Estrategias algorítmicas

- Dividir para conquistar
- Backtracking
- Algoritmos codiciosos
- Programación dinámica
- Branch & bound

Estrategias algorítmicas

- **Backtracking**
- Algoritmos codiciosos
- Programación dinámica

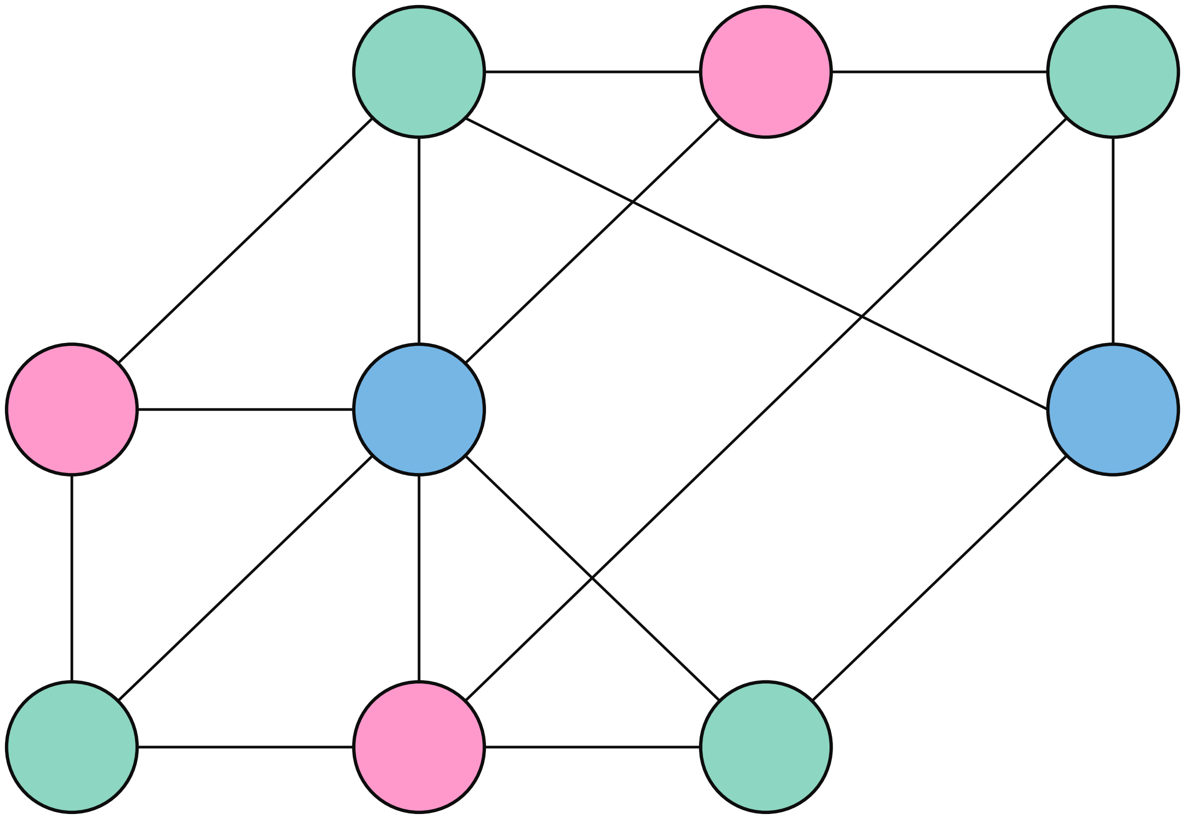
Motivación: Coloración de grafos

Dado un grafo no dirigido $\boldsymbol{G(V,E)}$ y 3 colores, llamamos una **3-coloración** del grafo a una asignación tal que

- Cada nodo tiene asignado color y sólo uno
- Nodos vecinos tienen colores distintos

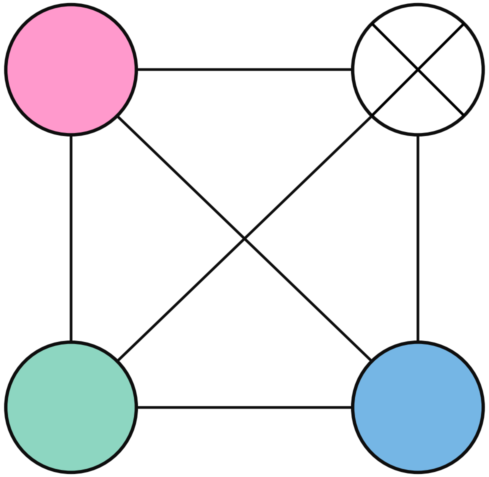
Queremos determinar si un grafo es 3-coloreable

Coloración de grafos



Este grafo es 3-coloreable. ¿Cómo podemos probarlo?

