

## Algoritmos

Pedro Hokama

## Fontes

- [cls] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest, Ronald L. Rivest e Clifford Stein.
- [timr] Algorithms Illuminated Series, Tim Roughgarden

Apresentação Baseada:

- Stanford Algorithms  
<https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V>  
<https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBLSz17A5HEK6>
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende

1 / 29

2 / 29

## QuickSort - revisão

5	6	4	3	2	8	1	7
---	---	---	---	---	---	---	---

1	4	3	2	5	8	6	7
---	---	---	---	---	---	---	---

1	4	3	2	7	6	8
---	---	---	---	---	---	---

2	3	4	6	7
---	---	---	---	---

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

## QuickSort

- Vamos analisar a complexidade do QuickSort.
- O que acontece com o QuickSort se a escolha de pivô for muito ruim?
- Como seria a árvore de recursão e complexidade no pior caso? Por exemplo se você tiver um arranjo já ordenado, e sempre escolhermos o primeiro elemento.

3 / 29

4 / 29

1 2 3 4 5 6 7 8

1 2 3 4 5 6 7 8

2 3 4 5 6 7 8

3 4 5 6 7 8

4 5 6 7 8

5 6 7 8

6 7 8

1 2 3 4 5 6 7 8

## QuickSort

- Nesse caso iremos dividir em duas partes, uma vazia e uma com  $n - 1$  elementos. E o trabalho da partição será pelo menos:

$$n + (n - 1) + (n - 2) + \dots + 1 = \Theta(n^2)$$

5 / 29

6 / 29

Suponha que implementamos o QuickSort, de forma que o pivô é sempre o primeiro elemento do arranjo. Qual é o tempo de execução desse algoritmo em um vetor de entrada que já está ordenado?

- a Não é possível estimar
- b  $\Theta(n)$
- c  $\Theta(n \log n)$
- d  $\Theta(n^2)$

- O tempo de execução do QuickSort depende da qualidade do pivô escolhido.
- Um bom pivô é aquele que divide os problemas em partes de tamanho parecido,
- enquanto um pivô ruim é aquele que deixa duas partes muito desiguais.

7 / 29

8 / 29

## QuickSort - o caso bom

- Se dividirmos o arranjo sempre no máximo, ou seja, na metade.
- Isso aconteceria se encontrássemos a mediana.
- Teremos duas partes com menos que metade dos elementos
- Podemos limitar superiormente pela mesma recorrência do MergeSort
- Sabemos então que no melhor caso QuickSort é  $O(n \log n)$

Seja  $T(n)$  o tempo de execução desse algoritmo em um vetor de tamanho  $n$ . Então

$$T(n) \leq 2T\left(\frac{n}{2}\right) + \Theta(n)$$

Pelo teorema mestre:

$$T(n) = O(n \log n)$$

9 / 29

## Aleatorização do QuickSort

- A aleatorização de algoritmos é uma ferramenta importante da bagagem de qualquer profissional da computação.
- Veremos como aplicar essa técnica no QuickSort e as vantagens que ela pode trazer.
- A ideia é escolher cada pivô aleatoriamente com a **mesma probabilidade**. E assim escolher um pivô "razoavelmente bom", em uma frequência "razoavelmente alta".
- Digamos que um pivô razoavelmente bom seria um que divida o arranjo em 25-75%, que é suficiente para garantir o tempo de execução de  $O(n \log n)$
- Como seria a árvore de recursão nesse caso?
- Note que metade dos elementos, se escolhidos como pivô, resultam numa divisão de 25-75% ou melhor.

11 / 29

Suponha que implementamos o QuickSort, de forma que (magicamente) sempre escolhamos a mediana como pivô. Qual é o tempo de execução desse algoritmo?

- a Não é possível estimar
- b  $\Theta(n)$
- c  $\Theta(n \log n)$
- d  $\Theta(n^2)$

- Infelizmente não é possível encontrar a mediana em  $O(1)$  para garantir esse tempo de execução.
- Felizmente não é necessário!

