

# Análise de Algoritmos

**Parte destes slides são adaptações de slides  
do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.**

# Quicksort e Select Aleatorizados

CLRS Secs 7.3, 7.4 e 9.2

# Particione

Rearranja  $A[p..r]$  de modo que  $p \leq q \leq r$  e  
 $A[p..q-1] \leq A[q] < A[q+1..r]$

**PARTICIONE** ( $A, p, r$ )

1  $x \leftarrow A[r]$   $\triangleright x$  é o “pivô”

2  $i \leftarrow p-1$

3 **para**  $j \leftarrow p$  **até**  $r-1$  **faça**

4 **se**  $A[j] \leq x$

5 **então**  $i \leftarrow i+1$

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)  $A[p..i] \leq x$       (i1)  $A[i+1..j-1] > x$       (i2)  $A[r] = x$

# Particione

Rearranja  $A[p..r]$  de modo que  $p \leq q \leq r$  e  $A[p..q-1] \leq A[q] < A[q+1..r]$

**PARTICIONE** ( $A, p, r$ )

1  $x \leftarrow A[r]$   $\triangleright x$  é o “pivô”

2  $i \leftarrow p-1$

3 **para**  $j \leftarrow p$  **até**  $r-1$  **faça**

4 **se**  $A[j] \leq x$

5 **então**  $i \leftarrow i + 1$

6  $A[i] \leftrightarrow A[j]$

7  $A[i+1] \leftrightarrow A[r]$

8 **devolva**  $i + 1$

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

# Quicksort aleatorizado

PARTICIONE-ALEA( $A, p, r$ )

1  $i \leftarrow \text{RANDOM}(p, r)$

2  $A[i] \leftrightarrow A[r]$

3 **devolva** PARTICIONE( $A, p, r$ )

# Quicksort aleatorizado

PARTICIONE-ALEA( $A, p, r$ )

- 1  $i \leftarrow \text{RANDOM}(p, r)$
- 2  $A[i] \leftrightarrow A[r]$
- 3 **devolva** PARTICIONE( $A, p, r$ )

QUICKSORT-ALE( $A, p, r$ )

- 1 **se**  $p < r$
- 2     **então**  $q \leftarrow \text{PARTICIONE-ALEA}(A, p, r)$
- 3             QUICKSORT-ALE( $A, p, q - 1$ )
- 4             QUICKSORT-ALE( $A, q + 1, r$ )

# Quicksort aleatorizado

PARTICIONE-ALEA( $A, p, r$ )

1  $i \leftarrow \text{RANDOM}(p, r)$

2  $A[i] \leftrightarrow A[r]$

3 **devolva** PARTICIONE( $A, p, r$ )

QUICKSORT-ALE( $A, p, r$ )

1 **se**  $p < r$

2     **então**  $q \leftarrow \text{PARTICIONE-ALEA}(A, p, r)$

3             QUICKSORT-ALE( $A, p, q - 1$ )

4             QUICKSORT-ALE( $A, q + 1, r$ )

Consumo esperado de tempo?

# Quicksort aleatorizado

PARTICIONE-ALEA( $A, p, r$ )

1  $i \leftarrow \text{RANDOM}(p, r)$

2  $A[i] \leftrightarrow A[r]$

3 **devolva** PARTICIONE( $A, p, r$ )

QUICKSORT-ALE( $A, p, r$ )

1 **se**  $p < r$

2     **então**  $q \leftarrow \text{PARTICIONE-ALEA}(A, p, r)$

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   $x \leftarrow A[r]$        $\triangleright x$  é o “pivô”
2   $i \leftarrow p-1$ 
3  para  $j \leftarrow p$  até  $r-1$  faça
4      se  $A[j] \leq x$ 
5          então  $i \leftarrow i+1$ 
6               $A[i] \leftrightarrow A[j]$ 
7   $A[i+1] \leftrightarrow A[r]$ 
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

