

# Análise de Algoritmos

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

# Clustering

$U$ : conjunto de  $n$  itens  $\{p_1, \dots, p_n\}$ .

$d(p_i, p_j)$ : distância entre  $p_i$  e  $p_j$ .

- $d(p_i, p_i) = 0$  para  $i = 1, \dots, n$
- $d(p_i, p_j) = d(p_j, p_i) > 0$  para  $i \neq j$

# Clustering

$U$ : conjunto de  $n$  itens  $\{p_1, \dots, p_n\}$ .

$d(p_i, p_j)$ : distância entre  $p_i$  e  $p_j$ .

- $d(p_i, p_i) = 0$  para  $i = 1, \dots, n$

- $d(p_i, p_j) = d(p_j, p_i) > 0$  para  $i \neq j$

$k$ : número de grupos.

**$k$ -clustering**: partição de  $U$  em  $k$  partes.

# Clustering

$U$ : conjunto de  $n$  itens  $\{p_1, \dots, p_n\}$ .

$d(p_i, p_j)$ : distância entre  $p_i$  e  $p_j$ .

•  $d(p_i, p_i) = 0$  para  $i = 1, \dots, n$

•  $d(p_i, p_j) = d(p_j, p_i) > 0$  para  $i \neq j$

$k$ : número de grupos.

**$k$ -clustering**: partição de  $U$  em  $k$  partes.

**espaçamento** de  $k$ -clustering: distância mínima entre dois elementos de partes distintas do clustering.

# Clustering

$U$ : conjunto de  $n$  itens  $\{p_1, \dots, p_n\}$ .

$d(p_i, p_j)$ : distância entre  $p_i$  e  $p_j$ .

•  $d(p_i, p_i) = 0$  para  $i = 1, \dots, n$

•  $d(p_i, p_j) = d(p_j, p_i) > 0$  para  $i \neq j$

$k$ : número de grupos.

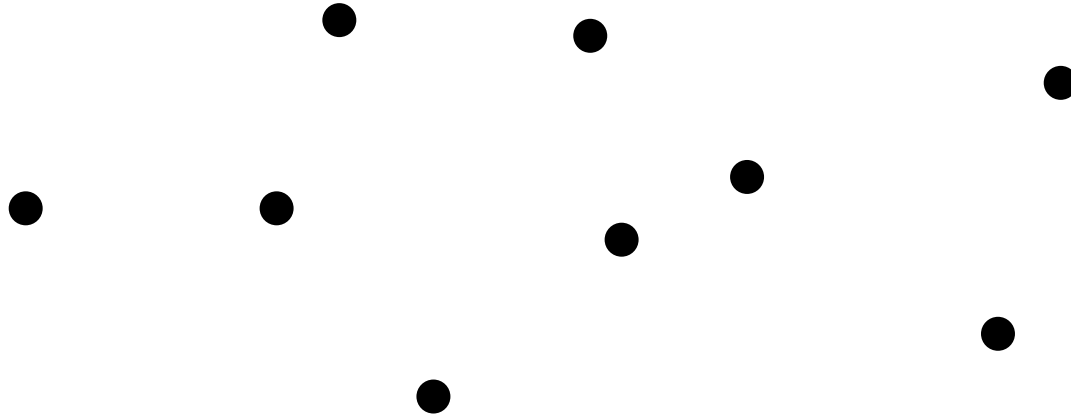
**$k$ -clustering**: partição de  $U$  em  $k$  partes.

