Fontes principais

1. Cormem T. H.; Leiserson C. E.; Rivest R.: Stein C.. *Introduction to Algorithms*, 3^a edição, MIT Press, 2009

2. Análise de algoritmo - IME/USP (prof. Paulo Feofiloff) `http://www.ime.usp.br/~pf/analise_de_algoritmos`

Divisão e conquista

Divisão e conquista

1. **Dividir** o problema (instância) em uma ou mais subproblemas.

2. **Conquistar** cada subproblema recursivamente

3. **Combinar** soluções

Busca binária

Busca binária

Encontrar x em um vetor ordenado

1. **Dividir:** comparar x com o elemento do meio
2. **Conquistar:** recursão sobre um subvetor
3. **Combinar:** simples

Busca binária

$T(n) = 1T(n/2) + \Theta(1)$

$n^{\log_b a} = n^{\log_2 1} = 1$

$f(n) = 1$ está "por cima"ou "por baixo"de $n^{\log_b a} = 1$?

$f(n) = 1 = \Theta(1)$ (Caso 2)

$\therefore T(n) = \Theta(1 \cdot \log n)$

Potência de um número

Potência de um número

Dado um número x, um inteiro $n \geq 0$, computar x^n

Algoritmo ingênuo:

$$\underbrace{x \cdot x \cdot x \cdots x}_{n} = x^n$$

Tempo: $\Theta(n)$

Potência de um número

$$x^n = \begin{cases} x^{\frac{n}{2}} \cdot x^{\frac{n}{2}} & \text{, se } n \text{ é par} \\ \\ x^{\frac{n-1}{2}} \cdot x^{\frac{n-1}{2}} \cdot x & \text{, se } n \text{ é ímpar} \end{cases}$$

$$\begin{aligned} T(n) &= T(n/2) + \Theta(1) \\ &= \Theta(\log n) \end{aligned}$$

Número de Fibonacci

Número de Fibonacci

$$F_n = \begin{cases} 0 & \text{, se } n = 0 \\ 1 & \text{, se } n = 1 \\ F_{n-1} + F_{n-2} & \text{, se } n \geq 2 \end{cases}$$

Algoritmo recursivo ingênuo

Tempo: $\Omega(\Phi^n)$ onde $\Phi = \frac{1+\sqrt{5}}{2}$

- $\triangleright$ algoritmo exponencial: ruim
- $\triangleright$ algoritmo polinomial: bom

Obs.: Φ é a razão áurea

Número de Fibonacci

Algoritmo bottom-up:

computar: $F_0, F_1, F_2, \cdots, F_n$

Tempo: $\Theta(n)$

Número de Fibonacci

Algoritmos recursivo ingênuo (versão 1)

$F_n = \frac{\Phi^n}{\sqrt{5}}$ arredondado para o inteiro mais próximo.

Tempo: $\Theta(\lg n)$

Obs.: Este método não é confiável, por causa da aritmética de ponto flutuante que é propensa a erros de arredondamento.

Número de Fibonacci

Teorema:

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

$\Rightarrow$ tempo: $\Theta(\lg n)$

Número de Fibonacci

Teorema:

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

Prova: Por indução sobre n

Passo base:

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^1 = \begin{bmatrix} F_2 & F_1 \\ F_1 & F_0 \end{bmatrix}$$

Número de Fibonacci

Passo indutivo:

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix}^{n-1} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Número de Fibonacci

Passo indutivo:

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix}^{n-1} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Número de Fibonacci

Passo indutivo:

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix}^{n-1} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n}$$

$\square$

Contagem de inversões

Contagem de inversões

Problema: Dada uma permutação $p[1..n]$, determinar o número de inversões em p.

Uma inversão é um par (i,j) de índices de p tal que $i < j$ e $p[i] > p[j]$.

Contagem de inversões (Exemplo)

Entrada:

	1	2	3	4	5	6	7	8	9
p	2	4	1	9	5	3	8	6	7

Saída:11

Inversões: $(1,3),(2,3),(4,5),(2,6),(4,6),(5,6),$
$(4,7),(4,8),(7,8),(4,9)$ e $(7,9)$.