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

# Consumo de tempo esperado

Supondo  $a < b$ ,

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

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

# Consumo de tempo esperado

Supondo  $a < b$ ,

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

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

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

# Consumo de tempo esperado

Supondo  $a < b$ ,

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

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

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

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

$$\mathbb{E}[X] = \text{????}$$

# Consumo de tempo esperado

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

CLRS (A.7), p.1060

# Conclusões

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

Do **exercício 7.4-4 do CLRS** temos que

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

# Select aleatorizado

**PARTICIONE-ALEA**( $A, p, r$ )

- 1  $i \leftarrow \text{RANDOM}(p, r)$
- 2  $A[i] \leftrightarrow A[r]$
- 3 **devolva** **PARTICIONE** ( $A, p, r$ )

**SELECT-ALEA** ( $A, p, r, k$ )

- 1 **se**  $p < r$  **então**
- 2      $q \leftarrow \text{PARTICIONE-ALEA}(A, p, r)$
- 3     **se**  $k = q - p + 1$
- 4         **então devolva**  $A[q]$
- 5     **se**  $k < q - p + 1$
- 6         **então** **SELECT-ALEA** ( $A, p, q - 1, k$ )
- 7         **senão** **SELECT-ALEA** ( $A, q + 1, r, k - (q - p + 1)$ )



# Select aleatorizado

PARTICIONE-ALEA( $A, p, r$ )

- 1  $i \leftarrow \text{RANDOM}(p, r)$
- 2  $A[i] \leftrightarrow A[r]$
- 3 **devolva** PARTICIONE( $A, p, r$ )

SELECT-ALEA( $A, p, r, k$ )

- 1 **se**  $p < r$  **então**
- 2      $q \leftarrow \text{PARTICIONE-ALEA}(A, p, r)$
- 3     **se**  $k = q - p + 1$
- 4         **então devolva**  $A[q]$
- 5     **se**  $k < q - p + 1$
- 6         **então** SELECT-ALEA( $A, p, q - 1, k$ )
- 7         **senão** SELECT-ALEA( $A, q + 1, r, k - (q - p + 1)$ )

Consumo esperado de tempo?

# 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 **SELECT-ALEA**.

Observe que  $X_{ab}$  não é a mesma de antes.

De novo, queremos calcular

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

# Consumo de tempo esperado

Vamos supor que  $k = n$ .

Supondo  $a < b$ ,

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

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

# Consumo de tempo esperado

Vamos supor que  $k = n$ .

Supondo  $a < b$ ,

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

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

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

# Consumo de tempo esperado

Vamos supor que  $k = n$ .

Supondo  $a < b$ ,

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

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

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

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

$$\mathbb{E}[X] = \text{????}$$

# Consumo de tempo esperado

$$\begin{aligned} E[X] &= \sum_{a=1}^{n-1} \sum_{b=a+1}^n E[X_{ab}] = \sum_{a=1}^{n-1} \sum_{b=a+1}^n \Pr\{X_{ab}=1\} \\ &= \sum_{a=1}^{n-1} \sum_{b=a+1}^n \frac{2}{n-a+1} \\ &= \sum_{a=1}^{n-1} \frac{2(n-a)}{n-a+1} \\ &< \sum_{a=1}^{n-1} 2 < 2n. \end{aligned}$$

# Consumo de tempo esperado

$$\begin{aligned} E[X] &= \sum_{a=1}^{n-1} \sum_{b=a+1}^n E[X_{ab}] = \sum_{a=1}^{n-1} \sum_{b=a+1}^n \Pr\{X_{ab}=1\} \\ &= \sum_{a=1}^{n-1} \sum_{b=a+1}^n \frac{2}{n-a+1} \\ &= \sum_{a=1}^{n-1} \frac{2(n-a)}{n-a+1} \\ &< \sum_{a=1}^{n-1} 2 < 2n. \end{aligned}$$

