29 marzo 2023

Moltiplicazioni di Matrici

Relazioni di ricorrenza

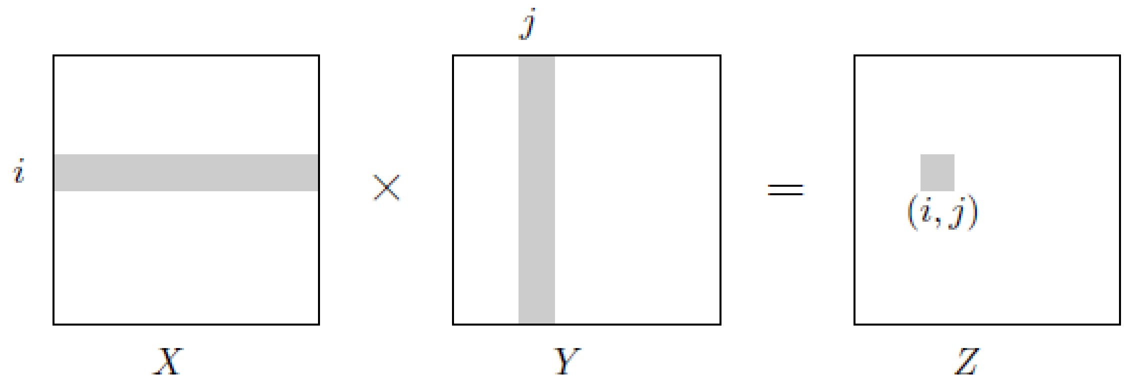
Teoremi generali

Matrix Multiplication
(and decimal wars!)

Ref. [KT] no; [DPV, CLRS] si

Matrix Multiplication

Matrix multiplication. Given two n-by-n matrices X and Y, compute X Y = Z.



Brute force. $\Theta(n^3)$ arithmetic operations.

Fundamental question. Can we improve upon brute force?

Matrix Multiplication: Warmup

Divide-and-conquer.

- Divide: partition A and B into ½n-by-½n blocks.
- Conquer: multiply 8 ½n-by-½n recursively.
- Combine: add appropriate products using 4 matrix additions.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{aligned} C_{11} &= (A_{11} \times B_{11}) + (A_{12} \times B_{21}) \\ C_{12} &= (A_{11} \times B_{12}) + (A_{12} \times B_{22}) \\ C_{21} &= (A_{21} \times B_{11}) + (A_{22} \times B_{21}) \\ C_{22} &= (A_{21} \times B_{12}) + (A_{22} \times B_{22}) \end{aligned}$$

$$\mathrm{T}(n) = \underbrace{8T(n/2)}_{\text{recursive calls}} + \underbrace{\Theta(n^2)}_{\text{add, form submatrices}} \Rightarrow \mathrm{T}(n) = \Theta(n^3)$$

Matrix Multiplication: Key Idea

Key idea. multiply 2-by-2 block matrices with only 7 multiplications.

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$\begin{aligned} P_1 &= A_{11} \times (B_{12} - B_{22}) \\ P_2 &= (A_{11} + A_{12}) \times B_{22} \\ P_3 &= (A_{21} + A_{22}) \times B_{11} \\ P_4 &= A_{22} \times (B_{21} - B_{11}) \\ P_5 &= (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\ P_6 &= (A_{12} - A_{22}) \times (B_{21} + B_{22}) \\ P_7 &= (A_{11} - A_{21}) \times (B_{11} + B_{12}) \end{aligned}$$

$$\begin{aligned} C_{11} &= P_5 + P_4 - P_2 + P_6 \\ C_{12} &= P_1 + P_2 \\ C_{21} &= P_3 + P_4 \\ C_{22} &= P_5 + P_1 - P_3 - P_7 \end{aligned}$$

Fast Matrix Multiplication

Fast matrix multiplication. (Strassen, 1969)

- Divide: partition A and B into ½n-by-½n blocks.
- Compute: 14 ½n-by-½n matrices via 10 matrix additions.
- Conquer: multiply 7 ½n-by-½n matrices recursively.
- Combine: 7 products into 4 terms using 8 matrix additions.

Analysis.

- Assume n is a power of 2.
- T(n) = # arithmetic operations.

$$\mathrm{T}(n) = \underbrace{7T(n/2)}_{\text{recursive calls}} + \underbrace{\Theta(n^2)}_{\text{add, subtract}} \Rightarrow \mathrm{T}(n) = \Theta(n^{\log_2 7}) = O(n^{2.81})$$

The next big improvement took place in the late 1970s,

New approach, (Arnold Schönhage):

It involves translating/reducing matrix multiplication into a different computational problem in linear algebra involving objects called tensors.
The particular tensors used in this problem are three-dimensional arrays of numbers composed of many different parts, each of which looks like a small matrix multiplication problem.
Strassen later called this approach the "laser method."

