# Quicksort e Select Aleatorizados

CLRS Secs 7.3, 7.4 e 9.2



# Relembremos o Particione

Rearranja $A[p\,..\,r]$ de modo que $p \le q \le r$ e
$A[p\,..\,q{-}1] \le A[q] < A[q{+}1\,..\,r]$

```
PARTICIONE (A, p, r)
1  x ← A[r]          ▷ x é o "pivô"
2  i ← p−1
3  para j ← p até r − 1 faça
4      se A[j] ≤ x
5          então i ← i + 1
6                A[i] ↔ A[j]
7  A[i+1] ↔ A[r]
8  devolva i + 1
```

Invariantes: no começo de cada iteração de 3–6,

(i0) $A[p\,..\,i] \le x$ (i1) $A[i{+}1\,..\,j{-}1] > x$ (i2) $A[r] = x$

Consumo de tempo: $\Theta(n)$ onde $n := r - p$.



# Quicksort aleatorizado

```
PARTICIONE-ALEA(A, p, r)
1  i ← RANDOM (p, r)
2  A[i] ↔ A[r]
3  devolva PARTICIONE (A, p, r)

QUICKSORT-ALE (A, p, r)
1  se p < r
2      então q ← PARTICIONE-ALEA (A, p, r)
3            QUICKSORT-ALE (A, p, q − 1)
4            QUICKSORT-ALE (A, q + 1, r)
```



# Como analizar?

Vamos estimar o tempo médio...do que, mesmo?

- Fixe uma permutação
- Considere as várias instâncias do QUICKSORT-ALE em cima dela, com a distribuição de probabilidade dada pelos sorteios ao longo da execução.
- Determine o tempo médio para essa permutação.

# Quicksort aleatorizado

```
PARTICIONE-ALEA(A, p, r)
1  i ← RANDOM (p, r)
2  A[i] ↔ A[r]
3  devolva PARTICIONE (A, p, r)

QUICKSORT-ALE (A, p, r)
1  se p < r
2      então q ← PARTICIONE-ALEA (A, p, r)
3            QUICKSORT-ALE (A, p, q − 1)
4            QUICKSORT-ALE (A, q + 1, r)
```

Consumo esperado de tempo?

Basta contar o número esperado de comparações na linha 4 do PARTICIONE.



# Consumo esperado de tempo

Basta contar o número esperado de comparações na linha 4 do PARTICIONE.

```
PARTICIONE (A, p, r)
1  x ← A[r]        ▷ x é o "pivô"
2  i ← p−1
3  para j ← p até r − 1 faça
4      se A[j] ≤ x
5          então i ← i + 1
6                A[i] ↔ A[j]
7  A[i+1] ↔ A[r]
8  devolva i + 1
```



# Consumo de tempo esperado

Suponha $A[p\,..\,r]$ permutação de $1\,..\,n$.

$X_{ab}$ = número de comparações entre $a$ e $b$ na linha 4 do PARTICIONE do QUICKSORT-ALE;

Queremos calcular

$$\begin{aligned} X &= \text{total de comparações "}A[j] \le x\text{"} \\ &= \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} X_{ab} \end{aligned}$$



# Consumo de tempo esperado

Supondo $a < b$,

$$X_{ab} = \begin{cases} 1 & \text{se primeiro pivô em } \{a, \ldots, b\} \text{ é } a \text{ ou } b \\ 0 & \text{caso contrário} \end{cases}$$

Qual a probabilidade de $X_{ab}$ valer 1?

$$\Pr\{X_{ab}{=}1\} = \frac{1}{b-a+1} + \frac{1}{b-a+1} = \mathrm{E}[X_{ab}]$$

$$X = \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} X_{ab}$$

$\mathrm{E}[X] =$ ????

# Consumo de tempo esperado

$$
\begin{aligned}
\mathrm{E}[X] &= \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \mathrm{E}[X_{ab}] \\
&= \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \Pr\{X_{ab}=1\} \\
&= \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \frac{2}{b-a+1} \\
&= \sum_{a=1}^{n-1} \sum_{k=1}^{n-a} \frac{2}{k+1} \\
&< \sum_{a=1}^{n-1} 2\left(\tfrac{1}{1}+\tfrac{1}{2}+\cdots+\tfrac{1}{n}\right) \\
&< 2n\left(\tfrac{1}{1}+\tfrac{1}{2}+\cdots+\tfrac{1}{n}\right) < 2n\,(1+\ln n)
\end{aligned}
$$

CLRS (A.7), p.1060



# Conclusões

O consumo de tempo esperado do algoritmo QUICKSORT-ALE é $\mathrm{O}(n \log n)$.

Do exercício 7.4-4 do CLRS temos que

O consumo de tempo esperado do algoritmo QUICKSORT-ALE é $\Theta(n \log n)$.



# $k$-ésimo menor elemento

CLRS 9



# $k$-ésimo menor

Problema: Encontrar o $k$-ésimo menor elemento de $A[1\,..\,n]$.

Suponha $A[1\,..\,n]$ sem elementos repetidos.

