

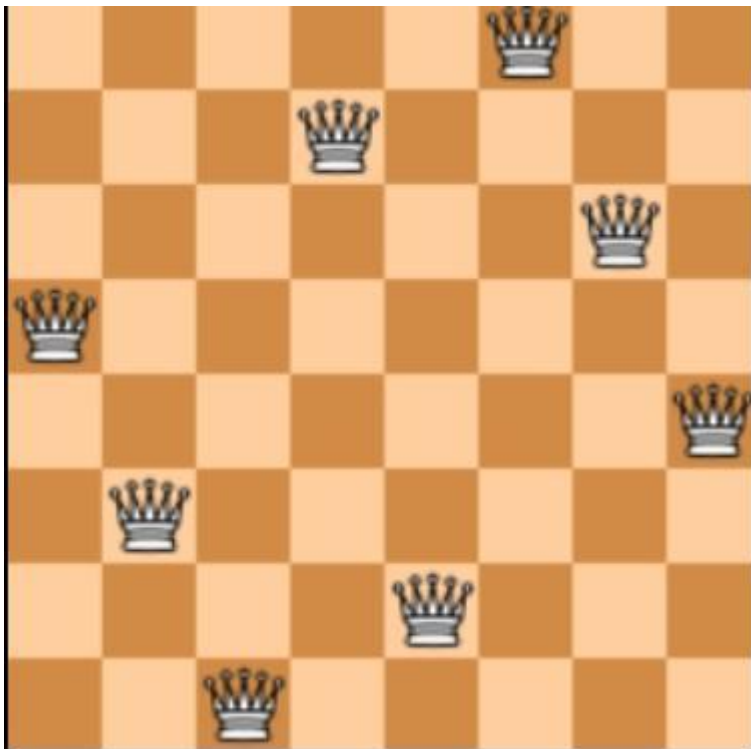
**Achraf Yasmine**  
**Benammar Mahmoud**  
**Giraud Loïc**

## **N-reines**

### **Explication du concept et règles du jeu :**

N-reines est un problème mathématiques classique qui consiste à placer  $n$  reines sur un échiquier de taille  $n \times n$  de telle sorte à ce qu'aucune reine n'attaque une autre. Il doit donc y avoir au plus 1 reine par ligne, colonne et diagonale.

Voici une solution du cas le plus commun de n-reines :  $n = 8$ .



On peut modéliser le problème pour tout  $n \geq 4$ , pour  $n = 2$  et  $n = 3$ , le problème n'a pas de solution.

Nombre de solutions possibles pour chaque n jusqu'à 12 :

| n  | nombre de solutions |
|----|---------------------|
| 1  | 1                   |
| 2  | 0                   |
| 3  | 0                   |
| 4  | 2                   |
| 5  | 10                  |
| 6  | 4                   |
| 7  | 40                  |
| 8  | 92                  |
| 9  | 352                 |
| 10 | 724                 |
| 11 | 2680                |
| 12 | 14200               |

On remarque que le nombre de solutions augmente très vite lorsque n augmente au-delà de 8.

### **Règles de modélisation en forme normale conjonctive :**

On choisit la variable  $R_{ij}$  : la case de la i-ème ligne et de la j-ème colonne est occupée par une reine ( $1 < i, j < n$ ).

**- Minimum une reine par ligne :**

$$\bigwedge i \in \{1, \dots, n\} (\bigvee j \in \{1, \dots, n\} R_{ij})$$

$$(R_{11} \vee R_{12} \vee \dots \vee R_{1n}) \wedge (R_{21} \vee R_{22} \vee \dots \vee R_{2n}) \wedge \dots \wedge (R_{n1} \vee R_{n2} \vee \dots \vee R_{nn})$$

**- Maximum une reine par ligne:**

$$\begin{aligned} & \forall (i,j) \in \{1,\dots,n\}, (\bigwedge i \in \{1,\dots,n\} \bigwedge j \in \{1,\dots,n\} (R(i,j) \rightarrow \bigwedge k \in \{1,\dots,n\} \neg R(i,k)) = \\ & (\bigwedge i \in \{1,\dots,n\} \bigwedge j \in \{1,\dots,n\} \bigwedge k \in \{1,\dots,n\}, (j \neq k) (\neg R(i,j) \vee \neg R(i,k)) \\ & (\neg R_{i1} \vee \neg R_{i2}) \wedge (\neg R_{i1} \vee \neg R_{i3}) \wedge (\neg R_{i1} \vee \neg R_{i4}) \wedge (\neg R_{i2} \vee \neg R_{i3}) \wedge (\neg R_{i2} \vee \neg R_{i4}) \\ & \wedge (\neg R_{i3} \vee \neg R_{i4}) \end{aligned}$$

