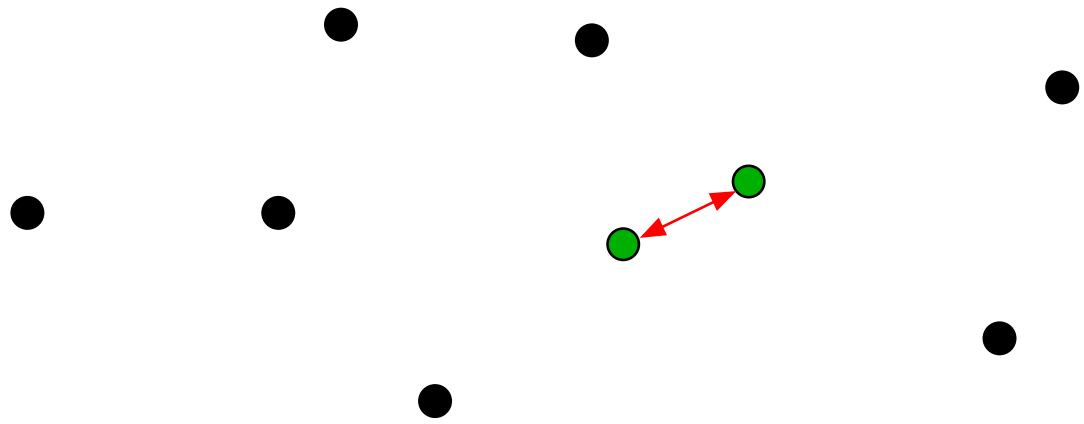
Análise de Algoritmos

Estes slides são adaptações de slides

do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

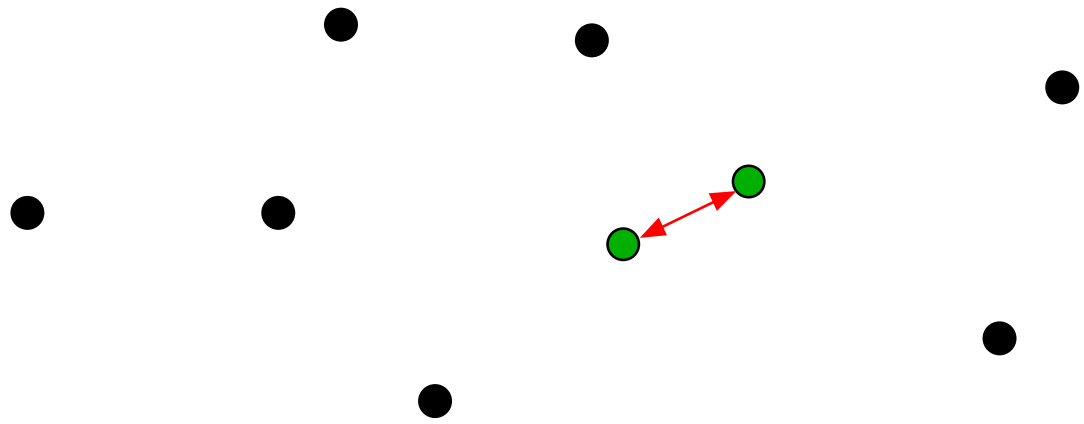
Par de pontos mais próximos

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



Par de pontos mais próximos

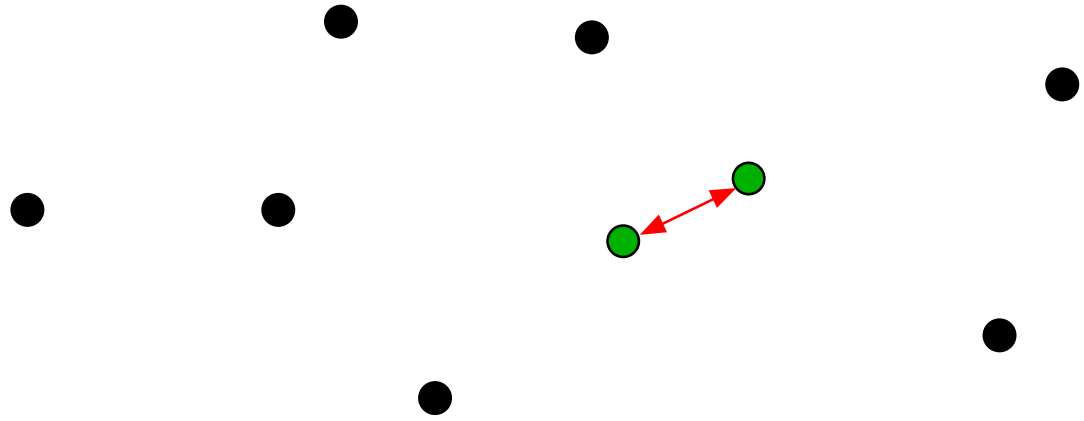
Problema: Dados n pontos no plano,
determinar dois deles que estão à distância mínima.