Contagem de inversões - solução ingênua

conta-inversões(p, n)

c = 0
para $i = 0$ **até** $n - 1$ **faça**
    para $j = 0$ **até** $n - 1$ **faça**
        se $p[i] = pj$ **então**
            $c = c + 1$
retorne c

Tempo: $O(n^2)$

Contagem de inversões

Solução melhor: divisão e conquista

Idéia: vamos ordenar e contar ao mesmo tempo!

Contagem de inversões - adaptação do merge-sort

Conta o número de inversões de $A[p..r]$, com $p \leq r$, e rearranja $A[p..r]$ em ordem crescente.

conta-e-ordena(A, p, r)

se $p \geq r$
    então retorne 0
    senão $q = \lfloor (p + r)/2 \rfloor$
        $c1 =$ conta-e-ordena(A, p, q)
        $c2 =$ conta-e-ordena$(A, q + 1, r)$
        $c3 =$ conta-e-intercala(A, p, q, r)
retorne $c1 + c2 + c3$

conta-e-intercala(A, p, q, r)

para $i = p$ **até** q **faça**
    $B[i] = A[i]$
para $j = q + 1$ **até** r **faça**
    $B[r + q + 1 - j] = A[j]$
$i = p$
$j = r$
$c = 0$
para $k = p$ **até** r **faça**
    se $B[i] \leq B[j]$ **então**
        $A[k] = B[i]$
        $i = i + 1$
    senão $A[k] = B[j]$
        $j = j - 1$
        $c = c + (q - i + 1)$
retorne c

Simulação

	p				q				r
A	22	33	55	77	99	11	44	66	88

B									

$c = 0$

Simulação

	k								
A									

	i								j
B	22	33	55	77	99	88	66	44	11

$c = 0$

Simulação

		k							
A	11								

	i							j	
B	22	33	55	77	99	88	66	44	11

$c = 0 + 5 = 5$

Simulação

			k						
A	11	22							

		i						j	
B	22	33	55	77	99	88	66	44	11

$c = 5$

Simulação

				k					
A	11	22	33						

			i					j	
B	22	33	55	77	99	88	66	44	11

$c = 5$

Simulação

					k				
A	11	22	33	44					

			i				j		
B	22	33	55	77	99	88	66	44	11

$c = 5 + 3 = 8$

Simulação

					k				
A	11	22	33	44	55				

				i			j		
B	22	33	55	77	99	88	66	44	11

$c = 8$

Simulação

							k		
A	11	22	33	44	55	66			

				i		j			
B	22	33	55	77	99	88	66	44	11

$c = 8 + 2 = 10$

Simulação

								k	
A	11	22	33	44	55	66	77		

					i	j			
B	22	33	55	77	99	88	66	44	11

$c = 10$

Simulação

									k
A	11	22	33	44	55	66	77	88	

					i = j				
B	22	33	55	77	99	88	66	44	11

$c = 10 + 1 = 11$

Simulação

A	11	22	33	44	55	66	77	88	99

					j	i			
B	22	33	55	77	99	88	66	44	11

$c = 11$

conta-e-intercala(A, p, q, r)

para $i = p$ **até** q **faça**
    $B[i] = A[i]$
para $j = q + 1$ **até** r **faça**
    $B[r + q + 1 - j] = A[j]$
$i = p$
$j = r$
$c = 0$
para $k = p$ **até** r **faça**
    se $B[i] \leq B[j]$ **então**
        $A[k] = B[i]$
        $i = i + 1$
    senão $A[k] = B[j]$
        $j = j - 1$
        $c = c + (q - i + 1)$
retorne c

Contagem de inversões

Análise do algoritmo conta-e-intercala:

- $\triangleright$ linhas 1-4 : $O(n)$
- $\triangleright$ linhas 5-7 : $O(1)$
- $\triangleright$ linhas 8-14 : $O(n)$
- $\triangleright$ linha 15 : $O(1)$

total: $T(n) = 11n + 4 = O(n)$

Contagem de inversões

Análise do algoritmo conta-e-ordena:

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

definida para n potência de 2.

