Aula 07 - QuickSort

Algoritmo de Ordenação

Prof. Marco Aurélio Stefanes
marco em dct.ufms.br
www.dct.ufms.br/ ~ marco

Algoritmo Quicksort

- Paradigma divisão e conquista
- Tempo de execução no pior caso é $\Theta(n^2)$
- Tempo esperado é $\Theta(n \lg n)$
- Constante da notação é pequena
- Ordenação in-place
- Usa técnica de algoritmos probabilísticos
- Na prática é melhor que os anteriores em geral
- Comportamento complexo

Algoritmo Quicksort

Princípio

Para ordenar um vetor $A[p \ldots r]$:

- **Divisão** o vetor é particionado em dois subvetores não-vazios $A[p \ldots q]$ e $A[q+1 \ldots r]$ t.q. cada elemento de $A[p \ldots q]$ é menor ou igual a cada elemento de $A[q+1 \ldots r]$.
- **Conquista** os dois subvetores são ordenados por chamadas recursivas ao quicksort
- **Combina** Nada a ser feito

Chave Particionamento em tempo linear

Algoritmo Quicksort

Código do Quicksort

Quicksort(A, p, r)
if $p < r$ **then**
  $q =$ particiona(A, p, r)
  Quicksort(A, p, q)
  Quicksort$(A, q+1, r)$

Chamada Inicial: Quicksort$(A, 1, n)$

Particiona

Problema: Rearranjar um dado vetor $A[p \ldots r]$ e devolver um índice q, $p \leq q \leq r$, tais que

$$A[p \ldots q-1] \leq A[q] < A[q+1 \ldots r]$$

Entra:

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

Sai:

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

Particiona

	i	j								x
A	99	33	55	77	11	22	88	66	33	44

Particiona

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

Particiona

	i	j								x
A	99	33	55	77	11	22	88	66	33	44

Particiona

	i		j								x
A		99	33	55	77	11	22	88	66	33	44
		i		j							x
A		33	99	55	77	11	22	88	66	33	44

Particiona

	i		j								x
A		99	33	55	77	11	22	88	66	33	44
		i		j							x
A		33	99	55	77	11	22	88	66	33	44
		i				j					x
A		33	99	55	77	11	22	88	66	33	44

Particiona

	i		j								x
A		99	33	55	77	11	22	88	66	33	44
		i		j							x
A		33	99	55	77	11	22	88	66	33	44
		i			j						x
A		33	99	55	77	11	22	88	66	33	44

Particiona

	i		j								x
A		99	33	55	77	11	22	88	66	33	44
		i		j							x
A		33	99	55	77	11	22	88	66	33	44
			i				j				x
A		33	11	55	77	99	22	88	66	33	44

Particiona

i		j								x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
		i				j				x
A	33	11	55	77	99	22	88	66	33	44
			i				j			x
A	33	11	22	77	99	55	88	66	33	44

Particiona

i		j								x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
		i				j				x
A	33	11	55	77	99	22	88	66	33	44
			i				j			x
A	33	11	22	77	99	55	88	66	33	44
			i						j	x
A	33	11	22	77	99	55	88	66	33	44

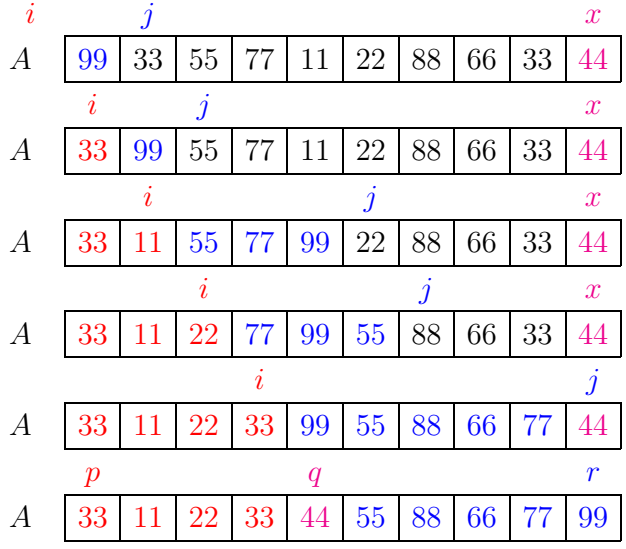
Particiona

i		j								x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
		i				j				x
A	33	11	55	77	99	22	88	66	33	44
			i				j			x
A	33	11	22	77	99	55	88	66	33	44
			i					j		x
A	33	11	22	77	99	55	88	66	33	44