10 / 29

## Revisão(zinha) de Probabilidade

Para completar a análise do QuickSort Aleatorizado e outros algoritmos, precisamos relembrar alguns conceitos de probabilidade:

- Espaço Amostral
- Eventos
- Variáveis Aleatórias
- Esperança

12 / 29

## Espaço Amostral

- **Espaço Amostral**  $\Omega$  = todos os possíveis resultados de um evento aleatório.
- cada resultado  $i \in \Omega$  tem probabilidade  $p(i) \geq 0$ .
- **Restrição:**  $\sum_{i \in \Omega} p(i) = 1$ , ou seja, com certeza um dos resultados de  $\Omega$  vai acontecer.

### Exemplo 1

Rolar dois dados de 6 lados.

$\Omega = \{(1, 1), (2, 1), (3, 1), \dots, (5, 6), (6, 6)\}$  (pares ordenados)

$p(i) = \frac{1}{36}$  para todos  $i \in \Omega$

### Exemplo 2

Escolher o índice do pivô aleatório da primeira iteração do QuickSort.

$\Omega = \{1, 2, \dots, n\}$

$p(i) = \frac{1}{n}$  para todo  $i \in \Omega$

13 / 29

## Evento

- Um **evento** é um subconjunto  $S \subseteq \Omega$ .
- A probabilidade de um evento  $S$  é

$$\sum_{i \in S} p(i).$$

- Considere o evento "a soma dos dados é 7". Qual é a probabilidade desse evento?

- a 1/36
- b 1/12
- c 1/6
- d 1/2

- $S = \{(1, 6), (2, 5), (3, 4), (4, 3), (5, 2), (6, 1)\}$
- $Pr[S] = 6/36 = 1/6$

14 / 29

- Considere o evento "o pivô aleatório escolhido induz uma divisão de pelo menos 25 – 75% ou melhor". Qual é a probabilidade desse evento?

- a  $1/n$
- b  $1/4$
- c  $1/2$
- d  $3/4$

- $S =$  qualquer escolha que não seja os  $1/4$  menores ou os  $1/4$  maiores.

$$Pr[S] = \frac{n}{2} = \frac{n}{2} \cdot \frac{1}{n} = \frac{1}{2}$$

15 / 29

## Variáveis Aleatórias

- Uma **Variável Aleatória**  $X$  é uma função:

$$X : \Omega \rightarrow \mathbb{R}$$

- Exemplo 1: A soma dos dois dados, ex:  $X((2, 5)) = 7$
- Exemplo 2: O valor do maior dado, ex:  $Y((2, 5)) = 5$
- Exemplo 3: Tamanho do subproblema passado na primeira chamada recursiva do QuickSort

16 / 29

## Esperança

- Seja  $X : \Omega \rightarrow \mathbb{R}$  uma Variável Aleatória.
- A **Esperança**  $E[X]$  de  $X$  é o valor médio de  $X$  ponderado pela probabilidade.

$$E[X] = \sum_{i \in \Omega} X(i) \cdot p(i)$$

- Qual a esperança da soma de dois dados?

- a 6.5
- b 7
- c 7.5
- d 8

- Qual a esperança do tamanho do subproblema passado na primeira chamada recursiva do QuickSort?

- a  $n/4$
- b  $n/3$
- c  $(n-1)/2$
- d  $3n/4$

- Seja  $X$  o tamanho do subproblema.

$$\begin{aligned} E[X] &= \frac{1}{n} \cdot 0 + \frac{1}{n} \cdot 1 + \frac{1}{n} \cdot 2 + \dots + \frac{1}{n} \cdot (n-1) \\ &= \frac{1 + 2 + \dots + (n-1)}{n} \\ &= \frac{(n-1)(1+n-1)/2}{n} \\ &= \frac{n^2 - n/2}{n} = \frac{n^2 - n}{2} \cdot \frac{1}{n} \\ &= \frac{n-1}{2} \end{aligned}$$