**Exercício:** Refaça os cálculos para um  $k$  arbitrário.

# Conclusões

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

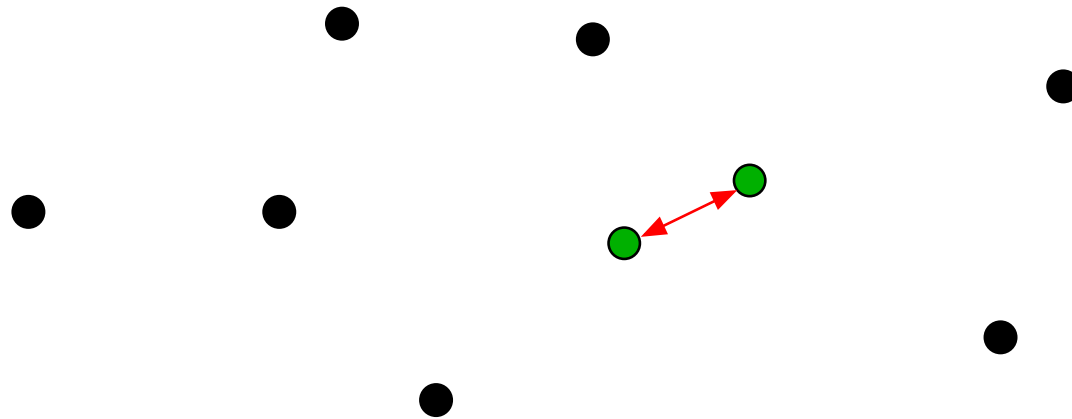


# Par de Pontos Mais Próximos

KT Sec 13.7

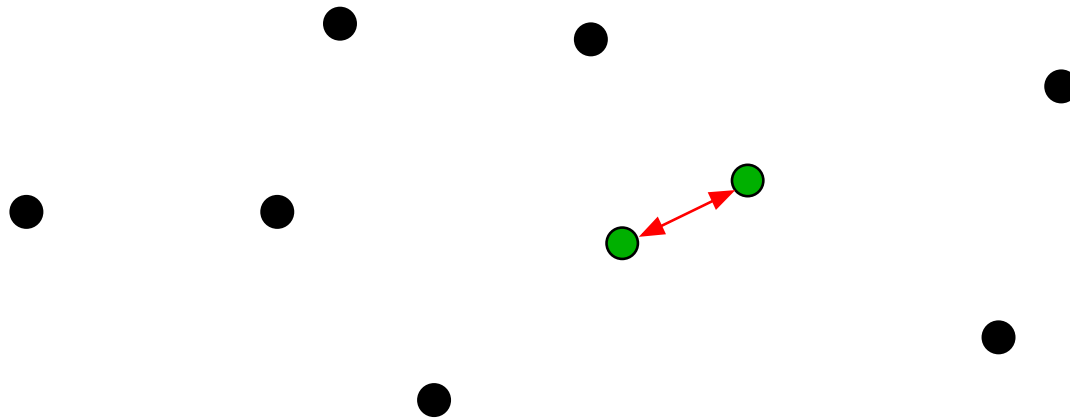
# Par de pontos mais próximos

**Problema:** Dados  $n$  pontos no plano, determinar dois deles que estão à distância mínima.



# Par de pontos mais próximos

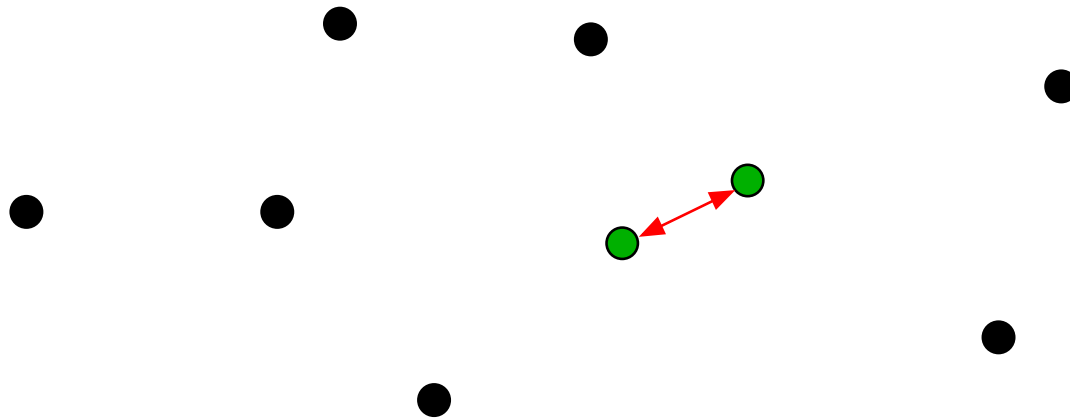
**Problema:** Dados  $n$  pontos no plano, determinar dois deles que estão à distância mínima.



