- La somma di due polinomi di grado d richiede tempo $\Theta(d)$: è sufficiente usare gli stessi x_i e sommare y_i^A e y_i^B per ciascun i;
- La moltiplicazione di due polinomi di grado d richiede tempo $\Theta(d)$: operazione analoga alla somma.
- Il grado di $C(x) = A(X)B(X)$ però è $2d$. Per moltiplicare necessari $2d+1$ punti distinti x_i sia per A sia per B.

Esempi di applicazione

Polinomi: rappresentazione

Teorema 2 (Univocità dei polinomi)

Un polinomio di grado d è univocamente caratterizzato dai suoi valori in $d+1$ punti distinti.

Dimostrazione.

Siano $x_0, x_1, \ldots, x_d$ $d+1$ punti distinti e $y_k = A(x_k)$ per $k = 0, 1, \ldots, d$ i valori del polinomio in questi punti. I valori $(y_0, y_1, \ldots, y_d)$ possono essere rappresentati come

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^d \\ 1 & x_1 & x_1^2 & \cdots & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_d & x_d^2 & \cdots & x_d^d \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_d \end{pmatrix}$$

La matrice è detta di Vandermonde e ha determinante $\prod_{0 \le j < k \le d}(x_k - x_j)$. Quindi non è singolare se i punti sono distinti. $\square$

Esempi di applicazione

Polinomi: rappresentazione

Conversioni di rappresentazione

- Se si ha una rappresentazione per coefficienti, quella per punti si determina in tempo $\Theta(d^2)$
- Se si ha una rappresentazione per valori, quella per coefficienti si determina per interpolazione usando il teorema di univocità. Tempo richiesto: $O(d^3)$
- Usando la formula di Lagrange

$$A(x) = \sum_{k=0}^{d} y_k \frac{\prod_{j \neq k}(x - x_j)}{\prod_{j \neq k}(x_k - x_j)}$$

si determinano i coefficienti in tempo $\Theta(d^2)$

Esempi di applicazione

Polinomi: strada alternativa per il prodotto

Il polinomio $C(x) = A(x)B(x)$ è quindi caratterizzabile univocamente

1. fissando $2d + 1$ punti distinti z_i (tempo $= \Theta(d)$).
2. valutando $A(z_i)$ e $B(z_i)$ per ciascun z_i (tempo $= \Theta(d^2)$).
3. moltiplicando $A(z_i)$ e $B(z_i)$ per ciascun z_i (tempo $= \Theta(d)$).
4. interpolando i valori ottenuti per determinare i coefficienti $c_0, \ldots, c_{2d}$ (tempo $= \Theta(d^2)$).

Il costo di questa caratterizzazione è $\Theta(d^2)$ nella migliore dell'ipotesi!
La trasformata veloce di Fourier permette di svolgere sia il passo 2 sia il 4 in tempo $\Theta(d \log d) \implies$ tempo calcolo prodotto $= \Theta(d \log d)$

Esempi di applicazione

Prodotto di polinomi: riorganizzazione di un polinomio per la valutazione

- Sia $n = 2^k$, per un k costante, il grado limite di un polinomio $A(x) = \sum_{i=0}^{n-1} a_i x^i$
- Siano $A_e(y)$ e $A_o(y)$ 2 polinomi di grado limite $\frac{n}{2}$ definiti usando i coefficienti pari (even) e dispari (odd) di $A()$

$$A_e(y) = a_0 + a_2 y + a_4 y^2 + \cdots + a_{n-2} y^{\frac{n}{2}-1}$$
$$A_o(y) = a_1 + a_3 y + a_5 y^2 + \cdots + a_{n-1} y^{\frac{n}{2}-1}$$

- Ne consegue che

$$A(x) = A_e(x^2) + x A_o(x^2)$$

Esempi di applicazione

Prodotto di polinomi: riorganizzazione di un polinomio per la valutazione

Esempio 1 (Riorganizzazione di un polinomio)

Sia $A(x) = 3 + 4x + 6x^2 + 2x^3 + x^4 + 10x^5$.

