Résolution de sudokus

1 Principe du jeu de sudoku

Le jeu prend la forme d'une grille de 9×9 cases structurées en 9 sous-grilles de 3×3 cases, appelées « régions ». Quelques cases contiennent des chiffres, elles sont dites "dévoilées". Le but du jeu est de remplir les cases vides (c'est à dire de les "dévoiler"). La contrainte à satisfaire est la suivante :

Chaque case doit contenir un chiffre compris entre 1 *et* 9 *de telle sorte que dans chaque groupement (ligne, colonne ou région) on trouve une unique occurrence de tout chiffre allant de* 1 *à* 9.

Lorsque la solution est unique, la grille prend le nom de sudoku.

Exemple :

0	8	7	0	0	0	5	2	0
9	1	0	5	0	2	0	4	6
2	0	0	0	0	0	0	0	7
0	9	0	0	2	0	0	1	0
0	0	0	1	0	6	0	0	0
0	4	0	0	9	0	0	8	0
6	0	0	0	0	0	0	0	3
5	7	0	3	0	1	0	6	8
0	3	8	0	0	0	9	5	0

Les cases non dévoilées sont à 0.

Vocabulaire : Les groupements sont les lignes, colonnes et régions. Ils sont numérotés de 0 à 8. La numérotation des régions est :

0	1	2
3	4	5
6	7	8

On peut :

— pour une région donnée, exprimer les coordonnées de la case située en haut et à gauche de cette région :

$$lig = 3 * (region/3) \qquad col = 3 * (region\%3)$$

— pour une case [lig][col], exprimer le numéro de sa région : $3 * (lig/3) + (col/3)$
— pour une case [lig][col], exprimer son numéro (compris entre 0 et 80) : $9 * lig + col$
— pour une case de numéro c (compris entre 0 et 80), exprimer son indice de ligne : $c/9$ et de colonne : $c\%9$

2 Méthode de résolution : implémentation de diverses règles

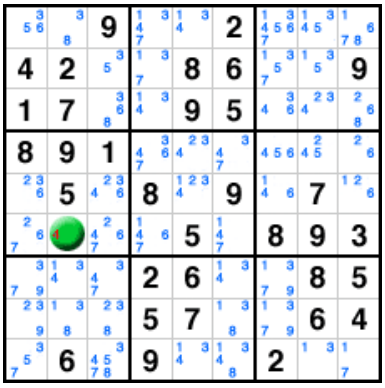
Pour chaque case non dévoilée les candidats sont les chiffres $1, 2, 3, 4, 5, 6, 7, 8, 9$. En tout, il y a donc : $9^{\text{nombre de cases non dévoilées}}$ grilles candidates, nombre trop important pour pouvoir faire une recherche exhaustive. Afin de diminuer ce nombre on applique les règles détaillées ci-dessous. Les divers principes déductifs utilisés dans la résolution sont décrits sur le site "http ://www.mots-croises.ch/Manuels/Sudoku/"

Alors que certaines règles permettent une détermination directe du contenu d'une case non dévoilée (R1 candidats seuls, R3 candidats uniques), d'autres permettent seulement de diminuer le nombre de candidats de certaines cases (R2, R4 considérations interrégions, R5 X-wing).

— **R1 Candidats seuls** : S'il n'y a qu'un candidat à une case, alors il doit être placé dans la case.

— **R2 Une seule occurrence de chaque chiffre dans chaque groupement** : Un candidat à une case ne peut être qu'un chiffre non encore placé dans un groupement (ligne, colonne et région) auquel appartient cette case.

Ainsi, dans la grille ci-contre, la liste des candidats de la case 0 ne peut contenir :
· ni 9 ni 2 : déjà placés sur la ligne 0
· ni 4 ni 1 ni 8 : déjà placés dans la colonne 0
· ni 2 ni 7 : déjà plaçés dans la région 0
il ne reste donc plus que 3 candidats : 3 5 et 6 pour cette case aprés l'application de R2.



L'application de ces deux seules règles est suffisante pour résoudre des sudokus faciles (cf exemple 1) mais, en général, elles ne suffisent pas. Dans l'exemple 2 leur utilisation permet seulement de diminuer à 3.510^{26} le nombre de grilles candidates. Il est donc nécessaire de faire intervenir d'autres règles.