**Esta aula:** algoritmo aleatorizado cujo consumo esperado de tempo é  $O(n)$ .

# Par de pontos mais próximos

**Problema:** Dados  $n$  pontos no plano, determinar dois deles que estão à distância mínima.



**Esta aula:** algoritmo aleatorizado cujo consumo esperado de tempo é  $O(n)$ .

**Hipótese simplificadora:** todos os pontos estão no quadrado  $[0, 1] \times [0, 1]$ .

# Esboço do algoritmo

**Problema:** Dados  $n$  pontos no quadrado  $[0, 1] \times [0, 1]$ , determinar dois deles que estão à distância mínima.

# Esboço do algoritmo

**Problema:** Dados  $n$  pontos no quadrado  $[0, 1] \times [0, 1]$ , determinar dois deles que estão à distância mínima.

Seja  $\delta = \text{dist}(p_1, p_2)$  e  $Q_{i,k} = \left[ \frac{i\delta}{2}, \frac{(i+1)\delta}{2} \right) \times \left[ \frac{k\delta}{2}, \frac{(k+1)\delta}{2} \right)$ , para  $i, k = 0, \dots, N$ , onde  $N = \lceil 2/\delta \rceil$ .

# Esboço do algoritmo

**Problema:** Dados  $n$  pontos no quadrado  $[0, 1] \times [0, 1]$ , determinar dois deles que estão à distância mínima.

Seja  $\delta = \text{dist}(p_1, p_2)$  e  $Q_{i,k} = \left[ \frac{i\delta}{2}, \frac{(i+1)\delta}{2} \right) \times \left[ \frac{k\delta}{2}, \frac{(k+1)\delta}{2} \right)$ , para  $i, k = 0, \dots, N$ , onde  $N = \lceil 2/\delta \rceil$ .

Para  $j \leftarrow 3$  até  $n$  faça:

Seja  $S$  o conjunto de pares  $(i, k)$  tais que existe um ponto  $p_\ell$  com  $\ell < j$  no quadrado  $Q_{i,k}$ .

# Esboço do algoritmo

**Problema:** Dados  $n$  pontos no quadrado  $[0, 1] \times [0, 1]$ , determinar dois deles que estão à distância mínima.

Seja  $\delta = \text{dist}(p_1, p_2)$  e  $Q_{i,k} = \left[\frac{i\delta}{2}, \frac{(i+1)\delta}{2}\right) \times \left[\frac{k\delta}{2}, \frac{(k+1)\delta}{2}\right)$ , para  $i, k = 0, \dots, N$ , onde  $N = \lceil 2/\delta \rceil$ .

Para  $j \leftarrow 3$  até  $n$  faça:

Seja  $S$  o conjunto de pares  $(i, k)$  tais que existe um ponto  $p_\ell$  com  $\ell < j$  no quadrado  $Q_{i,k}$ .

Calcule  $(r, s)$  tal que  $p_j$  está em  $Q_{r,s}$ .

Para cada  $(i, k)$  em  $S$  tal que  $|r - i| \leq 2$  e  $|s - k| \leq 2$  calcule  $\text{dist}(p_j, p_\ell)$  onde  $\ell < j$  e  $p_\ell \in Q_{i,k}$  e atualize  $\delta$  quando necessário.



# Esboço do algoritmo

**Problema:** Dados  $n$  pontos no quadrado  $[0, 1] \times [0, 1]$ , determinar dois deles que estão à distância mínima.

Seja  $\delta = \text{dist}(p_1, p_2)$  e  $Q_{i,k} = \left[\frac{i\delta}{2}, \frac{(i+1)\delta}{2}\right) \times \left[\frac{k\delta}{2}, \frac{(k+1)\delta}{2}\right)$ , para  $i, k = 0, \dots, N$ , onde  $N = \lceil 2/\delta \rceil$ .