Matrix multiplication and this problem involving tensors are equivalent to each other in a sense, yet researchers already had faster procedures for solving the latter one.

This is a very common paradigm in theoretical computer science: reducing between problems

Decimal war!

1969	$O(n^{2.\mathbf{81}})$	[Strassen]
1979 (December):	$O(n^{2.5218\mathbf{13}})$	
1980 (January):	$O(n^{2.5218\mathbf{01}})$	[Schönhage (laser method- tensor 3D)]
1987:	$O(n^{2.37\mathbf{6}})$	[Coppersmith-Winograd]
2012:	$O(n^{2.3728\mathbf{73}})$	[V. Vassilevska Williams]
2014:	$O(n^{2.3728\mathbf{639}})$	[François Le Gall]
2020:	$O(n^{2.3728\mathbf{596}})$	[V. Vassilevska Williams, J. Alman]
2022 (October):	$O(n^{2.37\mathbf{188}})$ *???*	*[Duan, Wu and Zhou]* *[announced in a preprint]*

Fast Matrix Multiplication in Theory

Best known: $O(n^{2.3728\mathbf{596}})$ [2020]

Virginia Vassilevska Williams
MIT

Josh Alman
Harvard University

Fast Matrix Multiplication in Theory

Best known. $O(n^{2.3728596})$ [VVW, JA 2020]

Conjecture. $O(n^{2+\varepsilon})$ for any $\varepsilon > 0$.

Caveat. Theoretical improvements to Strassen are progressively less practical.

They are galactic algorithms: the hidden constants by the Big O notation are too large so that they outperform any other algorithm for problems that are sufficiently large, but where "sufficiently large" is so big that the algorithm is never used in practice. Galactic algorithms were so named by Richard Lipton and Ken Regan,[1] because they will never be used on any data sets on Earth.

Eventhough, galactic algorithms may still contribute to computer science:

- may show new techniques
- available computational power may catch up to the crossover point
- can still demonstrate that conjectured bounds can be achieved, or that proposed bounds are wrong, and hence advance the theory of algorithms.

Relazioni di ricorrenza per vari algoritmi Divide-et-Impera

- Dividi il problema di taglia n in a sotto-problemi di taglia n/b
- Ricorsione sui sottoproblemi
- Combinazione delle soluzioni

$T(n)=$ tempo di esecuzione su input di taglia n

$$T(n)= D(n) + a\,T(n/b) + C(n)$$

Alcune relazioni di ricorrenza

Abbiamo considerato una sotto-famiglia

- $T(n) = qT(n/2) + cn$ con $T(2)=c$

  per $q=1$ allora $T(n)= O(n)$

  $q=2$ allora $T(n)= O(n \log_2 n)$

  $q>2$ allora $T(n)= O(n^{\log_2 q})$

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

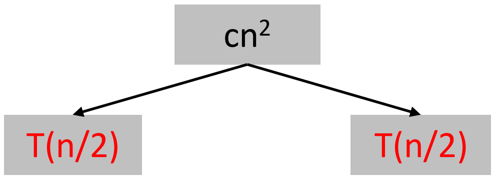
  $T(n) = ?$

Albero di ricorsione

$T(n) = 2\,T(n/2) + cn^2$

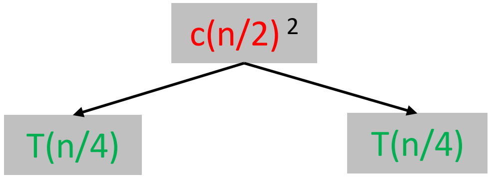
$T(2) = c$

Albero per T(n):

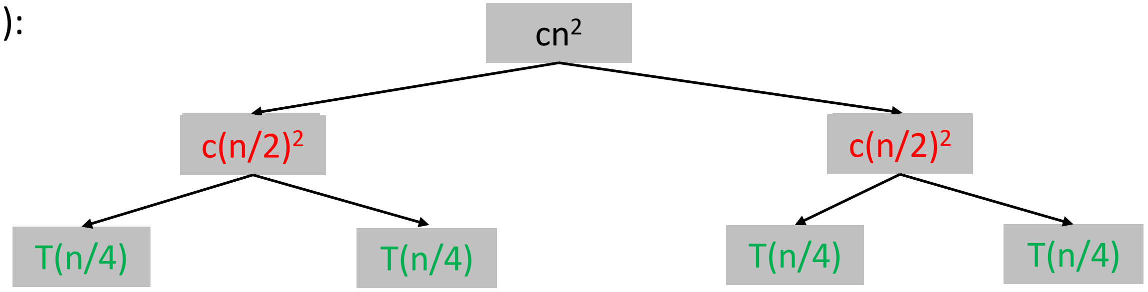


Albero per T(n/2):