$x$	2	3	4	5	6	7	8	9	10	11	12
$S'$						□□	□□	□□			
		□□	□□	□□	□□	□□	□□	□□			
	□□	□□	□□	□□	□□	□□	□□	□□	□□		
$ S' $	1	2	3	4	5	6	5	4	3	2	1

$$\begin{aligned} E[X] &= \frac{1}{36} \cdot 2 + \frac{2}{36} \cdot 3 + \frac{3}{36} \cdot 4 + \frac{4}{36} \cdot 5 + \frac{5}{36} \cdot 6 + \frac{6}{36} \cdot 7 + \frac{5}{36} \cdot 8 + \frac{4}{36} \cdot 9 + \frac{3}{36} \cdot 10 + \frac{2}{36} \cdot 11 + \frac{1}{36} \cdot 12 \\ &= \frac{2 + 6 + 12 + 20 + 30 + 42 + 40 + 36 + 30 + 22 + 12}{36} \\ &= \frac{246}{36} = 7 \end{aligned}$$

17 / 29

18 / 29

## Linearidade da Esperança

### Lema

Sejam  $X_1, \dots, X_n$  variáveis aleatórias definidas em  $\Omega$ . Então:

$$E \left[ \sum_{j=1}^n X_j \right] = \sum_{j=1}^n E[X_j]$$

- Vale mesmo quando as variáveis não são independentes!
- Exemplo:  $X_1$  e  $X_2$  são variáveis aleatórias que dizem o valor do primeiro e do segundo dado.

$$E[X_j] = \frac{1}{6}(1 + 2 + 3 + 4 + 5 + 6) = 3.5$$

- Qual o valor esperado para a soma de dois dados?

$$E[X] = E[X_1 + X_2] = E[X_1] + E[X_2] = 3.5 + 3.5 = 7.$$

19 / 29

20 / 29

**Teorema**

Para qualquer entrada de comprimento  $n$ , o tempo médio de execução do QuickSort (com pivôs aleatórios) é  $O(n \log n)$ .

- Primeiramente considere como entrada um arranjo  $A$  de comprimento  $n$ .
- Espaço amostral:  $\Omega =$  todas as possíveis sequências de escolhas de pivôs no QuickSort.
- Variável aleatória: para qualquer  $\sigma \in \Omega$ ,  $C(\sigma) =$  número de comparações entre dois elementos do arranjo feito pelo algoritmo QuickSort dado as escolhas  $\sigma$ . Note que o tempo de execução total do QuickSort é dominado por esse número.
- O objetivo então é mostrar que  $E[C] = O(n \log n)$

**Algoritmo 1:** QuickSort

**Entrada:** Um arranjo  $A$ , índices  $l$  e  $r$

**Saída:** Um arranjo com os mesmos elementos de  $A$  porém ordenados

- 1 se  $r - l \leq 0$  então devolva  $A$ ;
- 2  $p =$  Partição( $A, l, r$ );
- 3 QuickSort( $A, l, p-1$ );
- 4 QuickSort( $A, p+1, r$ );
- 5 devolva  $A$ ;

**Algoritmo 2:** Partição

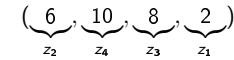
**Entrada:** Um arranjo  $A$ , índices  $l$  e  $r$

**Saída:** O mesmo arranjo mas particionado

- 1 Coloca o pivot em  $A[l]$  ;
- 2  $pivot = A[l]$  ;
- 3  $i = l + 1$ ;
- 4 para  $j = l + 1$  até  $r$  faça
- 5 se  $A[j] < pivot$  então
- 6 Troca  $A[j]$  e  $A[i]$ ;
- 7  $i = i + 1$ ;
- 8 Troca  $A[l]$  e  $A[i - 1]$ ;
- 9 devolva  $i - 1$ ;

• Notação:

- ▶  $z_i =$   $i$ -ésimo menor elemento
- ▶ ou seja,  $z_i$  não é o elemento que está originalmente na  $i$ -ésima posição do vetor, mas sim o elemento que vai ocupar a  $i$ -ésima posição no vetor quando ordenado.



- ▶ para uma escolha  $\sigma$ , e  $i < j$ , seja  $X_{ij}(\sigma) =$  número de vezes que  $z_i$  e  $z_j$  são comparados.

• Fixado dois elementos da entrada, quantas vezes eles podem ser comparados?