Para  $j \leftarrow 3$  até  $n$  faça:

Seja  $S$  o conjunto de pares  $(i, k)$  tais que existe um ponto  $p_\ell$  com  $\ell < j$  no quadrado  $Q_{i,k}$ .

Calcule  $(r, s)$  tal que  $p_j$  está em  $Q_{r,s}$ .

Para cada  $(i, k)$  em  $S$  tal que  $|r - i| \leq 2$  e  $|s - k| \leq 2$  calcule  $\text{dist}(p_j, p_\ell)$  onde  $\ell < j$  e  $p_\ell \in Q_{i,k}$  e atualize  $\delta$  quando necessário.

Devolva  $\delta$ .

# Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser  $O(n)$ ?

# Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser  $O(n)$ ?

Que ED usar para armazenar  $S$ ?

# Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser  $O(n)$ ?

Que ED usar para armazenar  $S$ ?

Que operações sofre  $S$ ?

# Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser  $O(n)$ ?

Que ED usar para armazenar  $S$ ?

Que operações sofre  $S$ ?

Por iteração, uma inserção e algumas consultas.

Em algumas iterações,  $S$  muda totalmente...

# Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser  $O(n)$ ?

Que ED usar para armazenar  $S$ ?

Que operações sofre  $S$ ?

Por iteração, uma inserção e algumas consultas.

Em algumas iterações,  $S$  muda totalmente...

Seria bom se...

inserções e buscas consumissem tempo (esperado)  $O(1)$ .

# Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser  $O(n)$ ?

Que ED usar para armazenar  $S$ ?

Que operações sofre  $S$ ?

Por iteração, uma inserção e algumas consultas.

Em algumas iterações,  $S$  muda totalmente...

Seria bom se...

inserções e buscas consumissem tempo (esperado)  $O(1)$ .

Que ED atinge isso?

# Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser  $O(n)$ ?

Que ED usar para armazenar  $S$ ?

Que operações sofre  $S$ ?

Por iteração, uma inserção e algumas consultas.

Em algumas iterações,  $S$  muda totalmente...

Seria bom se...

inserções e buscas consumissem tempo (esperado)  $O(1)$ .

Que ED atinge isso? Hashing!



# Primeira tentativa

DISTÂNCIA( $p, n$ )

1  $\delta \leftarrow \text{dist}(p_1, p_2)$

2 seja  $(i_\ell, k_\ell)$  tal que  $p_\ell \in Q_{i_\ell, k_\ell}$  para  $\ell = 1, 2$

3 **Crie-Hashing**( $H$ )    **Insira**( $H, i_1, k_1, p_1$ )    **Insira**( $H, i_2, k_2, p_2$ )

# Primeira tentativa

DISTÂNCIA( $p, n$ )

1  $\delta \leftarrow \text{dist}(p_1, p_2)$

2 seja  $(i_\ell, k_\ell)$  tal que  $p_\ell \in Q_{i_\ell, k_\ell}$  para  $\ell = 1, 2$

3 **Crie-Hashing**( $H$ )    **Insira**( $H, i_1, k_1, p_1$ )    **Insira**( $H, i_2, k_2, p_2$ )

4 **para**  $j \leftarrow 3$  **até**  $n$  **faça**

5     seja  $(r, s)$  tal que  $p_j \in Q_{r, s}$

6     **para**  $t \leftarrow -2$  **até**  $2$  **faça**

7         **para**  $u \leftarrow -2$  **até**  $2$  **faça**

8             **se** **Pertence**( $H, r + t, s + u$ )

# Primeira tentativa

DISTÂNCIA( $p, n$ )

1  $\delta \leftarrow \text{dist}(p_1, p_2)$

2 seja  $(i_\ell, k_\ell)$  tal que  $p_\ell \in Q_{i_\ell, k_\ell}$  para  $\ell = 1, 2$

3 **Crie-Hashing**( $H$ )    **Insira**( $H, i_1, k_1, p_1$ )    **Insira**( $H, i_2, k_2, p_2$ )

4 **para**  $j \leftarrow 3$  **até**  $n$  **faça**

5     seja  $(r, s)$  tal que  $p_j \in Q_{r, s}$

6     **para**  $t, u \leftarrow -2$  **até**  $2$  **faça**

7         **se** **Pertence**( $H, r + t, s + u$ )