Particiona

i		j								x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
		i				j				x
A	33	11	55	77	99	22	88	66	33	44
			i				j			x
A	33	11	22	77	99	55	88	66	33	44
				i						j
A	33	11	22	33	99	55	88	66	77	44

Particiona



Particiona

Rearranja $A[p \ldots r]$ de modo que $p \leq q \leq r$ e

$A[p \ldots q{-}1] \leq A[q] < A[q{+}1 \ldots r]$

Particiona(A, p, r)

1: $x = A[r]$ ▷ x é o "pivô"
2: $i = p{-}1$
3: **for** $j = p$ até $r - 1$ **do**
4: **if** $A[j] \leq x$ **then**
5: $i = i + 1$
6: $A[i] \leftrightarrow A[j]$
7: $A[i{+}1] \leftrightarrow A[r]$
8: devolva $i + 1$

Invariantes: no começo de cada iteração de 3–6,
(i0) $A[p \ldots i] \leq x$ (i1) $A[i{+}1 \ldots j{-}1] > x$ (i2) $A[r] = x$

Complexidade do Particiona

Quanto tempo consome em função de $n := r - p + 1$?

linha	consumo de todas as execuções da linha
1-2	$= 2\Theta(1)$
3	$= \Theta(n)$
4	$= \Theta(n)$
5-6	$= 2O(n)$
7-8	$= 2\Theta(1)$
total	$= \Theta(2n + 4) + O(2n)$ $= \Theta(n)$

Conclusão:
O algoritmo Particiona consome tempo $\Theta(n)$.

Complexidade do Quicksort

- Suponha que todas as entradas são distintas
- Seja $T(n)$ o tempo de pior caso para uma entrada de tamanho n

Análise de Pior Caso

- Entrada crescente ou decrescente
- Partição com o pivô de valor Mínimo ou Máximo
- Um lado da ártição sempre terá 0 elemento.

Note o tempo do particionamento :
$T(0) = \Theta(1)$
$T(n) = \Theta(n)$

Análise de Pior Caso

Recorrência para $n = r - p + 1$

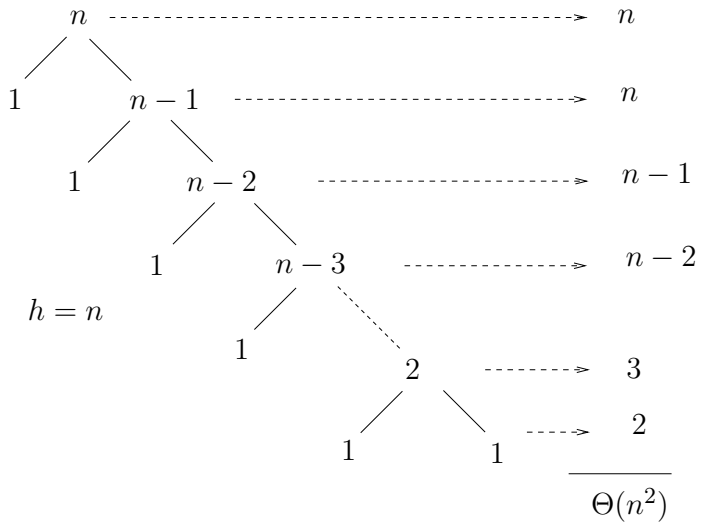
$$T(n) = T(0) + T(n-1) + \Theta(n)$$
$$T(n) = T(n-1) + \Theta(n) \quad \text{Série aritmética}$$
$$T(n) = \Theta(n^2)$$

Conclusão: Não é melhor que o Insertionsort

Em mais detalhes temos

$$T(n) = \max_{0 \le q \le n-1} \{T(q) + T(n-q-1)\} + \Theta(n) \quad \text{para } n = 3, 4, \ldots$$

Árvore de Recursão no Pior Caso



Análise de Melhor Caso

Recorrência para $n = r - p + 1$

Caso a partição divida em dois vetores de tamanho $n/2$, o quicksort roda mais rápido.

$$T(n) = 2T(n/2) + \Theta(n)$$
Mesmo que o MergeSort
$$= \Theta(n \lg n)$$

Apenas para ter intuição sobre o problema

Análise de Caso Médio

O caso médio do quicksort está mais próximo do melhor caso do que do pior caso.