Coloración de grafos



Este grafo no es 3-coloreable. ¿Cómo podemos probarlo?

Constraint Satisfaction Problems

Problemas como este se llaman de **satisfacción de restricciones**

Es una familia entera de problemas de combinatoria

¿Cómo podríamos generalizarlo?

CSPs en general

Tenemos un conjunto de **variables**

Cada variable puede tomar ciertos valores: tiene un **dominio**

Las **restricciones** prohíben ciertas combinaciones de valores

¿Cómo se vería esto en el problema de coloración de grafos?

Modelación de 3-coloración

Tenemos una variable por cada nodo del grafo

El dominio de cada variable es $\{1, 2, 3\}$

Cada arista prohíbe que dos variables tengan el mismo color

¿Cómo lo resolvemos?

Si el problema tiene solución, queremos una garantía

Si no tiene solución, también queremos una garantía

¿Cómo hacemos esto?

Fuerza bruta

Una forma es generar todas las permutaciones posibles

Luego para cada permutación verificar si cumple todas las restricciones

Esta estrategia se conoce como **fuerza bruta**

Sin embargo no es necesario probar **todas** las permutaciones posibles…

¿Es posible?

Dado un problema, ¿es posible resolverlo?

La idea es responder esa pregunta recursivamente

Si asignamos una variable, ¿qué nos queda?

$\mathbf{3 - col}\left(\boldsymbol{G}(\boldsymbol{V},\boldsymbol{E})\right)$:

$\boldsymbol{3 - col}\left(\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})\right)$:

Sea $\boldsymbol{v}$ un nodo sin color en $\boldsymbol{V}$

$\boldsymbol{for}\ \boldsymbol{c} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$:

$\boldsymbol{if}\ \boldsymbol{is\ valid}(\boldsymbol{v}, \boldsymbol{c}, \boldsymbol{E})$:

$\boldsymbol{v}.\mathrm{color} = \boldsymbol{c}$

$\mathbf{3 - col\big(G(V,E)\big)}$:

Sea $\boldsymbol{v}$ un nodo sin color en $\boldsymbol{V}$

$\boldsymbol{for\ c \in \{1,2,3\}}$:

$\quad \boldsymbol{if\ is\ valid(v,c,E)}$:

$\quad\quad \boldsymbol{v}.\text{color} = \boldsymbol{c}$

$\quad\quad \boldsymbol{if\ 3 - col\big(G(V,E)\big)}$:

$\quad\quad\quad \boldsymbol{return\ true}$

$\quad\quad \boldsymbol{v}.\text{color} = \mathbf{0}$

$\boldsymbol{return\ false}$

$\boldsymbol{3 - col}\big(\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})\big)$:

$\boldsymbol{if}$ Todos los nodos tienen color:

$\boldsymbol{return\ true}$

Sea $\boldsymbol{v}$ un nodo sin color en $\boldsymbol{V}$

$\boldsymbol{for}\ \boldsymbol{c} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$:

$\boldsymbol{if\ is\ valid}(\boldsymbol{v}, \boldsymbol{c}, \boldsymbol{E})$:

$\boldsymbol{v}.\mathrm{color} = \boldsymbol{c}$

$\boldsymbol{if}\ \boldsymbol{3 - col}\big(\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})\big)$:

$\boldsymbol{return\ true}$

$\boldsymbol{v}.\mathrm{color} = \mathbf{0}$

$\boldsymbol{return\ false}$

Backtracking

Esta estrategia se conoce como **backtracking**

La idea es **descartar** permutaciones que violan alguna restricción

Eso significa que **siempre** es igual o más rápido que fuerza bruta

```
is solvable(X, R):
    if is solution(X, R):
        return true
    x = choose unassigned variable(X)
    for v ∈ x.d:
        if is valid(x, v, R):
            assign(x, v)
            if is solvable(X, R):
                return true
            unassign(x, v)
    return false
```