— **R3 Candidats uniques** : Un candidat n'apparaissant que pour une seule case d'un groupement peut être placé dans la case correspondante.
Exemple d'application pour une ligne :

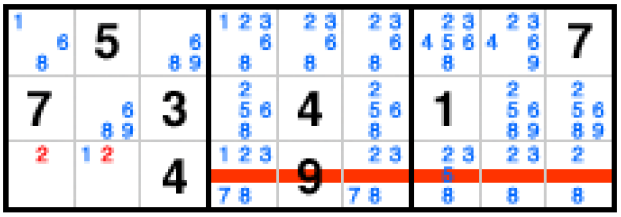
Avant application de la règle :

Après application de la règle :

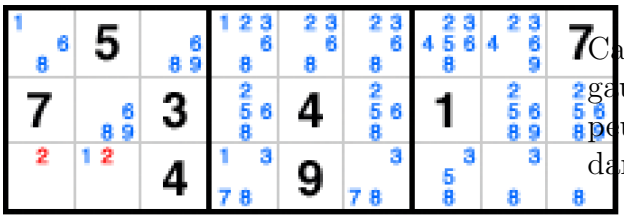
Dans l'exemple 2 l'application de R3 permet de diminuer à 1.1210^{15} le nombre de grilles candidates ce qui n'est pas encore assez faible pour envisager une recherche exhaustive.

— **R4 Considération interrégion** : dans une région donnée, lorsqu'un candidat n'est présent que dans des cases d'une seule ligne, on peut supprimer ce candidat des autres cases de cette ligne (c'est à dire des cases de la même ligne appartenant aux 2 autres régions ▬) :

Avant application de la règle :



Après application :



Car dans la région de gauche, le chiffre 2 ne peut apparaitre que dans la ligne du bas.

Cette règle s'applique de la même manière en colonne.

L'application des 4 règles permet de résoudre entièrement la grille de l'exemple 2.

— **R5 X-Wing** : Cette règle s'applique si la grille contient une configuration en X c'est à dire :

- s'il existe 2 groupements (par exemple 2 lignes) qui contiennent chacune exactement 2 cases pour lesquelles c est un candidat.
- et si ces 4 cases appartiennent aussi à 2 autres groupements (par exemple 2 colonnes).

Les 4 cases occupent alors les sommets d'un trapèze (configuration en X) . On peut alors supprimer le candidat c de toutes les cases de ces 2 colonnes qui ne sont pas dans les 2 lignes de départ.

Six configurations sont ainsi possibles :

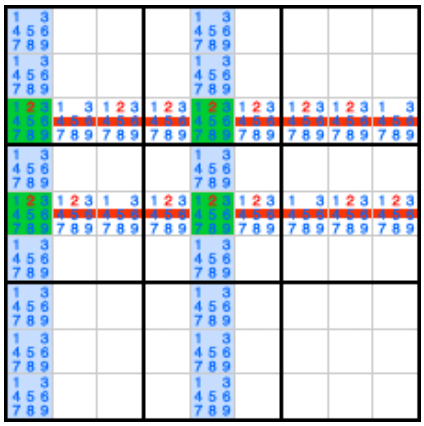
— lignes avec colonnes : entraînant la suppression de candidats dans les cases des 2 colonnes qui ne sont pas sur les 2 lignes.

— lignes avec régions : entraînant la suppression de candidats dans les cases des 2 régions qui ne sont pas sur les 2 lignes.
— colonnes avec lignes :
— colonnes avec régions : etc
— régions avec lignes :
— régions avec colonnes :

Exemples :

- colonnes avec lignes :

  Dans chacune des deux colonnes bleues, on trouve exactement deux cases pour lesquelles 2 est candidat deux fois . Ces 4 cases vertes appartiennent à deux lignes et sont en configuration X. Nécessairement la valeur 2 doit être placée aux extrémités de la même barre du X (sinon il y aurait deux 2 dans une même ligne ou dans une même colonne). Dans les 2 cas, on est sûr de pouvoir enlever le candidat 2 de toutes les cases de ces 2 lignes qui n'appartiennent pas aux 2 colonnes.