- grado limite deve essere potenza di 2
  $\Longrightarrow n = 8, A(x) = 3 + 4x + 6x^2 + 2x^3 + x^4 + 10x^5 + 0x^6 + 0x^7$.
- $A_e(y) = 3 + 6y + y^2 + 0y^3$.
- $A_o(y) = 4 + 2y + 10y^2 + 0y^3$.
- $A(x) = A_e(x^2) + xA_o(x^2) =$
  $3 + 6x^2 + x^4 + 0x^6 + x(4 + 2x^2 + 10x^4 + 0x^6)$.

Esempi di applicazione

Prodotto di polinomi: riorganizzazione di un polinomio per la valutazione

- Si scelgano $\frac{n}{2}$ coppie $\pm x_0, \pm x_1, \ldots \pm x_{\frac{n}{2}-1}$.
- Data una coppia $\pm x_i$, il calcolo di $A(x_i)$ può essere riciclato per il calcolo di $A(-x_i)$:

$$y = x_i^2 = -x_i^2$$
$$A(x_i) = A_e(y) + x_i \cdot A_o(y)$$
$$A(-x_i) = A_e(y) - x_i \cdot A_o(y)$$

- Valutare $A()$ in n punti può essere ridotto a valutare $A_e()$ e $A_o()$ in $\frac{n}{2}$ punti $x_0^2, \ldots, x_{\frac{n}{2}-1}^2$.
- Il costo diviene $T(n) = 2T(\frac{n}{2}) + O(n) = O(n \log n)$!
- Questo permetterebbe di risolvere la fase di valutazione in modo efficiente!

Esempi di applicazione

Prodotto di polinomi: valutazione *divide et impera*

Riorganizzazione di un polinomio (2)

Sia $A(x) = 3 + 4x + 6x^2 + 2x^3 + x^4 + 10x^5 + 0x^6 + 0x^7$ e $y = x^2$.

$$A(x) = (3 + 6x^2 + x^4 + 0x^6) + x(4 + 2x^2 + 10x^4 + 0x^6)$$

$$A_e(y) = 3 + 6y + y^2 + 0y^3$$

$$A_o(y) = 4 + 2y + 10y^2 + 0y^3$$

Si consideri la coppia $x = \pm 2, A(2) = 387, A(-2) = -301$:
$y = 4$

$$A(2) = A_e(4) + 2 \cdot A_o(4) = 43 + 2 \cdot 172 = 387$$

$$A(-2) = A_e(4) - 2 \cdot A_o(4) = 43 - 2 \cdot 172 = -301$$

Esempi di applicazione

Prodotto di polinomi: valutazione *divide et impera*

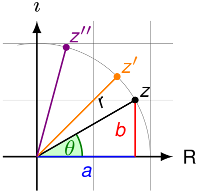
- I punti $x_0^2, \ldots, x_{\frac{n}{2}-1}^2$, alla prima chiamata ricorsiva, **non** sono coppie di valori opposti!
- Perché il metodo possa funzionare si devono avere coppie di valori opposti ad ogni chiamata ricorsiva.
- Il campo complesso è l'unico che ammette valori negativi per potenze positive di numeri: $\imath^2 = -1$, $\imath^3 = -\imath$, $\imath^4 = 1, \ldots$.

Esempi di applicazione

Prodotto di polinomi: qualche proprietà dei numeri complessi

Qualche proprietà dei numeri complessi.

- Numero complesso $z \stackrel{\text{def}}{=} a + \imath b$ con $a, b \in \mathrm{R}$ e $\imath$ *unità immaginaria* $\Rightarrow \imath^2 = -1$.
- z in coordinate polari $\stackrel{\text{def}}{=} (r, \theta)$, $r = \sqrt{a^2 + b^2}$ e $\theta \in [0, 2\pi) : \cos\theta = a/r, \sin\theta = b/r$.



- Coordinate polari rendono più semplici le moltiplicazioni/potenze: $z'' = zz' \Rightarrow r'' = rr'$ e $\theta'' = \theta + \theta'$.