**espaçamento** de  $k$ -clustering: distância mínima entre dois elementos de partes distintas do clustering.

**Problema:** Dados  $U$ ,  $d$  e  $k$ ,  
encontrar um  $k$ -clustering com espaçamento máximo.

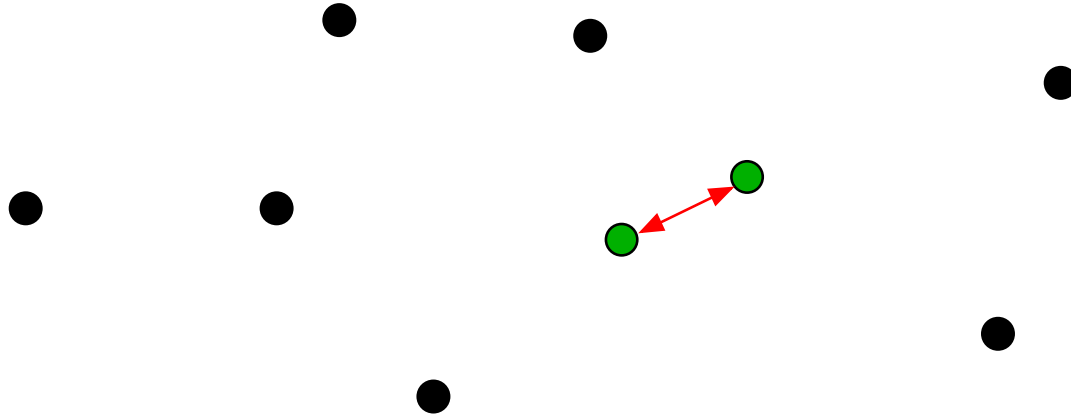
# Clustering

**Problema:** Dados  $U$ ,  $d$  e  $k$ ,  
encontrar um  $k$ -clustering com espaçamento máximo.



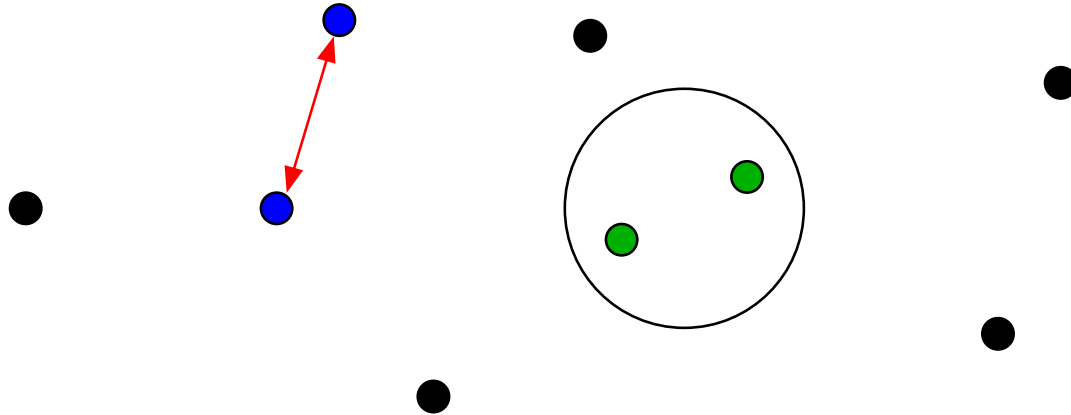
# Clustering

**Problema:** Dados  $U$ ,  $d$  e  $k$ ,  
encontrar um  $k$ -clustering com espaçamento máximo.



# Clustering

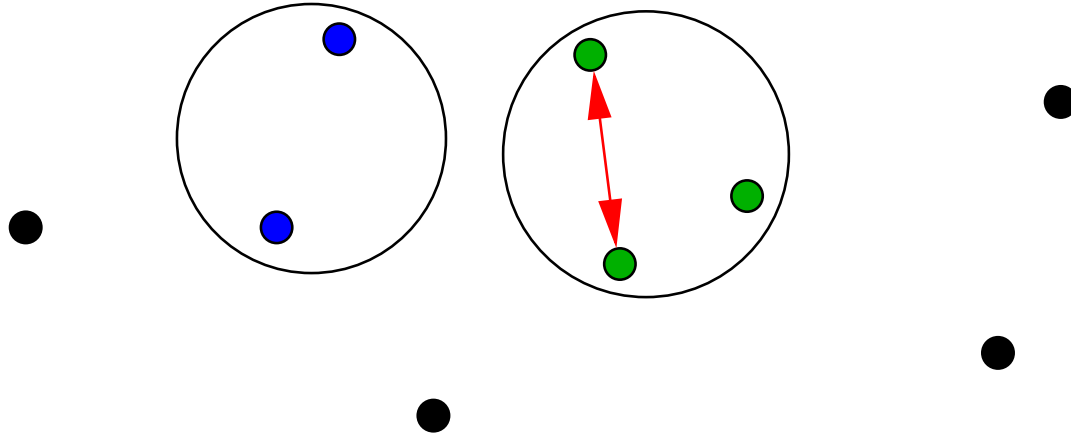
**Problema:** Dados  $U$ ,  $d$  e  $k$ ,  
encontrar um  $k$ -clustering com espaçamento máximo.