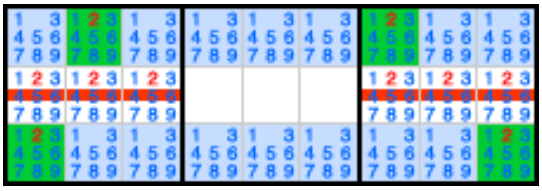


- régions avec lignes :

  Dans chacune des deux régions bleues, on trouve exactement deux cases pour lesquelles 2 est candidat deux fois. Ces 4 cases vertes appartiennent à 2 lignes et sont en configuration X. Nécessairement la valeur 2 doit être placée aux extrémités de la même barre du X (diagonale du trapèze) sinon il y aurait deux 2 dans une même ligne. Dans les 2 cas, on peut enlever le candidat 2 de toutes les cases de ces 2 lignes qui n'appartiennent pas aux 2 régions.

- lignes avec régions :

  Dans chacune des deux lignes bleues, on trouve exactement deux cases pour lesquelles 2 est candidat dzeux fois. Ces 4 cases vertes appartiennent à 2 régions et sont en configuration X. Nécessairement la valeur 2 doit être placée aux extrémités de la même barre du X (diagonale du trapèze) sinon il y aurait deux 2 dans une même ligne. Dans les 2 cas, on peut enlever le candidat 2 de toutes les cases de ces 2 régions qui n'appartiennent pas aux 2 lignes.



Remarques:

1. Certaines grilles ne peuvent être résolues par l'application des règles précédentes, mais le nombre de grilles candidates devient (quelquefois cf exemple 5) assez faible pour permettre une recherche exhaustive.
2. Attention, l'application d'une règle peut permettre de réappliquer les règles précédentes ! C'est seulement lorsque les tableaux de candidats n'évoluent plus qu'on a épuisé les règles utilisées. Il faut donc appeler les règles dans une boucle dont on sort lorsque l'applicatiuon de ces règles laisse le système inchangé.

3 Travail à réaliser, notation

On veut écrire un programme de résolution d'un sudoku : lorsqu'on lui fournit une grille initiale de sudoku, le programme cherche la solution et l'affiche à l'écran.

3.1 Contraintes d'implantation :

Une case sera inplémentée par le type T_case suivant :

```
typedef struct
{
    int val ;              /* Valeur de la case (0 si vide)*/
    int n_candidats ; /* Nombre de candidats possibles */
    int candidats[9] ; /* Liste de candidats */
} T_case ;
```

Pour des considérations d'optimisation de la mémoire, une grille de sudoku sera implémentée par le type T_sudoku suivant :

```
typedef struct
{
    T_case grille[81] ; /*vecteur de 81 cases*/
}
T_sudoku ;
```

A partir d'une case d'indice c (entier compris entre 0 et 80) on peut déterminer facilement l'indice de ligne ou de colonne associé. (voir page 1)

Vous devez obligatoirement utiliser les structures de données précédentes lors de votre codage.

3.2 Notation

La notation se fait en deux étapes :

- Etape 1 : Catégories de programmes
  Le programme que vous allez rendre sera classé dans l'une des catégories ci-dessous et sera noté par rapport à la note maximale associée sur un total de 14 points.

  — **V1 : Version basique : 10 points maximum**

  · sous-programmes de saisie (au clavier) et d'affichage de grilles,
  · sous programmes de correspondance entre une case du tableau et les ligne, colonne, région. **Ces sous programmes devront être spécifiés et des "assert" doivent contrôler les données d'entrée**.
  · sous-programmes destinés à la gestion des candidats lors de l'implémentation des diverses règles : étant donné un tableau t constitué de n entiers différents,
  — écrire un sous-programme de recherche d'une valeur v dans le tableau t ayant une dimension n : si la valeur v est présente dans t, le résultat est l'indice de la case sinon le résultat est n. Ce sous-programme sera utilisé pour savoir si une valeur fait partie des candidats à une case.

— écrire un sous-programme de suppression d'une valeur v supposée présente dans le tableau t. Ce sous-programme sera utilisé pour supprimer un candidat de la liste des candidats à une case.
— **Ces deux derniers sous-programmes doivent être spécifiés (Pré-condition, Post-condition), et l'invariant de boucle écrit en commentaire dans le programme.**

· Implémentation des règles R1 et R2 (Exemple 1),
· L'utilisateur peut choisir un mode "pas à pas" : dans ce cas, on affiche la grille après chaque affectation d'une case et le programme attend l'appui d'une touche pour continuer.