Prova utilizando a técnica da **árvore de recorrência**.

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de } 2 \end{cases}$$

Contagem de inversões

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de 2} \end{cases}$$

Desenrolando a recorrência:

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

Contagem de inversões

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de 2} \end{cases}$$

Desenrolando a recorrência:

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ T(n) &= 2T(2T(n/2^2) + n/2) + n = 2^2T(n/2^2) + 2n \end{aligned}$$

Contagem de inversões

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de 2} \end{cases}$$

Desenrolando a recorrência:

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ T(n) &= 2T(2T(n/2^2) + n/2) + n = 2^2T(n/2^2) + 2n \\ T(n) &= 2^2T(2T(n/2^3) + n/2^2) + 2n = 2^3T(n/2^3) + 3n \end{aligned}$$

Contagem de inversões

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de 2} \end{cases}$$

Desenrolando a recorrência:

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2T(2T(n/2^2) + n/2) + n = 2^2T(n/2^2) + 2n \\ &= 2^2T(2T(n/2^3) + n/2^2) + 2n = 2^3T(n/2^3) + 3n \\ &= 2^3T(2T(n/2^4) + n/2^3) + 3n = 2^4T(n/2^4) + 4n \end{aligned}$$

Contagem de inversões

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de 2} \end{cases}$$

Desenrolando a recorrência:

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2T(2T(n/2^2) + n/2) + n = 2^2T(n/2^2) + 2n \\ &= 2^2T(2T(n/2^3) + n/2^2) + 2n = 2^3T(n/2^3) + 3n \\ &= 2^3T(2T(n/2^4) + n/2^3) + 3n = 2^4T(n/2^4) + 4n \\ &= \cdots \\ &= 2^kT(n/2^k) + kn \end{aligned}$$

onde $k = \lg n$. Disso concluimos que $T(n) = n + n \lg n$

Contagem de inversões

Análise do algoritmo conta-e-ordena:

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

definida para n potência de 2.

Prova utilizando o **método da substituição**

Palpite: $T(n) = O(n \lg n)$

Contagem de inversões

Afirmação: A recorrência

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2T(n/2) + n & \text{se } n \geq 2, n \text{ potência de 2} \end{cases}$$

tem como solução $T(n) = n + n \lg n$

Prova: Por indução em n

Base: $n = 1$

$$T(1) = 1 + 1 \cdot \lg 1 = 1 + 1 \cdot 0 = 1$$

Contagem de inversões

Hipótese de indução: $T(n) = n + n \lg n$ para $m < n$

Passo de indução: $n \geq 2$

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

Contagem de inversões

Hipótese de indução: $T(n) = n + n \lg n$ para $m < n$

Passo de indução: $n \geq 2$

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2(n/2 + (n/2)\lg(n/2)) + n \end{aligned}$$

Contagem de inversões

Hipótese de indução: $T(n) = n + n \lg n$ para $m < n$

Passo de indução: $n \geq 2$

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2(n/2 + (n/2)\lg(n/2)) + n \\ &= 2n + n \lg(n/2) \end{aligned}$$

Contagem de inversões

Hipótese de indução: $T(n) = n + n \lg n$ para $m < n$

Passo de indução: $n \geq 2$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2(n/2 + (n/2)\lg(n/2)) + n \\
&= 2n + n\lg(n/2) \\
&= 2n + n(\lg n - 1)
\end{aligned}
$$

Contagem de inversões

Hipótese de indução: $T(n) = n + n \lg n$ para $m < n$

Passo de indução: $n \geq 2$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2(n/2 + (n/2)\lg(n/2)) + n \\
&= 2n + n\lg(n/2) \\
&= 2n + n(\lg n - 1) \\
&= n + n\lg n
\end{aligned}
$$