$T(n/2) = 2\,T(n/4) + c\,(n/2)^2$



Albero per T(n):



Albero di ricorsione

$T(n) = 2\,T(n/2)+cn^2$

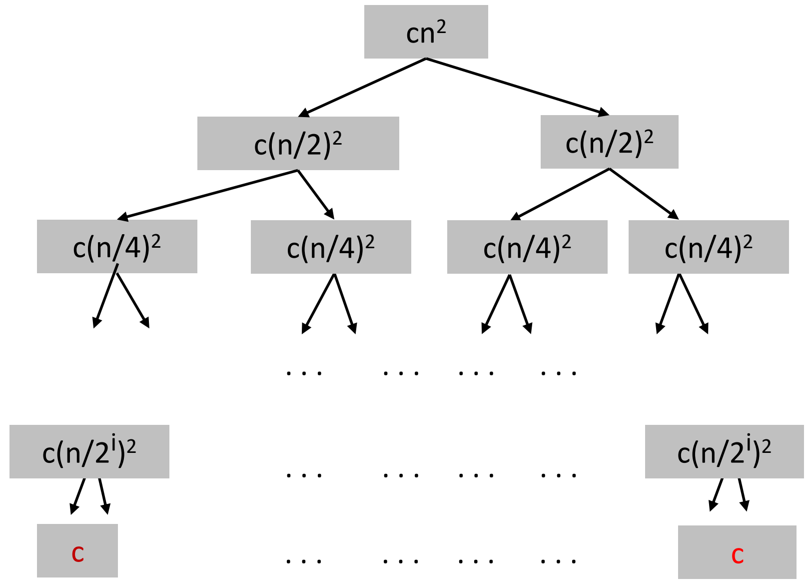
$T(2) = c$

$n/2^i = 2$ sse $n = 2^{i+1}$

sse $i+1 = \log_2 n$

sse $i = \log_2 n - 1$

sse $2^i = n/2$



$i = 0$	cn^2
$i = 1$	$2c(n/2)^2$
$i = 2$	$4c(n/4)^2$
. . .	. . .
$i = \log_2 n-2$	$2^i c(n/2^i)^2 = cn^2/2^i$
$i = \log_2 n-1$	$c\,(2^{\log n-1})$

$$T(n) = c\,(2^{\log n - 1}) + \sum_{i=0}^{\log_2 n-2} cn^2/2^i =$$

$$= c\,n/2 + c\,n^2 \sum_{i=0}^{\log_2 n-2} (1/2)^i$$

(continua)

$$T(n) = c\,(2^{\log n - 1}) + \sum_{i=0}^{\log_2 n - 2} cn^2/\,2^i$$

$$= c\,n/2 + c\,n^2 \sum_{i=0}^{\log_2 n - 2} (1/2)^i$$

$$\leq c\,n/2 + c\,n^2 \sum_{i=0}^{\infty} (1/2)^i$$

$$= c\,n/2 + 2\,c\,n^2$$

$T(n) = O(n^2)$

Inoltre dalla definizione $T(n) \geq cn^2$. Quindi $T(n) = \Theta(n^2)$.

Altre relazioni di ricorrenza

Abbiamo considerato una sotto-famiglia

- $T(n) = qT(n/2) + cn$ con $T(2)=c$

    per q=1 allora $T(n)= \Theta(n)$

    q=2 allora $T(n)= \Theta(n \log_2 n)$

    q>2 allora $T(n)= O(n^{\log_2 q})$

Più in generale.....

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

$T(n)= \Theta(n^2)$

Un teorema generale

Teorema: Se n é potenza di c, la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

é

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Esempi:

- Se $T(n) = 2T(n/3) + dn$, allora $T(n) = O(n)$
- Se $T(n) = 2T(n/2) + dn$, allora $T(n) = O(n \log n)$
- Se $T(n) = 4T(n/2) + dn$, allora $T(n) = O(n^2)$

Nota: Slide precedente per c=2; a=q (libro [KT]) .