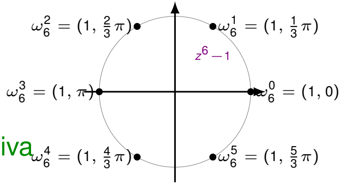
Esempi di applicazione

Prodotto di polinomi: qualche proprietà dei numeri complessi

- Ogni polinomio di grado *n* a coefficienti complessi ha *n* radici complesse.
- $z^n - 1$ ha *n* radici, dette *radici n-esime complesse dell'unità*.
- $(r, \theta) =$ una radice di $z^n - 1 \Rightarrow$ anche $(r^n, n\theta)$ è radice.
- $(1, 0)$ è radice $\Rightarrow$ le radici devono avere $n\theta = 0$.
- Poiché $j2\pi = 0$ per j intero, le n radici distinte sono: $(1, \frac{j}{n}2\pi)$ con $0 \leq j \leq n - 1$.

Esempi di applicazione

Prodotto di polinomi: qualche proprietà dei numeri complessi



- Le n radici distinte sono
$$(1, \frac{j}{n}2\pi)$$
con $0 \leq j \leq n-1$.
- La radice con $j = 1$ è radice primitiva n-esima dell'unità: $\omega_n \stackrel{\text{def}}{=} (1, \frac{2\pi}{n})$.
- Tutte le altre sono espresse come potenze di ω_n: $\omega_n^0, \omega_n, \ldots, \omega_n^{n-1}$
- Le n radici n-esime complesse dell'unità formano un gruppo rispetto alla moltiplicazione.

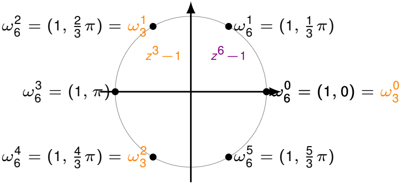
$$\omega_n^0 = \omega_n^n = 1 \quad \text{Elemento neutro}$$
$$\omega_n^{j+n} = \omega_n^j \omega_n^n = \omega_n^j$$
$$\omega_n^{-1} = \omega_n^{n-1}$$

Esempi di applicazione

Prodotto di polinomi: qualche proprietà dei numeri complessi

- **Lemma di cancellazione**
  Per $n \geq 0, k \geq 0$
  e $d > 0$ interi qualsiasi

$$\omega_{dn}^{dk} = \omega_n^k$$



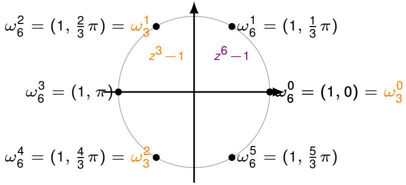
- **Lemma di dimezzamento**
  Se $n > 0$ è pari, allora
  - le n radici sono a coppie di valori opposti: $\omega_n^{\frac{n}{2}+j} = \omega_n^{\frac{n}{2}} \omega_n^j = -1\omega_n^j = -\omega_n^j$.
  - il quadrato delle n radici determina $\frac{n}{2}$ radici $\frac{n}{2}$-esime complesse dell'unità, a loro volta accoppiate (se $\frac{n}{2}$ è pari).
  - quando $n = 2^k$, le radici rimangono accoppiate fino a quando n diventa 1!

Esempi di applicazione

Prodotto di polinomi: qualche proprietà dei numeri complessi

- **Lemma di cancellazione**
  Per $n \geq 0, k \geq 0$
  e $d > 0$ interi qualsiasi

$$\omega_{dn}^{dk} = \omega_n^k$$



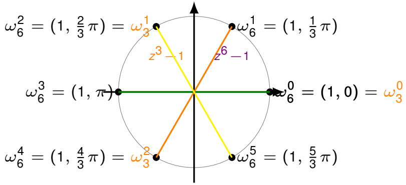
- **Lemma di dimezzamento**
  Se $n > 0$ è pari, allora
  - le n radici sono a coppie di valori opposti:
    $\omega_n^{\frac{n}{2}+j} = \omega_n^{\frac{n}{2}} \omega_n^j = -1\omega_n^j = -\omega_n^j$.
  - il quadrato delle n radici determina $\frac{n}{2}$ radici $\frac{n}{2}$-esime complesse dell'unità, a loro volta accoppiate (se $\frac{n}{2}$ è pari).
  - quando $n = 2^k$, le radici rimangono accoppiate fino a quando n diventa 1!