# Primeira tentativa

DISTÂNCIA( $p, n$ )

1  $\delta \leftarrow \text{dist}(p_1, p_2)$

2 seja  $(i_\ell, k_\ell)$  tal que  $p_\ell \in Q_{i_\ell, k_\ell}$  para  $\ell = 1, 2$

3 **Crie-Hashing**( $H$ )    **Insira**( $H, i_1, k_1, p_1$ )    **Insira**( $H, i_2, k_2, p_2$ )

4 **para**  $j \leftarrow 3$  **até**  $n$  **faça**

5     seja  $(r, s)$  tal que  $p_j \in Q_{r, s}$

6     **para**  $t, u \leftarrow -2$  **até**  $2$  **faça**

7         **se** **Pertence**( $H, r + t, s + u$ )

8             **então** seja  $p_\ell \in Q_{r+t, s+u}$  com  $\ell < j$

9                 **se**  $\delta > \text{dist}(p_j, p_\ell)$  **então**  $\delta \leftarrow \text{dist}(p_j, p_\ell)$

# Primeira tentativa

DISTÂNCIA( $p, n$ )

```
1   $\delta \leftarrow \text{dist}(p_1, p_2)$ 
2  seja  $(i_\ell, k_\ell)$  tal que  $p_\ell \in Q_{i_\ell, k_\ell}$  para  $\ell = 1, 2$ 
3  Crie-Hashing( $H$ )   Insira( $H, i_1, k_1, p_1$ )   Insira( $H, i_2, k_2, p_2$ )
4  para  $j \leftarrow 3$  até  $n$  faça
5     seja  $(r, s)$  tal que  $p_j \in Q_{r, s}$ 
6     para  $t, u \leftarrow -2$  até  $2$  faça
7         se Pertence( $H, r + t, s + u$ )
8             então seja  $p_\ell \in Q_{r+t, s+u}$  com  $\ell < j$ 
9                 se  $\delta > \text{dist}(p_j, p_\ell)$  então  $\delta \leftarrow \text{dist}(p_j, p_\ell)$ 
10        se  $\delta$  foi alterado nessa iteração
11            então Reconstrua( $H, p, j$ )
12            senão Insira( $H, r, s, p_j$ )
13  devolva  $\delta$ 
```

# Primeira tentativa

DISTÂNCIA( $p, n$ )

```
1   $\delta \leftarrow \text{dist}(p_1, p_2)$ 
2  seja  $(i_\ell, k_\ell)$  tal que  $p_\ell \in Q_{i_\ell, k_\ell}$  para  $\ell = 1, 2$ 
3  Crie-Hashing( $H$ )   Insira( $H, i_1, k_1, p_1$ )   Insira( $H, i_2, k_2, p_2$ )
4  para  $j \leftarrow 3$  até  $n$  faça
5     seja  $(r, s)$  tal que  $p_j \in Q_{r, s}$ 
6     para  $t, u \leftarrow -2$  até  $2$  faça
7         se Pertence( $H, r + t, s + u$ )
8             então seja  $p_\ell \in Q_{r+t, s+u}$  com  $\ell < j$ 
9                 se  $\delta > \text{dist}(p_j, p_\ell)$  então  $\delta \leftarrow \text{dist}(p_j, p_\ell)$ 
10        se  $\delta$  foi alterado nessa iteração
11            então Reconstrua( $H, p, j$ )
12            senão Insira( $H, r, s, p_j$ )
13 devolva  $\delta$ 
```