# Clustering

**Problema:** Dados  $U$ ,  $d$  e  $k$ ,  
encontrar um  $k$ -clustering com espaçamento máximo.



# Algoritmo inspirado no Kruskal

Lembram do algoritmo de Kruskal?

**Problema:** Dado um grafo  $G = (V, E)$  conexo, e um custo  $c_e > 0$  para cada aresta  $e$  em  $E$ .

Árvore geradora mínima

# Algoritmo inspirado no Kruskal

Lembram do algoritmo de Kruskal?

**Problema:** Dado um grafo  $G = (V, E)$  conexo, e um custo  $c_e > 0$  para cada aresta  $e$  em  $E$ .

Árvore geradora mínima

**MST-KRUSKAL**  $(V, E, c)$

- 1  $e_1, \dots, e_m \leftarrow \text{ORDENE}(E, c)$   $\triangleright$  pelo valor de  $c_e$
- 2  $T \leftarrow \emptyset$   $i \leftarrow 0$
- 3 **enquanto**  $G' = (V, T)$  não é conexo **faça**
- 4  $i \leftarrow i + 1$
- 5 **se**  $T \cup e_i$  não tem circuito
- 6 **então**  $T \leftarrow T \cup e_i$
- 7 **devolva**  $T$

# Algoritmo inspirado no Kruskal

Como adaptar **MST-KRUSKAL** para resolver o  $k$ -clustering?

# Algoritmo inspirado no Kruskal

Como adaptar **MST-KRUSKAL** para resolver o  $k$ -clustering?

Acrescentamos arestas até termos  $k$  componentes.

# Algoritmo inspirado no Kruskal

Como adaptar **MST-KRUSKAL** para resolver o  $k$ -clustering?

Acrescentamos arestas até termos  $k$  componentes.

**CLUSTERING-KRUSKAL** ( $V, E, c, k$ )

```
1   $e_1, \dots, e_m \leftarrow \text{ORDENE}(E, c)$      $\triangleright$  pelo valor de  $c_e$ 
2   $F \leftarrow \emptyset$        $i \leftarrow 0$ 
3  enquanto  $G' = (V, F)$  tem  $> k$  componentes faça
4       $i \leftarrow i + 1$ 
5      se  $F \cup e_i$  não tem circuito
6          então  $F \leftarrow F \cup e_i$ 
7  devolva  $F$ 
```

# Algoritmo inspirado no Kruskal

Como adaptar **MST-KRUSKAL** para resolver o  $k$ -clustering?

Acrescentamos arestas até termos  $k$  componentes.

**CLUSTERING-KRUSKAL** ( $V, E, c, k$ )

```
1   $e_1, \dots, e_m \leftarrow \text{ORDENE}(E, c)$      $\triangleright$  pelo valor de  $c_e$ 
2   $F \leftarrow \emptyset$        $i \leftarrow 0$ 
3  enquanto  $G' = (V, F)$  tem  $> k$  componentes faça
4       $i \leftarrow i + 1$ 
5      se  $F \cup e_i$  não tem circuito
6          então  $F \leftarrow F \cup e_i$ 
7  devolva  $F$ 
```

Este algoritmo resolve o problema?

# Análise da correção

CLUSTERING-KRUSKAL  $(V, E, c, k)$

- 1  $e_1, \dots, e_m \leftarrow \text{ORDENE}(E, c)$   $\triangleright$  pelo valor de  $c_e$
- 2  $F \leftarrow \emptyset$        $i \leftarrow 0$
- 3 **enquanto**  $G' = (V, F)$  tem  $\geq k$  componentes **faça**
- 4       $i \leftarrow i + 1$
- 5      **se**  $F \cup e_i$  não tem circuito
- 6          **então**  $F \leftarrow F \cup e_i$
- 7 **devolva** as componentes de  $F$

Qual é o espaçamento do clustering devolvido?