Esempi di applicazione

Prodotto di polinomi: qualche proprietà dei numeri complessi

- **Lemma di cancellazione**
  Per $n \geq 0, k \geq 0$
  e $d > 0$ interi qualsiasi

$$\omega_{dn}^{dk} = \omega_n^k$$



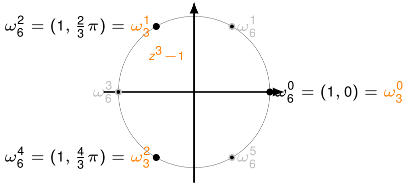
- **Lemma di dimezzamento**
  Se $n > 0$ è pari, allora
  - le n radici sono a coppie di valori opposti:
    $\omega_n^{\frac{n}{2}+j} = \omega_n^{\frac{n}{2}}\omega_n^j = -1\omega_n^j = -\omega_n^j$.
  - il quadrato delle n radici determina $\frac{n}{2}$ radici $\frac{n}{2}$-esime complesse dell'unità, a loro volta accoppiate (se $\frac{n}{2}$ è pari).
  - quando $n = 2^k$, le radici rimangono accoppiate fino a quando n diventa 1!

Esempi di applicazione

Prodotto di polinomi: qualche proprietà dei numeri complessi

- **Lemma di cancellazione**
  Per $n \geq 0, k \geq 0$
  e $d > 0$ interi qualsiasi

$$\omega_{dn}^{dk} = \omega_n^k$$



- **Lemma di dimezzamento**
  Se $n > 0$ è pari, allora
  - le n radici sono a coppie di valori opposti: $\omega_n^{\frac{n}{2}+j} = \omega_n^{\frac{n}{2}}\omega_n^j = -1\omega_n^j = -\omega_n^j$.
  - il quadrato delle n radici determina $\frac{n}{2}$ radici $\frac{n}{2}$-esime complesse dell'unità, a loro volta accoppiate (se $\frac{n}{2}$ è pari).
  - quando $n = 2^k$, le radici rimangono accoppiate fino a quando n diventa 1!

Esempi di applicazione

Prodotto di polinomi: trasformata veloce di Fourier (FTT)

Assemblando il tutto, si ottiene la trasformata veloce di Fourier (FFT).

Procedura $\texttt{FFT}(A)$

// Valuta il polinomio A di grado limite n nelle radici n-esime complesse dell'unità

// A è rappresentato come $(a_0, a_1, \ldots, a_{n-1})$ con $n = 2^k$

if $(n == 1)$ **then return** A; // base della ricorsione

$\omega_n = (1, \frac{2\pi}{n})$;

$\omega = 1$;

$A_e = (a_0, a_2, \ldots, a_{n-2})$ // Fase divide;

$A_o = (a_1, a_3, \ldots, a_{n-1})$;

$(y_{e0}, y_{e1}, \ldots, y_{e\frac{n}{2}-1}) = \texttt{FFT}(A_e)$// Fase impera;

$(y_{o0}, y_{o1}, \ldots, y_{o\frac{n}{2}-1}) = \texttt{FFT}(A_o)$;

...