Esta aula: algoritmo aleatorizado cujo
consumo esperado de tempo é $\mathrm{O}(n)$.

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 $\mathrm{O}(n)$.

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 $\delta = dist(p_1, p_2)$ 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 $i, k = 0, \ldots, N$, 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 $\delta = dist(p_1, p_2)$ 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 $i, k = 0, \ldots, N$, 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_ℓ 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 $\delta = dist(p_1, p_2)$ 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 $i, k = 0, \ldots, N$, 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_ℓ 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 $dist(p_j, p_\ell)$ onde $\ell < j$ e $p_\ell \in Q_{i,k}$

e atualize δ 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 $\delta = dist(p_1, p_2)$ 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 $i, k = 0, \ldots, N$, 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_ℓ 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 $dist(p_j, p_\ell)$ onde $\ell < j$ e $p_\ell \in Q_{i,k}$

e atualize δ quando necessário.

Devolva δ.

Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser $\mathrm{O}(n)$?

Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser $\mathrm{O}(n)$?

Que ED usar para armazenar S?

Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser $\mathrm{O}(n)$?

Que ED usar para armazenar S?

Que operações sofre S?

Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser $\mathrm{O}(n)$?

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 $\mathrm{O}(n)$?

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) $\mathrm{O}(1)$.

Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser $\mathrm{O}(n)$?

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) $\mathrm{O}(1)$.

Que ED atinge isso?

Consumo de tempo estimado

Como fazer para o consumo esperado de tempo ser $\mathrm{O}(n)$?

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) $\mathrm{O}(1)$.

Que ED atinge isso? Hashing!

Primeira tentativa

DISTÂNCIA-SH(p, n)

1 $\delta \leftarrow dist(p_1, p_2)$

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

3 CrieHashing(H) Insira(H, i_1, k_1, p_1) Insira(H, i_2, k_2, p_2)

Primeira tentativa

```
DISTÂNCIA-SH(p, n)
 1   δ ← dist(p_1, p_2)
 2   seja (i_ℓ, k_ℓ) tal que p_ℓ ∈ Q_{i_ℓ,k_ℓ} para ℓ = 1, 2
 3   CrieHashing(H)   Insira(H, i_1, k_1, p_1)   Insira(H, i_2, k_2, p_2)
 4   para j ← 3 até n faça
 5       seja (r, s) tal que p_j ∈ Q_{r,s}
 6       para t ← −2 até 2 faça
 7           para u ← −2 até 2 faça
 8               se Pertence(H, r + t, s + u)
```

Primeira tentativa

DISTÂNCIA-SH(p, n)

```
1   δ ← dist(p_1, p_2)
2   seja (i_ℓ, k_ℓ) tal que p_ℓ ∈ Q_{i_ℓ,k_ℓ} para ℓ = 1, 2
3   CrieHashing(H)   Insira(H, i_1, k_1, p_1)   Insira(H, i_2, k_2, p_2)
4   para j ← 3 até n faça
5       seja (r, s) tal que p_j ∈ Q_{r,s}
6       para t, u ← −2 até 2 faça
7           se Pertence(H, r + t, s + u)
```

Primeira tentativa

```
DISTÂNCIA-SH(p, n)
 1   δ ← dist(p_1, p_2)
 2   seja (i_ℓ, k_ℓ) tal que p_ℓ ∈ Q_{i_ℓ,k_ℓ} para ℓ = 1, 2
 3   CrieHashing(H)   Insira(H, i_1, k_1, p_1)   Insira(H, i_2, k_2, p_2)
 4   para j ← 3 até n faça
 5       seja (r, s) tal que p_j ∈ Q_{r,s}
 6       para t, u ← −2 até 2 faça
 7           se Pertence(H, r + t, s + u)
 8               então seja p_ℓ ∈ Q_{r+t,s+u} com ℓ < j
 9                   se δ > dist(p_j, p_ℓ) então δ ← dist(p_j, p_ℓ)
```

Primeira tentativa

```
DISTÂNCIA-SH(p, n)
 1   δ ← dist(p_1, p_2)
 2   seja (i_ℓ, k_ℓ) tal que p_ℓ ∈ Q_{i_ℓ,k_ℓ} para ℓ = 1, 2
 3   CrieHashing(H)   Insira(H, i_1, k_1, p_1)   Insira(H, i_2, k_2, p_2)
 4   para j ← 3 até n faça
 5       seja (r, s) tal que p_j ∈ Q_{r,s}
 6       para t, u ← −2 até 2 faça
 7           se Pertence(H, r + t, s + u)
 8               então seja p_ℓ ∈ Q_{r+t,s+u} com ℓ < j
 9                     se δ > dist(p_j, p_ℓ) então δ ← dist(p_j, p_ℓ)
10       se δ foi alterado nessa iteração
11           então Reconstrua(H, p, j)
12           senão Insira(H, r, s, p_j)
13   devolva δ
```