**- Minimum une reine par colonne :**

$$\begin{aligned} & \bigwedge j \in \{1,\dots,n\} (\bigvee i \in \{1,\dots,n\} R(i,j)) \\ & (R_{11} \vee R_{21} \vee \dots \vee R_{n1}) \wedge (R_{12} \vee R_{22} \vee \dots \vee R_{n2}) \wedge \dots \wedge (R_{1n} \vee R_{2n} \vee \dots \\ & \vee R_{nn}) \end{aligned}$$

**- Maximum une reine par colonne :**

$$\begin{aligned} & (\bigwedge i \in \{1,\dots,n\} \bigwedge j \in \{1,\dots,n\} (R(j,i) \rightarrow \bigwedge k \in \{1,\dots,n\} \neg R(k,i)) = (\bigwedge i \in \{1,\dots,n\} \bigwedge j \in \{1,\dots,n\} \bigwedge k \in \{1,\dots,n\}, (j \neq k) (\neg \\ & R(j,i) \vee \neg R(k,i)) \\ & (\neg R_{1j} \vee \neg R_{2j}) \wedge (\neg R_{1j} \vee \neg R_{3j}) \wedge (\neg R_{1j} \vee \neg R_{4j}) \wedge (\neg R_{2j} \vee \neg R_{3j}) \wedge (\neg R_{2j} \vee \neg R_{4j}) \\ & \wedge (\neg R_{3j} \vee \neg R_{4j}) \end{aligned}$$

**- Maximum une reine par diagonale ascendante :**

$$(\neg R_{ij} \vee \neg R_{kl}) \text{ pour tout } i+j = k+l, 1 \leq k < i \leq n, 1 \leq j < l \leq n$$

**- Maximum une reine par diagonale descendante :**

$$(\neg R_{ij} \vee \neg R_{kl}) \text{ pour tout } i-j = k-l, 1 \leq i < k \leq n, 1 \leq j < l \leq n$$

La conjonction de toutes ces règles nous donne la forme normale conjonctive (FNC) du problème des n-reines.