Esempi di applicazione

Prodotto di polinomi: trasformata veloce di Fourier (FTT)

Procedura `continua FFT(A)`

1: **for** $(j = 0; \quad j \leq \frac{n}{2} - 1; \quad j{+}{+})$ **do** // Fase ricombina
2: $\quad y_j = y_{ej} + \omega y_{oj};$
3: $\quad y_{j+\frac{n}{2}} = y_{ej} - \omega y_{oj};$
4: $\quad \omega = \omega\omega_n;$
5: **endfor**
6: **return** y

Complessità

$T(n) = 2T(\frac{n}{2}) + O(n) = O(n \log n);$

Esempi di applicazione

Prodotto di polinomi: valutazione *divide et impera*

Esempio 2 (FFT($2 - x + 2x^2 - 3x^3$))

Chiamata principale

- $\omega_4 = (1, \frac{\pi}{2}), \omega = 1; A_e = (2, 2); A_o = (-1, -3);$
- $(y_{e0}, y_{e1}) = \texttt{FFT}(A_e)$;// Chiamata ricorsiva 1
- $(y_{o0}, y_{o1}) = \texttt{FFT}(A_o)$;// Chiamata ricorsiva 2

Chiamata ricorsiva 1

- $\omega_2 = (1, \pi), \omega = 1; A_e = 2; A_o = 2;$
- $y_{e0} = \texttt{FFT}(A_e) = 2$;//Risolviamo subito caso base
- $y_{o0} = \texttt{FFT}(A_o) = 2;$
- $y_0 = y_{e0} + 1y_{o0} = 2 + 2 = 4;$
- $y_1 = y_{e0} - 1y_{o0} = 2 - 2 = 0;$
- **return** $(4, 0)$

Esempi di applicazione

Prodotto di polinomi: valutazione *divide et impera*

FFT$(2 - x + 2x^2 - 3x^3)$ (2)

Chiamata ricorsiva 2

- $\omega_2 = (1, \pi)$, $\omega = 1$; $A_e = -1$; $A_o = -3$;
- $y_{e0} = \mathtt{FFT}(A_e) = -1$;//Risolviamo subito caso base
- $y_{o0} = \mathtt{FFT}(A_o) = -3$;
- $y_0 = y_{e0} + 1y_{o0} = -1 - 3 = -4$;
- $y_1 = y_{e0} - 1y_{o0} = -1 + 3 = 2$;
- **return** $(-4, 2)$

Esempi di applicazione

Prodotto di polinomi: valutazione *divide et impera*

FFT($2 - x + 2x^2 - 3x^3$) (3)

Ritorno alla chiamata principale

- $y_e = (4, 0)$; $y_o = (-4, 2)$;
- Ciclo for $j = 0$
- $y_0 = y_{e0} + 1y_{o0} = 4 - 4 = 0$;
- $y_2 = y_{e0} - 1y_{o0} = 4 + 4 = 8$;
- $\omega = (1, \frac{\pi}{2})$
- Ciclo for $j = 1$
- $y_1 = y_{e1} + (1, \frac{\pi}{2})y_{o1} = 0 + (1, \frac{\pi}{2})2 = (2, \frac{\pi}{2})$
- $y_3 = y_{e1} - (1, \frac{\pi}{2})y_{o1} = 0 - (1, \frac{\pi}{2})2 = (2, \frac{3\pi}{2})$
- **return** $(0, (2, \frac{\pi}{2}), 8, (2, \frac{3\pi}{2}))$

Esempi di applicazione

Prodotto di polinomi: interpolazione

Manca la fase di interpolazione per completare l'algoritmo di moltiplicazione di due polinomi.

- La procedura `FFT` realizza la valutazione di A e B in n punti;
- Dagli n valori di A e B, si costruiscono due vettori di $2n$ valori riempiendoli di 0.
- Si determina il vettore moltiplicazione moltiplicando i due vettori punto a punto;
- Il vettore di moltiplicazione è un vettore di valutazione $y(C)$ per C.
- I coefficienti di C si ottengono dai valori per interpolazione;

Esempi di applicazione

Prodotto di polinomi: interpolazione

- Posto $m = 2n$, il vettore $y(C)$ può essere scritto come

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{m-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_m & \omega_m^2 & \cdots & \omega_m^{m-1} \\ 1 & \omega_m^2 & \omega_m^4 & \cdots & \omega_m^{2(m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_m^{m-1} & \omega_m^{2(m-1)} & \cdots & \omega_m^{(m-1)(m-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{m-1} \end{pmatrix}$$