# Análise da correção

CLUSTERING-KRUSKAL  $(V, E, c, k)$

- 1  $e_1, \dots, e_m \leftarrow \text{ORDENE}(E, c)$   $\triangleright$  pelo valor de  $c_e$
- 2  $F \leftarrow \emptyset$        $i \leftarrow 0$
- 3 **enquanto**  $G' = (V, F)$  tem  $\geq k$  componentes **faça**
- 4       $i \leftarrow i + 1$
- 5      **se**  $F \cup e_i$  não tem circuito
- 6          **então**  $F \leftarrow F \cup e_i$
- 7 **devolva** as componentes de  $F$

Qual é o espaçamento do clustering devolvido?

É o custo da próxima aresta que seria incluída em  $F$ .

# Análise da correção

CLUSTERING-KRUSKAL  $(V, E, c, k)$

- 1  $e_1, \dots, e_m \leftarrow \text{ORDENE}(E, c)$   $\triangleright$  pelo valor de  $c_e$
- 2  $F \leftarrow \emptyset$        $i \leftarrow 0$
- 3 **enquanto**  $G' = (V, F)$  tem  $\geq k$  componentes **faça**
- 4       $i \leftarrow i + 1$
- 5      **se**  $F \cup e_i$  não tem circuito
- 6          **então**  $F \leftarrow F \cup e_i$
- 7 **devolva** as componentes de  $F$

Qual é o espaçamento do clustering devolvido?

É o custo da próxima aresta que seria incluída em  $F$ .

A próxima aresta que não forma circuito com  $F$ .

# Análise da correção

Seja  $e$  a próxima aresta que seria incluída em  $F$ .

# Análise da correção

Seja  $e$  a próxima aresta que seria incluída em  $F$ .

Seja  $\mathcal{C} = \{C_1, \dots, C_k\}$  o  $k$ -clustering devolvido,  
e  $\mathcal{C}^* = \{C_1^*, \dots, C_k^*\}$  um  $k$ -clustering ótimo.

# Análise da correção

Seja  $e$  a próxima aresta que seria incluída em  $F$ .

Seja  $\mathcal{C} = \{C_1, \dots, C_k\}$  o  $k$ -clustering devolvido, e  $\mathcal{C}^* = \{C_1^*, \dots, C_k^*\}$  um  $k$ -clustering ótimo.

O espaçamento de  $\mathcal{C}$  é  $c_e$ .

Basta mostrar que o espaçamento de  $\mathcal{C}^*$  é no máximo  $c_e$ .

# Análise da correção

Seja  $e$  a próxima aresta que seria incluída em  $F$ .

Seja  $\mathcal{C} = \{C_1, \dots, C_k\}$  o  $k$ -clustering devolvido, e  $\mathcal{C}^* = \{C_1^*, \dots, C_k^*\}$  um  $k$ -clustering ótimo.

O espaçamento de  $\mathcal{C}$  é  $c_e$ .

Basta mostrar que o espaçamento de  $\mathcal{C}^*$  é no máximo  $c_e$ .

Algum  $C_i$  não está contido em nenhum  $C_j^*$ .

# Análise da correção

Seja  $e$  a próxima aresta que seria incluída em  $F$ .

Seja  $\mathcal{C} = \{C_1, \dots, C_k\}$  o  $k$ -clustering devolvido, e  $\mathcal{C}^* = \{C_1^*, \dots, C_k^*\}$  um  $k$ -clustering ótimo.

O espaçamento de  $\mathcal{C}$  é  $c_e$ .

Basta mostrar que o espaçamento de  $\mathcal{C}^*$  é no máximo  $c_e$ .

Algum  $C_i$  não está contido em nenhum  $C_j^*$ .

Seja  $j$  tal que  $C_i \cap C_j^* \neq \emptyset$ .

# Análise da correção

Seja  $e$  a próxima aresta que seria incluída em  $F$ .

Seja  $\mathcal{C} = \{C_1, \dots, C_k\}$  o  $k$ -clustering devolvido, e  $\mathcal{C}^* = \{C_1^*, \dots, C_k^*\}$  um  $k$ -clustering ótimo.