$\square$

Contagem de inversões

Hipótese de indução: $T(n) = n + n \lg n$ para $m < n$

Passo de indução: $n \geq 2$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2(n/2 + (n/2) \lg(n/2)) + n \\
&= 2n + n \lg(n/2) \\
&= 2n + n(\lg n - 1) \\
&= n + n \lg n
\end{aligned}
$$

Máximo e o mínimo de um vetor

Máximo e o mínimo de um vetor

Entrada: um vetor $A[]$, um par (l, r) representando $A[l \cdots r]$
Saída: um par (A_i, A_j) contendo o maior e o menor valor de $A[l \cdots r]$

Máximo e o mínimo de um vetor

maxMin(A, l, r)

se $l = r$
    então retorne $(A[l], A[r])$
    senão se $l = r$
        então se $A_l \geq A_r$
            então retorne $(A[l], A[r])$
            senão retorne $(A[r], A[l])$
        senão $k = \lfloor (l + r)/\rfloor 2$
            $(MaxL, MinL) =$ maxMin(A, l, k)
            $(MaxR, MinR) =$ maxMin$(A, l + 1, k)$
            $max = maximo(MaxL, MaxR)$
            $min = maximo(MinL, MinR)$
       retorne (max, min)

Tempo gasto pelo maxMin

Seja $T(n)$ o número de comparações feitas pelo maxMin para um vetor com n elementos. Então:

$T(1) = 0$, $T(2) = 1$
$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 2$

Vamos mostrar que

$$T(n) \leq \frac{5n}{3} - 2$$

Tempo gasto pelo maxMin

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 2$$

$$\leq \frac{5\lceil n/2 \rceil}{3} - 2 + \frac{5\lceil n/2 \rceil}{3} - 2 + 2$$

$$\leq \frac{5}{3}(\lceil n/2 \rceil + \lfloor n/2 \rfloor) - 2$$

$$\leq \frac{5n}{3} - 2$$

seleção do k-ésimo mínimo

seleção do k-ésimo mínimo

Dado um vetor $L[1 \cdots n]$ com n elementos, escreva um algoritmo para determinar o k-ésimo menor elemento, para $1 \leq k \leq n$.

seleção do k-ésimo mínimo

Dado um vetor $L[1 \cdots n]$ com n elementos, escreva um algoritmo para determinar o k-ésimo menor elemento, para $1 \leq k \leq n$.

Solução: ordenar o vetor.

seleção do k-ésimo mínimo

Dado um vetor $L[1 \cdots n]$ com n elementos, escreva um algoritmo para determinar o k-ésimo menor elemento, para $1 \leq k \leq n$.

Solução: ordenar o vetor.

Tempo: $O(n \log n)$

seleção do k-ésimo mínimo

Dado um vetor $L[1 \cdots n]$ com n elementos, escreva um algoritmo para determinar o k-ésimo menor elemento, para $1 \leq k \leq n$.

Solução: ordenar o vetor.

Tempo: $O(n \log n)$

Utilizando divisão e conquista, vamos projetar um algoritmo linear.

seleção do k-ésimo mínimo

Idéia: dividir L em três sublistas L_1, L_2 e L_3 de acordo com um elemento m de L tais que:

▷ L_1 contenha todos os elementos **menores** do que m.
▷ L_2 contém todos os elementos **iguais** a m.
▷ L_3 contenha todos os elementos **maiores** do que m.

Escolha de m

- $\triangleright$ Divida L em $|L|/5$ listas de 5 elementos cada;
- $\triangleright$ Ordene separadamente cada uma dessas listas;
- $\triangleright$ Seja M a lista das medianas das listas de 5 elementos;
- $\triangleright$ m será a mediana de M;

seleção do k-ésimo mínimo

Algoritmo: seleção(k, L)

Entrada: Um inteiro k e um vetor $L[1..n]$.