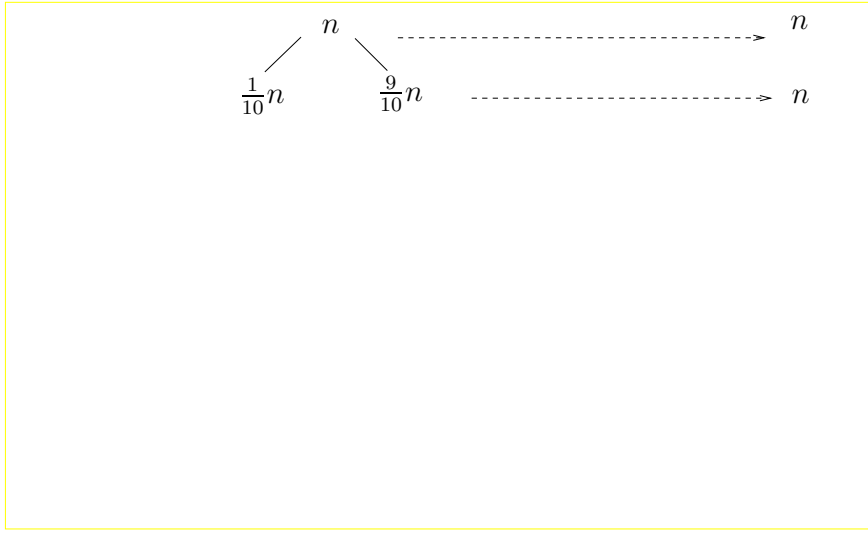
Suponha que a divisão é sempre

$$\frac{n}{10} : \frac{9n}{10}$$

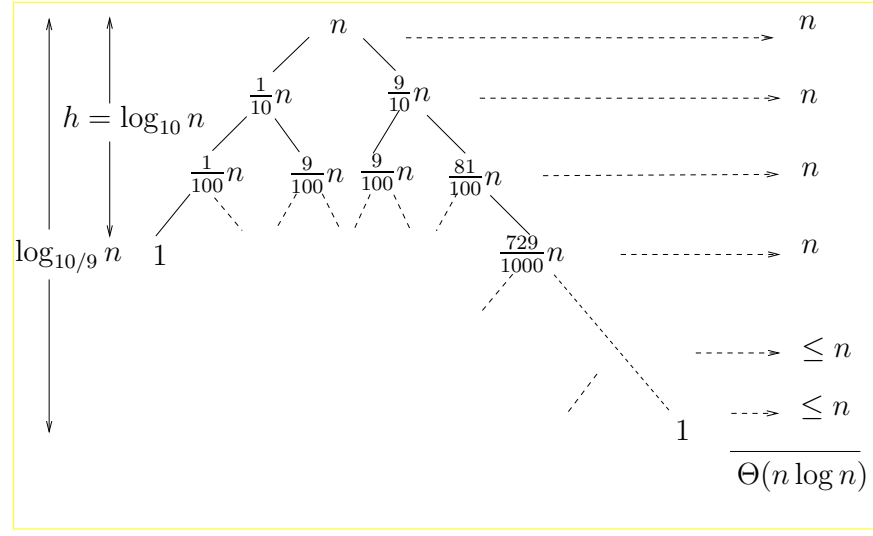
Recorrência:

$$T(n) = T(\frac{n}{10}) + T(\frac{9n}{10}) + \Theta(n)$$

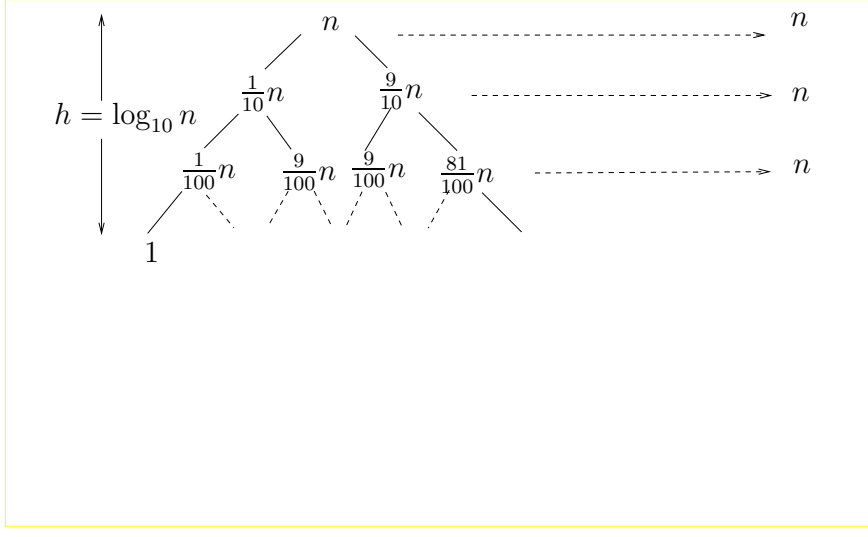
Árvore de Recursão neste caso



Árvore de Recursão neste caso



Árvore de Recursão neste caso



Análise de Caso Médio

Suponha que alternamos melhor caso $M(n)$ com pior caso $P(n)$

Neste caso temos a recorrência:

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

Continuamos com tempo assintótico igual ao do melhor caso

Quicksort Probabilístico

Idéia: Particionar tomando um pivô aleatório

- Tempo de execução independente da ordem da entrada
- Não é necessário fazer suposição sobre a distribuição da entrada
- Nenhuma entrada específica produz o pior caso
- Pior caso somente é produzido com um gerador de números aleatório

Particiona Probabilístico

Há duas formas de induzir uma distribuição onde todas as permutações da entrada são igualmente prováveis.

Particiona_prob (A, p, r)

$i = random(p, r)$
troque $A[r]$ com $A[i]$
devolva Particiona(A, p, r)

Quicksort é alterado para chamar o Particiona_prob ao invés do Particiona

Caso Médio do Quicksort

$T(n)$ Tempo médio esperado para ordenar um vetor com n elementos

Recorrência

$$T(n) = \frac{1}{n}\big(T(1) + T(n-1) + \sum_{q=1}^{n-1}(T(q) + T(n-q))\big) + \Theta(n)$$

Note que

$$\begin{aligned}\frac{1}{n}\big((T(1) + T(n-1))\big) &= \frac{1}{n}(\Theta(1) + O(n^2)) \\ &= O(n)\end{aligned}$$

Caso Médio do Quicksort

Assim

$$T(n) = \frac{1}{n}\sum_{q=1}^{n-1}\big(T(q) + T(n-q)\big) + \Theta(n).$$

Note que cada termo $T(k)$, para $k = 1, 2, \ldots, n-1$ ocorre como $T(q)$ e como $T(n-q)$. Juntando os dois termos:

$$T(n) = \frac{2}{n}\sum_{k=1}^{n-1} T(k) + \Theta(n).$$

Caso Médio do Quicksort

Resolvendo a Recorrência
Método da Substiuição: Suponha
$T(n) \leq an \lg n + b, \quad a, b > 0$ const.

$$\begin{aligned} T(n) &= \frac{2}{n}\sum_{k=1}^{n-1} T(k) + \Theta(n) \\ &\leq \frac{2}{n}\sum_{k=1}^{n-1}(ak \lg k + b) + \Theta(n) \\ &= \frac{2a}{n}\sum_{k=1}^{n-1} k \log k + \frac{2b}{n}(n-1) + \Theta(n) \end{aligned}$$

Caso Médio do Quicksort

Fato: $\sum_{k=1}^{n-1} k \log k \leq \frac{1}{2}n^2 \lg n - \frac{1}{8}n^2$

$$\begin{aligned} T(n) &\leq \frac{2a}{n}\left(\frac{1}{2}n^2 \lg n - \frac{1}{8}n^2\right) + \frac{2b}{n}(n-1) + \Theta(n) \\ &\leq an \lg n - \frac{a}{4}n + 2b + \Theta(n) \\ &= an \lg n + b + \left(\Theta(n) + b - \frac{a}{4}n\right) \\ &\leq an \lg n + b \end{aligned}$$

Escolhendo a para $\frac{a}{4}n$ dominar $\Theta(n) + b$