- a 1
- b 0 ou 1
- c 0, 1 ou 2
- d algo entre 0 e  $n - 1$

• Podemos então facilmente escrever  $C$  em função de  $X$

$$C(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n X_{ij}(\sigma), \quad \forall \sigma \in \Omega$$

- Pela linearidade da esperança (lembrando que ela se aplica mesmo se as variáveis não forem independentes)

$$E[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n E[X_{ij}].$$

• Como:

$$\begin{aligned}
 E[X_{ij}] &= 0 \cdot Pr[X_{ij} = 0] + 1 \cdot Pr[X_{ij} = 1] \\
 &= 0 \cdot Pr[X_{ij} = 0] + 1 \cdot Pr[X_{ij} = 1] \\
 &= Pr[X_{ij} = 1] = Pr[z_i \text{ e } z_j \text{ serem comparados}]
 \end{aligned}$$

$$E[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n Pr[z_i \text{ e } z_j \text{ serem comparados}]$$

### Lema

Para todo  $i < j$ ,  $Pr[z_i \text{ e } z_j \text{ serem comparados}] = \frac{2}{(j-i+1)}$

- Fixe  $z_i, z_j$  com  $i < j$ .
- Considere o conjunto  $\{z_i, z_{i+1}, \dots, z_{j-1}, z_j\}$ .
- Desde que nenhum desses números seja escolhido como pivô, todos serão passados juntos para a mesma chamada recursiva.
- Considere então o primeiro entres  $\{z_i, z_{i+1}, \dots, z_{j-1}, z_j\}$  que é escolhido como pivô:
  - ▶ se  $z_i$  ou  $z_j$  for escolhido primeiro, eles serão escolhidos.
  - ▶ se escolher um dos outros então  $z_i$  e  $z_j$  nunca serão comparados.
- como a escolha é aleatória qualquer um dos elementos do conjunto tem a mesma chance de ser escolhido primeiro (do conjunto) e portando a chance deles serem comparados é

$$2/(j - i + 1)$$

□

25 / 29

- Então

$$E[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n Pr[z_i \text{ e } z_j \text{ serem comparados}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{2}{(j-i+1)}$$

$$E[C] = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{1}{(j-i+1)}$$

- Agora note que a soma interna

$$\sum_{j=i+1}^n \frac{1}{(j-i+1)} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n-i+1}$$

- e ela obtêm a maior quantidade de termos (e o maior valor) quando  $i = 1$
- A maior soma então é

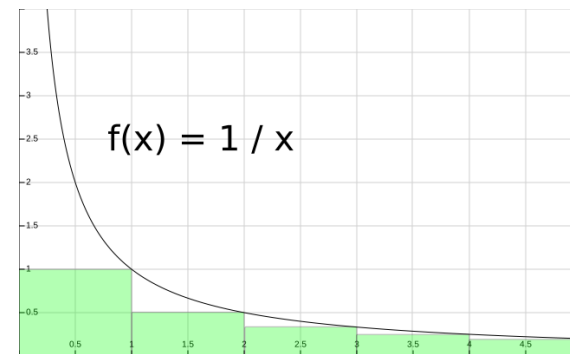
$$\sum_{j=2}^n \frac{1}{j} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n}$$

26 / 29

Limitamos então superiormente a esperança por:

$$\begin{aligned} E[C] &= 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{1}{(j-i+1)} \\ &\leq 2 \cdot n \cdot \sum_{j=2}^n \frac{1}{(j-i+1)} \\ &\leq 2 \cdot n \cdot \sum_{k=2}^n \frac{1}{k} \end{aligned}$$

$$E[C] \leq 2 \cdot n \cdot \sum_{k=2}^n \frac{1}{k}$$



27 / 29

28 / 29

$$E[C] \leq 2 \cdot n \cdot \sum_{k=2}^n \frac{1}{k}$$

A soma agora pode ser limitada superiormente por:

$$\sum_{k=2}^n \frac{1}{k} \leq \int_1^n \frac{1}{x} dx = \ln x \Big|_1^n = \ln n - \ln 1 = \ln n$$

Portanto:

$$E[C] \leq 2 \cdot n \cdot \ln n$$

e portanto:

O tempo de execução esperado do QuickSort é  $O(n \log n)$

□