**Consumo de tempo esperado:**  $O(n)$  exceto pela linha 11.

# Versão final

DISTÂNCIA( $p, n$ )

```
1  Embaralhe( $p, n$ )    ▷ permutação aleatória dos pontos dados
2   $\delta \leftarrow dist(p_1, p_2)$ 
3  seja  $(i_\ell, k_\ell)$  tal que  $p_\ell \in Q_{i_\ell, k_\ell}$  para  $\ell = 1, 2$ 
4  Crie-Hashing( $H$ )    Insira( $H, i_1, k_1, p_1$ )    Insira( $H, i_2, k_2, p_2$ )
5  para  $j \leftarrow 3$  até  $n$  faça
6      seja  $(r, s)$  tal que  $p_j \in Q_{r, s}$ 
7      para  $t, u \leftarrow -1$  até  $1$  faça
8          se Pertence( $H, r + t, s + u$ )
9              então seja  $p_\ell \in Q_{r+t, s+u}$  com  $\ell < j$ 
10                 se  $\delta < dist(p_j, p_\ell)$  então  $\delta \leftarrow dist(p_j, p_\ell)$ 
11     se  $\delta$  foi alterado nessa iteração
12         então Reconstrua( $H, p, j$ )
13         senão Insira( $H, r, s, p_j$ )
14 devolva  $\delta$ 
```

# Versão final

DISTÂNCIA( $p, n$ )

```
1  Embaralhe( $p, n$ )    ▷ permutação aleatória dos pontos dados
2   $\delta \leftarrow dist(p_1, p_2)$ 
3  seja  $(i_\ell, k_\ell)$  tal que  $p_\ell \in Q_{i_\ell, k_\ell}$  para  $\ell = 1, 2$ 
4  Crie-Hashing( $H$ )   Insira( $H, i_1, k_1, p_1$ )   Insira( $H, i_2, k_2, p_2$ )
5  para  $j \leftarrow 3$  até  $n$  faça
6      seja  $(r, s)$  tal que  $p_j \in Q_{r, s}$ 
7      para  $t, u \leftarrow -1$  até  $1$  faça
8          se Pertence( $H, r + t, s + u$ )
9              então seja  $p_\ell \in Q_{r+t, s+u}$  com  $\ell < j$ 
10                 se  $\delta < dist(p_j, p_\ell)$  então  $\delta \leftarrow dist(p_j, p_\ell)$ 
11     se  $\delta$  foi alterado nessa iteração
12         então Reconstrua( $H, p, j$ )
13     senão Insira( $H, r, s, p_j$ )
14 devolva  $\delta$ 
```

Qual é o consumo de tempo esperado?



# Consumo de tempo

Seja  $X$  o número de inserções em  $H$ .

O consumo de tempo é proporcional a  $E[X]$ .

# Consumo de tempo

Seja  $X$  o número de inserções em  $H$ .

O consumo de tempo é proporcional a  $E[X]$ .

Seja  $X_j$  a variável binária que vale 1 sse a linha 12 foi executada na iteração  $j$ .

$$X = \sum_{j=1}^n (1 + (j-1)X_j) \leq n + \sum_{j=1}^n j X_j$$

# Consumo de tempo

Seja  $X$  o número de inserções em  $H$ .

O consumo de tempo é proporcional a  $E[X]$ .

Seja  $X_j$  a variável binária que vale 1 sse a linha 12 foi executada na iteração  $j$ .

$$X = \sum_{j=1}^n (1 + (j-1)X_j) \leq n + \sum_{j=1}^n j X_j$$

Mas então

$$E[X] \leq n + \sum_{j=1}^n j E[X_j] = n + \sum_{j=1}^n j \Pr\{X_j = 1\}.$$

# Consumo de tempo

Seja  $X$  o número de inserções em  $H$ .

O consumo de tempo é proporcional a  $E[X]$ .

Seja  $X_j$  a variável binária que vale 1 sse a linha 12 foi executada na iteração  $j$ .

$$X = \sum_{j=1}^n (1 + (j-1)X_j) \leq n + \sum_{j=1}^n j X_j$$

Mas então

$$E[X] \leq n + \sum_{j=1}^n j E[X_j] = n + \sum_{j=1}^n j \Pr\{X_j = 1\}.$$

Note que  $\Pr\{X_j = 1\} = 2/j$ , logo  $E[X] \leq n + 2n = 3n$ .

# Hashing universal

$U$ : conjunto universo (contém todas as possíveis chaves).

$n$ : um número muito menor que  $|U|$ .

$\mathcal{H}$ : conjunto de funções de  $U$  em  $\{0, \dots, n - 1\}$ .