### **Exemple de FNC complète pour n = 4 :**

$(R_{11} \vee R_{12} \vee R_{13} \vee R_{14}) \wedge (R_{21} \vee R_{22} \vee R_{23} \vee R_{24}) \wedge (R_{31} \vee R_{32} \vee R_{33} \vee R_{34}) \wedge (R_{41} \vee R_{42} \vee R_{43} \vee R_{44})$   
 $\wedge (R_{11} \vee R_{21} \vee R_{31} \vee R_{41}) \wedge (R_{12} \vee R_{22} \vee R_{32} \vee R_{42}) \wedge (R_{13} \vee R_{23} \vee R_{33} \vee R_{43}) \wedge (R_{14} \vee R_{24} \vee R_{34} \vee R_{44})$   
 $\wedge (\neg R_{21} \vee \neg R_{12}) \wedge (\neg R_{31} \vee \neg R_{22}) \wedge (\neg R_{31} \vee \neg R_{13}) \wedge (\neg R_{22} \vee \neg R_{13}) \wedge (\neg R_{41} \vee \neg R_{32}) \wedge (\neg R_{41} \vee \neg R_{23}) \wedge (\neg R_{41} \vee \neg R_{14}) \wedge (\neg R_{32} \vee \neg R_{23}) \wedge (\neg R_{32} \vee \neg R_{14})$   
 $\wedge (\neg R_{23} \vee \neg R_{14}) \wedge (\neg R_{42} \vee \neg R_{33}) \wedge (\neg R_{42} \vee \neg R_{24}) \wedge (\neg R_{33} \vee \neg R_{24}) \wedge (\neg R_{43} \vee \neg R_{34})$   
 $\wedge (\neg R_{31} \vee \neg R_{42}) \wedge (\neg R_{21} \vee \neg R_{32}) \wedge (\neg R_{21} \vee \neg R_{43}) \wedge (\neg R_{32} \vee \neg R_{43}) \wedge (\neg R_{11} \vee \neg R_{22}) \wedge (\neg R_{11} \vee \neg R_{33}) \wedge (\neg R_{11} \vee \neg R_{44}) \wedge (\neg R_{22} \vee \neg R_{33}) \wedge (\neg R_{22} \vee \neg R_{44})$   
 $\wedge (\neg R_{33} \vee \neg R_{44}) \wedge (\neg R_{12} \vee \neg R_{23}) \wedge (\neg R_{12} \vee \neg R_{34}) \wedge (\neg R_{23} \vee \neg R_{34}) \wedge (\neg R_{13} \vee \neg R_{24})$   
 $\wedge (\neg R_{11} \vee \neg R_{12}) \wedge (\neg R_{11} \vee \neg R_{13}) \wedge (\neg R_{11} \vee \neg R_{14}) \wedge (\neg R_{12} \vee \neg R_{13}) \wedge (\neg R_{12} \vee \neg R_{14}) \wedge (\neg R_{13} \vee \neg R_{14})$   
 $\wedge (\neg R_{21} \vee \neg R_{22}) \wedge (\neg R_{21} \vee \neg R_{23}) \wedge (\neg R_{21} \vee \neg R_{24}) \wedge (\neg R_{22} \vee \neg R_{23}) \wedge (\neg R_{22} \vee \neg R_{24}) \wedge (\neg R_{23} \vee \neg R_{24})$   
 $\wedge (\neg R_{31} \vee \neg R_{32}) \wedge (\neg R_{31} \vee \neg R_{33}) \wedge (\neg R_{31} \vee \neg R_{34}) \wedge (\neg R_{32} \vee \neg R_{33}) \wedge (\neg R_{32} \vee \neg R_{34}) \wedge (\neg R_{33} \vee \neg R_{34})$   
 $\wedge (\neg R_{41} \vee \neg R_{42}) \wedge (\neg R_{41} \vee \neg R_{43}) \wedge (\neg R_{41} \vee \neg R_{44}) \wedge (\neg R_{42} \vee \neg R_{43}) \wedge (\neg R_{42} \vee \neg R_{44}) \wedge (\neg R_{43} \vee \neg R_{44})$   
 $\wedge (\neg R_{11} \vee \neg R_{21}) \wedge (\neg R_{11} \vee \neg R_{31}) \wedge (\neg R_{11} \vee \neg R_{41}) \wedge (\neg R_{21} \vee \neg R_{31}) \wedge (\neg R_{21} \vee \neg R_{41}) \wedge (\neg R_{31} \vee \neg R_{41})$   
 $\wedge (\neg R_{12} \vee \neg R_{22}) \wedge (\neg R_{12} \vee \neg R_{32}) \wedge (\neg R_{12} \vee \neg R_{42}) \wedge (\neg R_{22} \vee \neg R_{32}) \wedge (\neg R_{22} \vee \neg R_{42}) \wedge (\neg R_{32} \vee \neg R_{42})$   
 $\wedge (\neg R_{13} \vee \neg R_{23}) \wedge (\neg R_{13} \vee \neg R_{33}) \wedge (\neg R_{13} \vee \neg R_{43}) \wedge (\neg R_{23} \vee \neg R_{33}) \wedge (\neg R_{23} \vee \neg R_{43}) \wedge (\neg R_{33} \vee \neg R_{43})$   
 $\wedge (\neg R_{14} \vee \neg R_{24}) \wedge (\neg R_{14} \vee \neg R_{34}) \wedge (\neg R_{14} \vee \neg R_{44}) \wedge (\neg R_{24} \vee \neg R_{34}) \wedge (\neg R_{24} \vee \neg R_{44}) \wedge (\neg R_{34} \vee \neg R_{44})$

## Implementation du probleme:

Nous avons produit 3 programmes:

**Dimacs.py:** un programme qui récupère un fichier txt en entrée représentant une instance de notre problème et qui crée un fichier dimacs.txt représentant cette instance.

**Affichage.py:** un programme qui récupère un fichier txt en entrée contenant le modèle sorti par un SAT-solveur et qui crée un fichier interface.txt qui représente la solution de ce problème de manière compréhensible pour l'utilisateur.

**N-reines.py:** un programme qui enchaîne les étapes des deux premiers programmes tout en faisant l'appel au SAT-solveur (PySAT) et qui affiche également la solution sous un format plus simple sur la sortie standard.