Saída: O k-ésimo menor elemento de L.

seleção do k-ésimo mínimo

1. Se $n < 15$ então "ordene L e devolva L_k;
2. Divida L em listas de 5 elementos cada;
3. Ordene separadamente cada uma dessas listas;
4. Seja M a lista das medianas das listas de 5 elementos;
5. $m = \text{seleção}(\lceil |M|/2 \rceil, M)$;
6. Sejam L_1, L_2 e L_3 as sublistas dos elementos de L que são menores, iguais e maiores do que m, respectivamente;
7. Se $|L_1| \geq k$ então devolva $\text{seleção}(k, L_1)$
8. senão Se $(|L_1| + |L_2|) \geq k$
   - então devolva m
   - senão devolva $\text{seleção}(k - |L_1| - |L2|, L_3)$

seleção do k-ésimo mínimo

Tempo de execução:

- $\triangleright$ linhas 1 a 4: $O(n)$
- $\triangleright$ linha 5: $T(n/5)$
- $\triangleright$ linha 6: $O(n)$
- $\triangleright$ linhas 7 e 8: $T(3n/4)$

seleção do k-ésimo mínimo

Podemos expressar o tempo $T(n)$ pela seguinte recorrência:

$$T(n) \leq \begin{cases} cn & \text{para } n < 15, \\ T(n/5) + T(3n/4) + cn & \text{para } n \geq 15 \end{cases}$$

seleção do k-ésimo mínimo

Vamos mostrar, por indução em n que

$$T(n) \leq 20cn$$

seleção do k-ésimo mínimo

Vamos mostrar, por indução em n que

$$T(n) \leq 20cn$$

Prova:

Base: A afirmação é verdadeira para $n < 15$

Hipótese: Vamos supor, por hipótese de indução, que ela é válida para valores menores do que n, e vamos provar a sua validade para n.

Passo de indução:

$$T(n) \leq T(n/5) + T(3n/4) + cn$$

Passo de indução:

$$\begin{aligned} T(n) &\leq T(n/5) + T(3n/4) + cn \\ &\leq 20c(n/5) + 20c(3n/4) + cn \end{aligned}$$

Passo de indução:

$$\begin{aligned} T(n) &\leq T(n/5) + T(3n/4) + cn \\ &\leq 20c(n/5) + 20c(3n/4) + cn \\ &= 20c(n/5 + 3n/4) + cn \end{aligned}$$

Passo de indução:

$$\begin{aligned}
T(n) &\leq T(n/5) + T(3n/4) + cn \\
&\leq 20c(n/5) + 20c(3n/4) + cn \\
&= 20c(n/5 + 3n/4) + cn \\
&= 20c(19n/20) + cn
\end{aligned}$$

Passo de indução:

$$
\begin{aligned}
T(n) &\le T(n/5) + T(3n/4) + cn \\
&\le 20c(n/5) + 20c(3n/4) + cn \\
&= 20c(n/5 + 3n/4) + cn \\
&= 20c(19n/20) + cn \\
&= 20cn
\end{aligned}
$$

Multiplicação de matriz

Multiplicação de matriz

Entrada: $A = [a_{ij}]$, $B = [b_{ij}]$

$C = [c_{ij}] = A \cdot B$

$$C_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Multiplicação de matriz

Algoritmo padrão : $\Theta(n^3)$

para $i = 1$ **até** n **faça**
  para $j = 1$ **até** n **faça**
    $c_{ij} = 0$
    para $k = 1$ **até** n **faça**
      $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$

Multiplicação de matriz

Algoritmo divisão e conquista

Idéia: matriz $n \times n$

▷ Dividir a matriz em 4 submatrizes de dimensões $n/2 \times n/2$

$$\underbrace{\left[\begin{array}{c|c} \mathsf{r} & \mathsf{s} \\ \hline \mathsf{t} & \mathsf{u} \end{array}\right]}_{C} = \underbrace{\left[\begin{array}{c|c} \mathsf{a} & \mathsf{b} \\ \hline \mathsf{c} & \mathsf{d} \end{array}\right]}_{A} \underbrace{\left[\begin{array}{c|c} \mathsf{e} & \mathsf{f} \\ \hline \mathsf{g} & \mathsf{h} \end{array}\right]}_{B}$$