Exemplo: 33 é o 4o. menor elemento de:

| 1 | | | | | | | | | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 99 | 32 | 88 | 34 | **33** | 11 | 97 | 55 | 66 | $A$ |

| 1 | | | 4 | | | | | | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 11 | 22 | 32 | **33** | 34 | 55 | 66 | 88 | 97 | 99 | ordenado |

# Mediana

Mediana é o $\lfloor \frac{n+1}{2} \rfloor$-ésimo menor ou o $\lceil \frac{n+1}{2} \rceil$-ésimo menor elemento.

Exemplo: a mediana é 34 ou 55:

| 1 | | | | | | | | | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 99 | 32 | 88 | **34** | 33 | 11 | 97 | **55** | 66 | $A$ |

| 1 | | | | 5 | 6 | | | | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 11 | 22 | 32 | 33 | **34** | **55** | 66 | 88 | 97 | 99 | ordenado |



# $k$-ésimo menor

Recebe $A[1\,..\,n]$ e $k$ tal que $1 \le k \le n$
e devolve valor do $k$-ésimo menor elemento de $A[1\,..\,n]$.

```
SELECT-ORD (A, n, k)
1  ORDENE (A, n)
2  devolva A[k]
```

O consumo de tempo do SELECT-ORD é $\Theta(n \lg n)$.

Dá para fazer melhor?



# Menor

Recebe um vetor $A[1\,..\,n]$ e devolve o valor do menor elemento.

```
MENOR (A, n)
1  menor ← A[1]
2  para k ← 2 até n faça
3      se A[k] < menor
4          então menor ← A[k]
5  devolva menor
```

O consumo de tempo do algoritmo MENOR é $\Theta(n)$.



# Segundo menor

Recebe um vetor $A[1\,..\,n]$ e devolve o valor do segundo menor elemento, supondo $n \ge 2$.

```
SEG-MENOR (A, n)
1  menor ← min{A[1], A[2]}    segmenor ← max{A[1], A[2]}
2  para k ← 3 até n faça
3      se A[k] < menor
4          então segmenor ← menor
5                menor ← A[k]
6      senão se A[k] < segmenor
7          então segmenor ← A[k]
8  devolva segmenor
```

O consumo de tempo do SEG-MENOR é $\Theta(n)$.

# Algoritmo linear?

Será que conseguimos fazer um algoritmo linear

para a mediana?

para o $k$-ésimo menor?

Sim!

Usaremos o PARTICIONE do QUICKSORT!



# Select aleatorizado

```
PARTICIONE-ALEA(A, p, r)
1  k ← RANDOM(p, r)
2  A[k] ↔ A[r]
3  devolva PARTICIONE(A, p, r)

SELECT-ALEA (A, p, r, k)
1  se p = r então devolva A[p]
2  q ← PARTICIONE-ALEA(A, p, r)
3  se k = q − p + 1
4      então devolva A[q]
5  se k < q − p + 1
6      então devolva SELECT-ALEA(A, p, q − 1, k)
7      senão devolva SELECT-ALEA(A, q+1, r, k−(q−p+1))
```

Consumo esperado de tempo?

$X$ = número de comparações no PARTICIONE-ALEA



# Quebrar em fases

KT 13.5

Uma fase é uma sucessão de chamadas recursivas

a fase muda quando o tamanho do vetor cai de pelo menos 3/4

Na fase $k$, o tamanho do vetor é $\leq \left(\frac{3}{4}\right)^k$

Se $X_k$ é o número de chamadas da fase $k$, vale

$$X \leq \sum_{k \geq 0} X_k n \left(\frac{3}{4}\right)^k$$

Logo,

$$E(X) \leq n \sum_{k \geq 0} E(X_k) \left(\frac{3}{4}\right)^k$$



# Com pivô sortudo

Suponha que cada pivô caia na faixa boa


a chamada recursiva faz mudar de fase.

Difícil dar sorte sempre, mas quanto tempo precisa esperar?

# Um pouco mais de probabilidade

Considere uma sequência de sorteios de bom ou mau, onde a probabilidade de bom é $p > 0$ a cada sorteio.

quantos sorteios em média para aparecer o primeiro bom?

$X =$ instante do primeiro sorteio favorável

$$\Pr[X = j] = (1-p)^{j-1}p$$

$$E(X) = \sum_{j\geq 1} j \Pr[X = j] = \sum_{j\geq 1} j(1-p)^{j-1}p$$

$$\leq p\sum_{j=1}^{\infty} j(1-p)^{j-1} = \frac{1}{p}$$



# Terminando

Para um pivô cair na faixa boa, $p = 1/2$. Logo, o número esperado de sorteios para que o tamanho do vetor caia por $3/4$ é $\leq 2$.
Da análise anterior,

$$E(X) \leq n\sum_{k\geq 0} E(X_k)\left(\frac{3}{4}\right)^k$$

$$\leq 2n\sum_{k\geq 0}\left(\frac{3}{4}\right)^k$$

$$\leq 8n$$



# Conclusões

O consumo de tempo esperado do algoritmo SELECT-ALEA é $O(n)$.



# Próxima aula

CLRS Cap 6 Heapsort