## Format des fichiers utilisés pour tous les programmes:

### **entree.txt:**

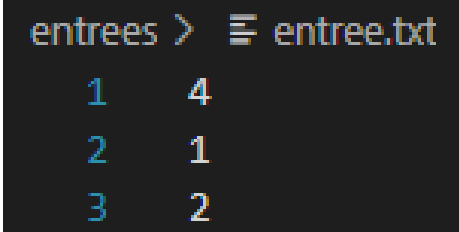
Les fichiers txt représentant une instance de notre problème sont au format suivant:

**1re ligne:** un entier correspondant à la taille de l'échiquier.

**2e ligne:** un entier compris entre 0 et la taille correspondant au nombre de reines placées dans notre instance.

**lignes qui suivent:** un entier correspondant à la position de la reine placée sur l'échiquier en entrée.

**Exemple** d'un fichier entree.txt:

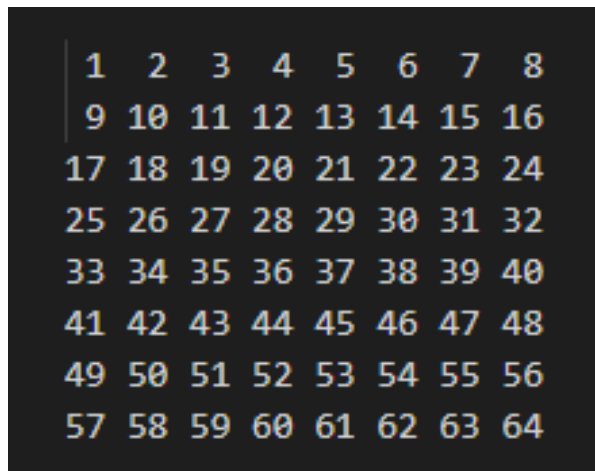


```
entrees > ≡ entree.txt
1      4
2      1
3      2
```

Celui-ci correspond à une instance avec  $n = 4$  et une seule reine placée en position 2.

## Echiquier:

Voici un exemple avec  $n = 8$  de comment les positions des reines sont numérotées pour tous nos programmes.



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  |
| 9  | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |

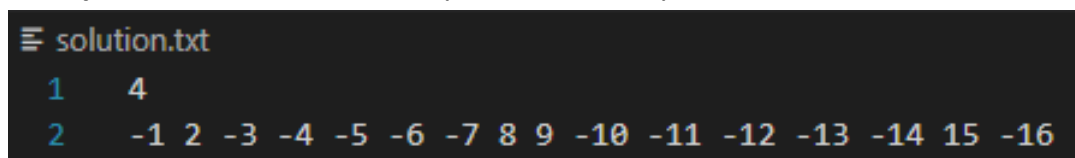
## solution.txt:

Les fichiers txt utilisés par **affichage.py** contenant la solution à afficher sont au format suivant:

**1re ligne:** un entier correspondant à la taille de l'échiquier.

**2e ligne:** une série d'entiers représentant la solution du problème.

**Exemple** du fichier solution.txt pour l'instance précédente:



```
≡ solution.txt
1      4
2     -1 2 -3 -4 -5 -6 -7 8 9 -10 -11 -12 -13 -14 15 -16
```

Un entier précédé d'un "-" signifie qu'il n'y a pas de reine sur cette case.

Un entier sans le "-" signifie qu'il y a une reine sur cette case.

## interface.txt

Un fichier txt qui affiche la solution de manière lisible pour l'utilisateur.

**Exemple** du fichier interface.txt pour l'instance précédente:

```
≡ interface.txt
1  + -- + -- + -- + -- +
2  |   | RR |   |   |
3  + -- + -- + -- + -- +
4  |   |   |   | RR |
5  + -- + -- + -- + -- +
6  | RR |   |   |   |
7  + -- + -- + -- + -- +
8  |   |   | RR |   |
9  + -- + -- + -- + -- +
10
```

## Utilisation des programmes:

### Dimacs.py

Le programme dimacs.py est lancé avec un fichier au format entree.txt comme argument:

```
\Projet INF402>python n-reines.py ./entrees/entree.txt
```

Une fois l'exécution terminée, le fichier dimacs.txt décrivant l'instance donnée est produit et peut être consulté ou donné à un SAT-solveur.

Des commentaires ont été ajoutés au milieu du fichier DIMACS pour séparer les différentes règles afin de rendre la lecture plus simple pour un utilisateur.

Nous avons choisi d'utiliser une fonction **calc\_clauses(n)** qui calcule le nombre de clauses qui vont être produites à l'avance afin de pouvoir écrire la première ligne du fichier DIMACS.

Le programme récupère également les positions des reines données en entrée afin de les ajouter au reste des clauses générées.

