# Projet 2. Résolveur de problèmes de satisfaction de contraintes

**Réaliser par :**

**Mtibaa Amal**

**Riahi Mohamed Wassim**

**IMPORTANT :**

Il faut télécharger le package networkx et matplotlib, pour l'affichage de graphes pour coloringMap

## Introduction

On se propose de créer un résolveur de problèmes de satisfaction de contraintes en utilisant des algorithmes généraux

## Explication du travail fait

On a créé une classe csp pour la formalisation de problèmes de satisfaction de contraintes. Cette classe est représentée par un ensemble de variables. On associe à chaque variable un ensemble de domaines et de contraintes.

Le code de cette classe se trouve dans les fichiers **csp.py** et **cspUtil.py**

Notre application sélectionne le problème à partir des fichiers existants contenant la formalisation du CSP. On a sous le dossier **CSP** dans le projet les dossiers **N-Reines**, **Sudoku** et **ColoringMap** qui contiennent chaque une, un fichier pour les contraintes, un fichier pour les variables et un fichier pour les domaines.

Les variables sont écrites sur une seule ligne séparées par un espace.
Le domaine de chaque variable est écrit sur une seule ligne séparé par un espace.
Les contraintes de chaque variable sont écrites chaqu'une dans une ligne de la forme :

<NomVar1> <operateur> <NomVar2>

Les opérateurs développés sont : <, >, <=, >= ,= , != , if (pour 8 reines)
Les contraintes des variables sont séparés par une ligne contenant « ## ».

On a développé 3 algorithmes pour la résolution de ces problèmes qui sont Backtracking, Forward check et  Forward check + Arc consistency AC3

### 1. Backtracking :

Pour l'algorithme backtracking, on a utilisé l'heuristique LCV qui choisit la variable qui a moins de contraintes. Elle retourne un tableau qui contient la valeur 1 et le tableau d'affectation en cas de succès ou 0 et None en cas d'échec.

Le code de cet algorithme se trouve dans **backtrcaking.py**

### 2. Forward checking : Backtracking + Mise à jour domaines

Pour l'algorithme de forward checking, on a utilisé l'heuristique MRV qui choisit la variables qui a le domaine le plus petit.

Le code de cet algorithme se trouve dans **ForwardCheck.py**

### 3. Arc consisty 3+ForwardCheck :

On a développé l'algorithme Arc Consisty 3 dans **ARC.py** et l'algorithme qui réunit l'arc consisty AC3 et forwardCheck dans **forward_CheckAC3.py**

## Interface Graphique

On a créé une interface graphique **Interface.py** qui permet de tester ces algorithmes avec 3 jeux : Sudoku, Coloring Map et 8 reines.

- ✓ Pour le jeu Sudoku, on peut modifier l'état initial de la grille et exécuter les 3 algorithmes.
- ✓ Pour le jeu ColoringMap (comme déjà montrer dans la vidéo **coloringMapExamples)** on peut modifier la map dans les fichiers sous CSP/ColoringMap et exécuter les 3 algorithmes
- ✓ Pour le jeu 8 reines, on peut exécuter le backtracking.

Une trace sur l'exécution se trouve dans la console.