Esempi

Sia $T(1) = 1$. Valutiamo

- $T(n) = 2T(n/2) + 6n \qquad T(n) = O(n \log n)$
- $T(n) = 3T(n/3) + 6n - 9 \qquad T(n) = O(n \log n)$
- $T(n) = 2T(n/3) + 5n \qquad T(n) = O(n)$
- $T(n) = 2T(n/3) + 12n + 16 \qquad T(n) = O(n)$
- $T(n) = 4T(n/2) + n \qquad T(n) = O(n^{\log_2 4}) = O(n^2)$
- $T(n) = 3T(n/2) + 9n \qquad T(n) = O(n^{\log_2 3}) = O(n^{1.584\ldots})$

Un teorema generale

Teorema: Se n é potenza di c, la soluzione alla ricorrenza

$$T(n) = \begin{cases} d & \text{se } n \leq 1 \\ aT(n/c) + bn & \text{altrimenti} \end{cases}$$

é

$$T(n) = \begin{cases} O(n) & \text{se } a < c \\ O(n \log n) & \text{se } a = c \\ O(n^{\log_c a}) & \text{se } a > c \end{cases}$$

Esempi:

- Se $T(n) = 2T(n/3) + dn$, allora $T(n) = O(n)$
- Se $T(n) = 2T(n/2) + dn$, allora $T(n) = O(n \log n)$
- Se $T(n) = 4T(n/2) + dn$, allora $T(n) = O(n^2)$

Più in generale.....

Master Theorem

facoltativo

Per forme ancora piú generali del Teorema, che permettono la risoluzione di equazioni di ricorrenza del tipo generale

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

sussiste il seguente risultato

1. Se $f(n) = O(n^{\log_b a - \epsilon})$, per qualche $\epsilon > 0$, allora $T(n) = \Theta(n^{\log_b a})$

2. Se $f(n) = \Theta(n^{\log_b a})$, allora $T(n) = \Theta(n^{\log_b a} \log n)$

3. Se $f(n) = \Omega(n^{\log_b a + \epsilon})$, per qualche $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ per qualche costante $c < 1$ e n sufficientemente grande, allora $T(n) = \Theta(f(n))$

Nota: Vale anche con parte intera inferiore o superiore di n/b
La soluzione T(n) è in notazione Θ

Applicazioni del Master Theorem

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

1.Se $f(n) = O(n^{\log_b a-\epsilon})$, per qualche $\epsilon > 0$, allora $T(n) = \Theta(n^{\log_b a}$

2. Se $f(n) = \Theta(n^{\log_b a})$, allora $T(n) = \Theta(n^{\log_b a} \log n)$

3. Se $f(n) = \Omega(n^{\log_b a+\epsilon})$, per qualche $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ per qualche costante $c < 1$ e n sufficientemente grande, allora $T(n) = \Theta(f(n))$

Confrontare f(n) con n^{logba}: qual è «più grande»?

Il più grande (asintoticamente e *polinomialmente*) vince!

Esempio 1: T(n)= 2 T(n/2)+ log n: a = b = 2, f(n)= log n vs $n^{logba} = n^{log22} = n^1$
$f(n)=O(n^{1-\varepsilon})$ per $\varepsilon=1/2$, quindi T(n)= Θ(n)

Esempio 2: T(n)= 2 T(n/2)+ n: a=b=2, f(n)= n vs $n^{logba}= n^{log22}= n^1$
f(n)= Θ(n), quindi T(n)= Θ(n log n)

Esempio 3: T(n)= 2 T(n/2)+ n^3: a=b=2, f(n)= n^3 vs $n^{logba}= n^{log22}= n^1$
$f(n)= \Omega(n^{1+\varepsilon})$ e inoltre $2(n/2)^3 \leq cn^3$ per c= ¼<1
quindi $T(n)= \Theta(n^3)$

Caso di non applicabilità

$T(n) = 2T(n/2) + n \log n$

$a = b = 2$,

$f(n) = n \log n$ vs $n^{\log_b a} = n^{\log_2 2} = n^1$

$f(n) = n \log n = \Omega(n^1)$, ma non esiste nessun ε per cui $f(n)=\Omega(n^{1+\varepsilon})$

Il Master Theorem non si applica a questa relazione di ricorrenza; bisogna applicare gli altri metodi

Altri esempi

Sia $T(1) = 1$. Valutate

- $\boxed{T(n) = 2T(n/2) + n^3}$
- $T(n) = T(9n/10) + n$
- $T(n) = 16T(n/4) + n^2$
- $T(n) = 7T(n/3) + n^2$
- $T(n) = 7T(n/2) + n^2$
- $T(n) = 2T(n/3) + \sqrt{n}$
- $\boxed{T(n) = T(n-1) + n}$
- $\boxed{T(n) = T(\sqrt{n}) + 1}$

Occorrenze consecutive di 2 (D&I) (dalla piattaforma)

Si scriva lo pseudo-codice di un algoritmo ricorsivo basato sulla tecnica Divide et Impera che prende in input un array di interi positivi e restituisce il massimo numero di occorrenze **consecutive** del numero ‘2’.

Ad esempio, se l'array contiene la sequenza <2 2 3 6 2 2 2 2 3 3> allora l'algoritmo restituisce 4.
Occorre specificare l'input e l'output dell’algoritmo.