Multiplicação de matriz

$$\underbrace{\left[\begin{array}{c|c} \mathsf{r} & \mathsf{s} \\ \hline \mathsf{t} & \mathsf{u} \end{array}\right]}_{C} = \underbrace{\left[\begin{array}{c|c} \mathsf{a} & \mathsf{b} \\ \hline \mathsf{c} & \mathsf{d} \end{array}\right]}_{A} \underbrace{\left[\begin{array}{c|c} \mathsf{e} & \mathsf{f} \\ \hline \mathsf{g} & \mathsf{h} \end{array}\right]}_{B}$$

$$\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dg \\ u &= cf + dh \end{aligned}$$

Obtemos 8 multiplicações recursivas de matrizes $n/2 \times n/2$ e 4 adições de matrizes ($\Theta(n^2)$)

Multiplicação de matriz

$$\underbrace{\left[\begin{array}{c|c} \mathsf{r} & \mathsf{s} \\ \hline \mathsf{t} & \mathsf{u} \end{array}\right]}_{C} = \underbrace{\left[\begin{array}{c|c} \mathsf{a} & \mathsf{b} \\ \hline \mathsf{c} & \mathsf{d} \end{array}\right]}_{A} \underbrace{\left[\begin{array}{c|c} \mathsf{e} & \mathsf{f} \\ \hline \mathsf{g} & \mathsf{h} \end{array}\right]}_{B}$$

$$r = ae + bg$$
$$s = af + bh$$
$$t = ce + dg$$
$$u = cf + dh$$

$$T(n) = 8T(n/2) + \Theta(n^2)$$

Multiplicação de matriz

$$\underbrace{\left[\begin{array}{c|c} \mathsf{r} & \mathsf{s} \\ \hline \mathsf{t} & \mathsf{u} \end{array}\right]}_{C} = \underbrace{\left[\begin{array}{c|c} \mathsf{a} & \mathsf{b} \\ \hline \mathsf{c} & \mathsf{d} \end{array}\right]}_{A} \underbrace{\left[\begin{array}{c|c} \mathsf{e} & \mathsf{f} \\ \hline \mathsf{g} & \mathsf{h} \end{array}\right]}_{B}$$

$$r = ae + bg$$
$$s = af + bh$$
$$t = ce + dg$$
$$u = cf + dh$$

$$T(n) = 8T(n/2) + \Theta(n^2)$$

$= \Theta(n^3)$ (caso 1)

Multiplicação de matriz

Algoritmo de Strassen

Idéia: reduzir o número de multiplicações

$$
\begin{aligned}
P_1 &= a \cdot (f - h) \\
P_2 &= (a + b) \cdot h \\
P_3 &= (c + d) \cdot e \\
P_4 &= d \cdot (g - e) \\
P_5 &= (a + d) \cdot (e + h) \\
P_6 &= (b - d) \cdot (g + h) \\
P_7 &= (a - c) \cdot (e + f)
\end{aligned}
$$

$$
\begin{aligned}
r &= P_5 + P_4 - P_2 + P6 \\
s &= P_1 + P_2 \\
t &= P_3 + P_4 \\
u &= P_5 + P_1 - P_3 - P_7
\end{aligned}
$$

Multiplicação de matriz

Algoritmo de Strassen

1. **Dividir:** Computar os termos por produto

2. **Conquistar:** Recursivamente computar $P_1, P_2, \cdots, P_7$

3. **Combinar:** Computar r, s, t, u ($\Theta(n^2)$)

Tempo: $T(n) = 7T(n/2) + \Theta(n^2)$

$= \Theta(n^{\lg_2 7}) = O(n^{2.81})$

O melhor algoritmo encontrado até o momento: $O(n^{2.376})$

Obrigado