- Ogni elemento della matrice è $V_{k,j} = \omega_m^{kj}$ dove k è indice riga e j indice colonna;
- Se si inverte la matrice, si possono determinare i coefficienti a.
- Si può dimostrare che $V_{k,j}^{-1} = \frac{1}{m}\omega_m^{-kj}$

Esempi di applicazione

Prodotto di polinomi: interpolazione

- Di conseguenza, l'algoritmo FFT(A) può essere riusato come algoritmo di interpolazione. È sufficiente:
  - sostituire ω_m con ω_m^{-1},
  - scambiare a con y
  - dividere per m i risultati ottenuti.
- Quindi l'algoritmo prodotto di polinomi ha complessità $O(n \log n)$.

Esempi di applicazione

Mediana

Definizione 1 (Mediana)

Dato un insieme S di interi, la mediana m è il 50° percentile di S: 50% degli elementi di S sono $\leq m$.

- Se si ordinano gli elementi, la complessità è $O(n \log n)$.
- Ma non ha senso dover ordinare/spostare gli elementi quando si vuole conoscere solo chi è la mediana.

Generalizziamo:

PROBLEMA SELEZIONE

DESCRIZIONE: Un insieme S di interi e un intero k.

QUESITO: Determinare il k° elemento più piccolo di S.

Se $k = 1$, si ricerca il minimo. Se $k = \lfloor |S|/2 \rfloor$, si ricerca la mediana.

Esempi di applicazione

Mediana: algoritmo

L'idea di tipo *divide et impera* per risolvere il problema è la seguente:

- Dato un numero v, si suddivide S in 3 sottoinsiemi:
  1. $S_L = \{\text{elementi inferiori a } v\}$;
  2. $S_v = \{\text{elementi pari a } v\}$;
  3. $S_R = \{\text{elementi superiori a } v\}$;
- la ricerca del k° elemento più piccolo può essere quindi ristretta come:

$$\text{selezione}(S,k) = \begin{cases} \text{selezione}(S_L,k) & k \leq |S_L| \\ v & |S_L| < k \leq |S_L| + |S_v| \\ \text{selezione}(S_R, k - |S_L| - |S_v|) & k > |S_L| + |S_v| \end{cases}$$

- Le tre liste si determinano in tempo $O(n)$.
- Dato S, la dimensione del sottoproblema è $\leq \max\{|S_L|, |S_R|\}$

Esempi di applicazione

Mediana: pseudocodifica

Procedura `selezione`(S, k)

$v =$ elemento di S scelto casualmente;
// Fase *divide*
$S_L = \{$elementi inferiori a $v\}$;
$S_v = \{$elementi pari a $v\}$;
$S_R = \{$elementi superiori a $v\}$;
// Fase *impera*
if $(|S_L| < k \leq |S_L| + |S_v|)$ **then return** v;
if $(k \leq |S_L|)$ **then**
    return `selezione`(S_L, k);
else
    return `selezione`$(S_R, k - |S_L| - |S_v|)$;
endif

Esempi di applicazione

Mediana: scelta di v

Aspetto cruciale: come scegliere v.

- Se v è scelto sempre in modo che $|S_L| \approx |S_R| \approx \frac{1}{2}|S|$
  $\Rightarrow T(n) = T(\frac{n}{2}) + O(n) = O(n)$. Ma è quello che cerchiamo!
- Il caso peggiore è quando v è sempre l'elemento massimo:
  $\Rightarrow T(n) = T(n-1) + O(n) = O(n^2)$.
- Che valore assume il caso medio nell'intervallo
  $[O(n), \ldots, \Theta(n^2)]$?
- Assumere il caso medio è sensato nella pratica?
- Serve un po' di calcolo delle probabilità...

Esempi di applicazione

Mediana: distribuzione di v

Definiamo v buono se risiede tra il 25° e il 75° percentile.