O espaçamento de  $\mathcal{C}$  é  $c_e$ .

Basta mostrar que o espaçamento de  $\mathcal{C}^*$  é no máximo  $c_e$ .

Algum  $C_i$  não está contido em nenhum  $C_j^*$ .

Seja  $j$  tal que  $C_i \cap C_j^* \neq \emptyset$ .

Seja  $f$  aresta de  $F$  em  $C_i$  com um único extremo em  $C_j^*$ .



# Análise da correção

Seja  $e$  a próxima aresta que seria incluída em  $F$ .

Seja  $\mathcal{C} = \{C_1, \dots, C_k\}$  o  $k$ -clustering devolvido, e  $\mathcal{C}^* = \{C_1^*, \dots, C_k^*\}$  um  $k$ -clustering ótimo.

O espaçamento de  $\mathcal{C}$  é  $c_e$ .

Basta mostrar que o espaçamento de  $\mathcal{C}^*$  é no máximo  $c_e$ .

Algum  $C_i$  não está contido em nenhum  $C_j^*$ .

Seja  $j$  tal que  $C_i \cap C_j^* \neq \emptyset$ .

Seja  $f$  aresta de  $F$  em  $C_i$  com um único extremo em  $C_j^*$ .

Claro que  $c_f \leq c_e$  e espaçamento de  $\mathcal{C}^*$  é no máximo  $c_f$ .

# Implementação

Como se implementa este algoritmo?

# Implementação

Como se implementa este algoritmo?

Como se implementa o algoritmo de Kruskal?

# Implementação

Como se implementa este algoritmo?

Como se implementa o algoritmo de Kruskal?

Estrutura de dados para conjuntos disjuntos...

# Implementação

Como se implementa este algoritmo?

Como se implementa o algoritmo de Kruskal?

Estrutura de dados para conjuntos disjuntos...

MAKESET( $x$ )

1.  $pai[x] \leftarrow x$
2.  $rank[x] \leftarrow 0$

FINDSET( $x$ )

1. **se**  $x \neq pai[x]$
2.     **então**  $pai[x] \leftarrow \text{FINDSET}(pai[x])$
3. **devolva**  $pai[x]$

# Union-Find

MAKESET( $x$ )

1.  $pai[x] \leftarrow x$
2.  $rank[x] \leftarrow 0$

UNION( $x, y$ )

1.  $x' \leftarrow \text{FINDSET}(x)$
2.  $y' \leftarrow \text{FINDSET}(y)$
3. **se**  $x' \neq y'$
4.     **então** LINK( $x', y'$ )

FINDSET( $x$ )

1. **se**  $x \neq pai[x]$
2.     **então**  $pai[x] \leftarrow \text{FINDSET}(pai[x])$
3. **devolva**  $pai[x]$

LINK( $x, y$ )

1. **se**  $rank[x] > rank[y]$
2.     **então**  $pai[y] \leftarrow x$
3.     **senão**  $pai[x] \leftarrow y$
4.     **se**  $rank[x] = rank[y]$
5.         **então**  $rank[y] \leftarrow rank[y] + 1$

# Union-Find

MAKESET( $x$ )

1.  $pai[x] \leftarrow x$
2.  $rank[x] \leftarrow 0$

UNION( $x, y$ )

1.  $x' \leftarrow \text{FINDSET}(x)$
2.  $y' \leftarrow \text{FINDSET}(y)$
3. **se**  $x' \neq y'$
4.     **então** LINK( $x', y'$ )

FINDSET( $x$ )

1. **se**  $x \neq pai[x]$
2.     **então**  $pai[x] \leftarrow \text{FINDSET}(pai[x])$
3. **devolva**  $pai[x]$

LINK( $x, y$ )

1. **se**  $rank[x] > rank[y]$
2.     **então**  $pai[y] \leftarrow x$
3.     **senão**  $pai[x] \leftarrow y$
4.             **se**  $rank[x] = rank[y]$
5.                     **então**  $rank[y] \leftarrow rank[y] + 1$

**Semana que vem:** análise desta estrutura de dados.