Pour l'écriture du reste des règles décrites dans notre modélisation, nous avons écrit des boucles qui en fonction de la taille de l'échiquier écrivent toutes les combinaisons de clauses nécessaires pour les règles suivantes:

- Minimum une reine par ligne
- Maximum une reine par ligne
- Minimum une reine par colonne
- Maximum une reine par colonne
- Maximum une reine par diagonale ascendante
- Maximum une reine par diagonale descendante

Voici un **exemple** de la boucle pour "Maximum une reine par ligne":

```
# Maximum 1 reine par ligne
dimacs.write("c maximum une reine par ligne \n")
for i in range(taille):
    for j in range(1, taille+1):
        for n in range(j+1, taille+1):
            variable1 = i*taille + j
            variable2 = i*taille + n
            dimacs.write(f"-{variable1} -{variable2} 0\n")
```

Voici le début du fichier dimacs.txt généré par le programme:

```
dimacs.txt
1  p cnf 16 85
2  c reines donnees en entree
3  2 0
4  c minimum une reine par ligne
5  1 2 3 4 0
6  5 6 7 8 0
7  9 10 11 12 0
8  13 14 15 16 0
9  c maximum une reine par ligne
10 -1 -2 0
11 -1 -3 0
12 -1 -4 0
13 -2 -3 0
14 -2 -4 0
15 -3 -4 0
16 -5 -6 0
17 -5 -7 0
18 -5 -8 0
```



## Affichage.py

Le programme `affichage.py` est lancé avec un fichier au format `solution.txt` comme argument:

```
\Projet INF402>python affichage.py solution.txt
```

Ce programme récupère simplement la taille de l'échiquier et le modèle fourni par le SAT-solveur dans le fichier `solution.txt` puis commence à écrire une grille dans le fichier `interface.txt` avec le format vu précédemment.

Lors de cette boucle, pour chaque position sur l'échiquier, le programme vérifie la liste contenant le modèle pour savoir s'il faut placer une reine ou laisser une case vide.

## N-reines.py

`N-reines.py` est le programme principal du projet qui enchaîne les étapes des deux programmes décrits précédemment ainsi que l'appel au SAT-solveur PySAT.

Ce programme est également lancé avec un fichier au format `entree.txt` comme argument:

```
\Projet INF402>python n-reines.py ./entrees/entree.txt
```

Pour ce programme, nous avons utilisé la bibliothèque Python "PySAT" qui contient le SAT-solveur Glucose3 pour résoudre notre problème des  $n$ -reines.

Notre programme commence par vérifier que la taille de l'échiquier donnée dans le fichier en entrée est une valeur pour laquelle le jeu des  $n$ -reines a une solution.

Si l'entrée contient une valeur de  $n$  pour laquelle il n'y a pas de solution, le programme affiche un message sur la sortie standard et se termine.

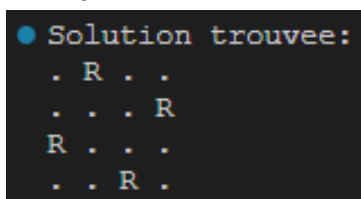
Si le  $n$  donné en entrée est valide, le programme exécute le même processus que `dimacs.py` et crée le fichier `dimacs.txt` contenant la modélisation de notre instance.

Nous avons ensuite choisi d'écrire une fonction **dimacs\_to\_cnf** pour récupérer les clauses écrites dans notre fichier dimacs.txt au début de notre programme dans un format que l'on peut passer au solveur Glucose3.

Cela nous permet de faire l'appel au SAT-solveur et si l'instance donnée est satisfaisable,

- le programme écrit le modèle dans le fichier solution.txt
- il génère le fichier interface.txt correspondant à la solution trouvée
- et il affiche un message sur la sortie standard "Solution trouvée:" suivi d'une version simplifiée de la sortie d'interface.txt

**Exemple** de la sortie standard pour notre instance:



```
● Solution trouvee:
. R . .
. . . R
R . . .
. . R .
```

Si le SAT-solveur retourne que l'instance est insatisfaisable, le programme affiche sur la sortie standard qu'il n'y a pas de solution pour cette instance et se termine.

### **Série pertinente d'instances de notre problème:**

Dans le répertoire **/entrees**, nous avons mis une série d'instances afin de pouvoir tester notre programme.

Pour différentes tailles d'échiquier, nous avons créé des instances vides afin de laisser le SAT-solveur générer une solution.

Des instances avec quelques reines placées afin de forcer une solution.

Des instances complètes satisfaisables et insatisfaisables afin de vérifier le bon fonctionnement des différentes règles.