- v è buono $\Longrightarrow |S_L|, |S_R| \leq \frac{3}{4}|S|$.
- 50% di S sono elementi buoni.
- Evento elemento buono è un evento bernoulliano con $p = \frac{1}{2}$.
- Per un evento bernoulliano, numero lanci X prima che si verifichi un successo è distribuito come una v.c. geometrica:

$$Pr(X = k) = (1-p)^{k-1}p$$
$$E(X) = \frac{1}{p}$$

- Quindi, mediamente ogni 2 scelte casuali di v si ha un valore buono $\Longrightarrow$ dimensione dell'istanza diminuisce di almeno $\frac{1}{4}$.

Esempi di applicazione

Mediana: complessità nel caso medio

La complessità media dell'algoritmo `selezione()` può essere calcolata come:

- Sia $s(n) = O(n)$ costo di costruzione dei 3 insiemi S_L, S_v, S_R.
- Sia $w(n)$ costo totale della procedura prima di ottenere un valore buono per v: $E(w(n)) = O(n)$.
- La complessità media può quindi essere espressa come

$$\begin{aligned} E(T(n)) &\le E(T(\frac{3}{4}n) + w(n)) + s(n) \\ &= E(T(\frac{3}{4}n)) + O(n) + O(n) \\ &= O(n) \end{aligned}$$

Questo algoritmo è componente fondamentale per molti altri algoritmi *divide et impera*, tra i quali:

- Quicksort.
- Ricerca elemento massimo nel piano.

Altri problemi risolvibili con *divide et impera*

Elenco incompleto di problemi che ammettono un buon algoritmo di risoluzione del tipo *divide et impera*.

Problema	**Costo naïve**	**Costo *divide et impera***
Ricerca in una lista ordinata	$O(n)$	ricerca binaria: $O(\log n)$
Ordinamento per confronto	$O(n^2)$	Mergersort: $O(n \log n)$ Quicksort: $O(n^2)$ nel caso peggiore, $O(n \log n)$ caso medio.
Ricerca del massimo in un piano	$O(n^2)$	$O(n \log n)$
Ricerca della coppia più vicina in un piano	$O(n^2)$	$O(n \log n)$
Inviluppo convesso	$O(n^2)$	$O(n \log n)$

Esercizi

Equazioni di ricorrenza

Esercizio 2

Risolvere le seguenti equazioni di ricorrenza:

$$T(n) = 2T(n/3) + 1 \tag{1}$$

$$T(n) = 5T(n/4) + n \tag{2}$$

$$T(n) = 7T(n/7) + n \tag{3}$$

$$T(n) = 9T(n/3) + n^2 \tag{4}$$

$$T(n) = 8T(n/2) + n^3 \tag{5}$$

$$T(n) = 49T(n/25) + n^{3/2}\log n \tag{6}$$

$$T(n) = T(n-1) + 2 \tag{7}$$

$$T(n) = T(n-1) + n^c \tag{8}$$

$$T(n) = T(n-1) + c^n \tag{9}$$

$$T(n) = 2T(n-1) + 1 \tag{10}$$

Esercizi

Ricerca in matrice ordinata

Esercizio 3

Risolvere il seguente problema

PROBLEMA RICERCA IN MATRICE ORDINATA

DESCRIZIONE: Una matrice di $n \times n$ di elementi confrontabili. Gli elementi in ogni riga sono ordinati in modo crescente come anche gli elementi in ogni colonna.
Un elemento x.

QUESITO: Determinare in quale posizione x è presente nella matrice.

Esercizi

Minimo in una matrice

Esercizio 4

Risolvere il seguente problema

PROBLEMA MINIMO IN UNA MATRICE

DESCRIZIONE: Una matrice di $n \times n$ di elementi confrontabili.

QUESITO: Determinare in quale posizione è presente il minimo elemento nella matrice.

Esercizi

Eliminazione di duplicati

Esercizio 5

Risolvere il seguente problema

PROBLEMA ELIMINAZIONE DUPLICATI IN UNA ARRAY

DESCRIZIONE: Un array di n elementi confrontabili.

QUESITO: Eliminare eventuali duplicati.