Primeira tentativa

```
DISTÂNCIA-SH(p, n)
 1   δ ← dist(p_1, p_2)
 2   seja (i_ℓ, k_ℓ) tal que p_ℓ ∈ Q_{i_ℓ,k_ℓ} para ℓ = 1, 2
 3   CrieHashing(H)   Insira(H, i_1, k_1, p_1)   Insira(H, i_2, k_2, p_2)
 4   para j ← 3 até n faça
 5       seja (r, s) tal que p_j ∈ Q_{r,s}
 6       para t, u ← −2 até 2 faça
 7           se Pertence(H, r + t, s + u)
 8               então seja p_ℓ ∈ Q_{r+t,s+u} com ℓ < j
 9                     se δ > dist(p_j, p_ℓ) então δ ← dist(p_j, p_ℓ)
10       se δ foi alterado nessa iteração
11           então Reconstrua(H, p, j)
12           senão Insira(H, r, s, p_j)
13   devolva δ
```

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

Versão final

```
DISTÂNCIA-SH(p, n)
 1   EMBARALHE(p, n)      ▷ permutação aleatória dos pontos dados
 2   δ ← dist(p_1, p_2)
 3   seja (i_ℓ, k_ℓ) tal que p_ℓ ∈ Q_{i_ℓ,k_ℓ} para ℓ = 1, 2
 4   CrieHashing(H)   Insira(H, i_1, k_1, p_1)   Insira(H, i_2, k_2, p_2)
 5   para j ← 3 até n faça
 6       seja (r, s) tal que p_j ∈ Q_{r,s}
 7       para t, u ← −1 até 1 faça
 8           se Pertence(H, r + t, s + u)
 9               então seja p_ℓ ∈ Q_{r+t,s+u} com ℓ < j
10                       se δ < dist(p_j, p_ℓ) então δ ← dist(p_j, p_ℓ)
11       se δ foi alterado nessa iteração
12           então Reconstrua(H, p, j)
13           senão Insira(H, r, s, p_j)
14   devolva δ
```

Versão final

```
DISTÂNCIA-SH(p, n)
 1   EMBARALHE(p, n)      ▷ permutação aleatória dos pontos dados
 2   δ ← dist(p_1, p_2)
 3   seja (i_ℓ, k_ℓ) tal que p_ℓ ∈ Q_{i_ℓ,k_ℓ} para ℓ = 1, 2
 4   CrieHashing(H)   Insira(H, i_1, k_1, p_1)   Insira(H, i_2, k_2, p_2)
 5   para j ← 3 até n faça
 6       seja (r, s) tal que p_j ∈ Q_{r,s}
 7       para t, u ← −1 até 1 faça
 8           se Pertence(H, r + t, s + u)
 9               então seja p_ℓ ∈ Q_{r+t,s+u} com ℓ < j
10                       se δ < dist(p_j, p_ℓ) então δ ← dist(p_j, p_ℓ)
11       se δ foi alterado nessa iteração
12           então Reconstrua(H, p, j)
13           senão Insira(H, r, s, p_j)
14   devolva δ
```

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 $\mathrm{E}[X]$.

Consumo de tempo

Seja X o número de inserções em H.

O consumo de tempo é proporcional a $\mathrm{E}[X]$.

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 $\mathrm{E}[X]$.

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

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

Consumo de tempo

Seja X o número de inserções em H.

O consumo de tempo é proporcional a $\mathrm{E}[X]$.

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

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

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

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 $\{0, \ldots, n-1\}$.

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 $\{0, \ldots, n-1\}$.

Se, para cada par de chaves k, ℓ 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 $\{0, \ldots, n-1\}$.

Se, para cada par de chaves k, ℓ 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 $|S| = n$ 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 $\mathrm{E}[X] \leq 1$.

Prova feita em aula.

Exemplo de coleção universal de hashing

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

$\mathbb{Z}_p$: conjunto $\{0, \ldots, p-1\}$.
$\mathbb{Z}_p^*$: conjunto $\{1, \ldots, p-1\}$.

Exemplo de coleção universal de hashing

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

$\mathbb{Z}_p$: conjunto $\{0, \ldots, p-1\}$.
$\mathbb{Z}_p^*$: conjunto $\{1, \ldots, p-1\}$.

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 $U \subseteq \{0, \ldots, p-1\}$.

$\mathbb{Z}_p$: conjunto $\{0, \ldots, p-1\}$.
$\mathbb{Z}_p^*$: conjunto $\{1, \ldots, p-1\}$.

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.