# Hashing universal

$U$ : conjunto universo (contém todas as possíveis chaves).

$n$ : um número muito menor que  $|U|$ .

$\mathcal{H}$ : conjunto de funções de  $U$  em  $\{0, \dots, n - 1\}$ .

Se, para cada par de chaves  $k, \ell$  em  $U$ , o número de funções  $h$  em  $\mathcal{H}$  tais que  $h(k) = h(\ell)$  é no máximo  $|\mathcal{H}|/n$ , então  $\mathcal{H}$  é uma **coleção universal** de hashing (para  $U$  e  $n$ ).

# Hashing universal

$U$ : conjunto universo (contém todas as possíveis chaves).

$n$ : um número muito menor que  $|U|$ .

$\mathcal{H}$ : conjunto de funções de  $U$  em  $\{0, \dots, n - 1\}$ .

Se, para cada par de chaves  $k, \ell$  em  $U$ , o número de funções  $h$  em  $\mathcal{H}$  tais que  $h(k) = h(\ell)$  é no máximo  $|\mathcal{H}|/n$ , então  $\mathcal{H}$  é uma **coleção universal** de hashing (para  $U$  e  $n$ ).

**Teorema:** Seja  $\mathcal{H}$  uma coleção universal de hashing para  $U$  e  $n$ , seja  $S \subseteq U$  tal que  $|S| \leq n$  e  $u \in U$ . Se  $h$  é escolhida aleatoriamente de  $\mathcal{H}$  e  $X$  é o número de elementos  $s$  em  $S$  tais que  $h(s) = h(u)$ , então  $E[X] \leq 1$ .

Prova feita em aula.

# Exemplo de coleção universal de hashing

Seja  $p$  um primo tal que  $U \subseteq \{0, \dots, p - 1\}$ .

$\mathbb{Z}_p$ : conjunto  $\{0, \dots, p - 1\}$ .

$\mathbb{Z}_p^*$ : conjunto  $\{1, \dots, p - 1\}$ .



# Exemplo de coleção universal de hashing

Seja  $p$  um primo tal que  $U \subseteq \{0, \dots, p - 1\}$ .

$\mathbb{Z}_p$ : conjunto  $\{0, \dots, p - 1\}$ .

$\mathbb{Z}_p^*$ : conjunto  $\{1, \dots, p - 1\}$ .

Para todo  $a$  em  $\mathbb{Z}_p^*$  e  $b$  em  $\mathbb{Z}_p$ , seja

$$h_{a,b}(k) = ((ak + b) \bmod p) \bmod n,$$

para todo  $k$  em  $U$ .

# Exemplo de coleção universal de hashing

Seja  $p$  um primo tal que  $U \subseteq \{0, \dots, p - 1\}$ .

$\mathbb{Z}_p$ : conjunto  $\{0, \dots, p - 1\}$ .

$\mathbb{Z}_p^*$ : conjunto  $\{1, \dots, p - 1\}$ .

Para todo  $a$  em  $\mathbb{Z}_p^*$  e  $b$  em  $\mathbb{Z}_p$ , seja

$$h_{a,b}(k) = ((ak + b) \bmod p) \bmod n,$$

para todo  $k$  em  $U$ .

A coleção  $\mathcal{H} = \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } b \in \mathbb{Z}_p\}$  é universal.

Prova feita em aula.

# Exemplo de coleção universal de hashing

Seja  $p$  um primo tal que  $U \subseteq \{0, \dots, p - 1\}$ .

$\mathbb{Z}_p$ : conjunto  $\{0, \dots, p - 1\}$ .

$\mathbb{Z}_p^*$ : conjunto  $\{1, \dots, p - 1\}$ .

Para todo  $a$  em  $\mathbb{Z}_p^*$  e  $b$  em  $\mathbb{Z}_p$ , seja

$$h_{a,b}(k) = ((ak + b) \bmod p) \bmod n,$$

para todo  $k$  em  $U$ .

A coleção  $\mathcal{H} = \{h_{a,b} : a \in \mathbb{Z}_p^* \text{ e } b \in \mathbb{Z}_p\}$  é universal.

Prova feita em aula.

(KT Sec 13.6 para hashing)