— **V2 : Version intermédiaire : 12 points maximum**
La version V1 étant déjà implémentée :
· Ajout de l'implémentation des règles R3 et R4 (Exemples 2 et 3).
· Possibilité d'utilisation de fichiers texte associés à une grille : une grille peut être lue à partir d'un fichier texte. Chaque fichier texte contient une succession de 81 caractères (chiffres allant de 0 à 9) séparés par un caractère de contrôle (tabulation, espace, retour à la ligne). Le nom du fichier texte sera passés en argument de la ligne de commande. Si le fichier est absent on lit la grille au clavier.

— **V3 : Version «top black belt» : 14 points maximum**

— Implémentation de la règle R5 (Exemple 4)
— Recherche exhaustive (ou force brute) : on essaie de manière systématique toutes les grilles encore candidates jusqu'à trouver la bonne. Cette stratégie n'est applicable que si le nombre de grilles candidates est raisonnable (Exemple 5). L'algorithme demandé est itératif.

Attention : Pour obtenir les points relatifs à une version, il faut que les versions inférieures aient été traitées dans leur intégralité. (par exemple, une résolution exhaustive n'apportera aucun point si les règles R1,R2,R3 et R4 n'ont pas été implémentées etc.)

- Etape 2 : Détermination de la note finale
Les trois principaux points qui guideront la notation dans chaque catégorie sont :

— Point 1 : 60% de la note finale
compilation, execution : le programme compile et s'exécute sans bugs.

— Point 2 : 25% de la note finale
qualité du code :
· pertinence des type utilisés,
· efficacité des algorithmes,
· simplicité, efficacité du code C,
· lisibilité du code : indentation, choix du nom des identificateurs,
· respect des règles de codage,
· commentaires

— Point 3 : 15% de la note finale

  · modularité : pertinence du découpage en modules
  · pertinence des .h
  · compilation séparée, makefile : automatisation de la compilation.

4 Exemples :

4.1 Exemple 1 : R1+R2

0	8	7	0	0	0	5	2	0
9	1	0	5	0	2	0	4	6
2	0	0	0	0	0	0	0	7
0	9	0	0	2	0	0	1	0
0	0	0	1	0	6	0	0	0
0	4	0	0	9	0	0	8	0
6	0	0	0	0	0	0	0	3
5	7	0	3	0	1	0	6	8
0	3	8	0	0	0	9	5	0

Après application des règles R1 et R2, le sudoku est résolu.

4	8	7	6	3	9	5	2	1
9	1	3	5	7	2	8	4	6
2	6	5	8	1	4	3	9	7
8	9	6	4	2	3	7	1	5
7	5	2	1	8	6	4	3	9
3	4	1	7	9	5	6	8	2
6	2	4	9	5	8	1	7	3
5	7	9	3	4	1	2	6	8
1	3	8	2	6	7	9	5	4

4.2 Exemple 2 : R1+R2+R3

0	0	0	0	7	0	0	0	5
2	0	7	1	0	0	4	0	0
0	6	0	4	0	0	1	0	7
5	0	6	7	0	0	2	0	0
0	0	0	0	9	0	0	0	4
0	9	0	5	0	0	3	0	0
0	0	0	0	0	0	0	0	8
3	0	5	0	1	8	0	6	2
9	0	8	0	4	2	0	1	3

Après application des règles R1 et R2 :
12 cases dévoilées.

0	0	0	0	7	0	0	0	5
2	0	7	1	0	0	4	0	0
8	6	0	4	0	0	1	0	7
5	0	6	7	0	0	2	0	0
0	0	0	0	9	0	0	0	4
0	9	0	5	0	0	3	0	0
0	0	0	3	5	7	9	4	8
3	4	5	9	1	8	7	6	2
9	7	8	6	4	2	5	1	3

Après application des règles R1 R2 et R3 :
sudoku résolu

1	3	4	8	7	9	6	2	5
2	5	7	1	6	3	4	8	9
8	6	9	4	2	5	1	3	7
5	8	6	7	3	4	2	9	1
7	1	3	2	9	6	8	5	4
4	9	2	5	8	1	3	7	6
6	2	1	3	5	7	9	4	8
3	4	5	9	1	8	7	6	2
9	7	8	6	4	2	5	1	3

4.3 Exemple 3 : R1+R2+R3+R4

0	0	0	5	0	0	0	1	0
6	0	5	0	2	0	0	0	0
0	0	1	8	0	0	5	0	3
0	0	4	0	6	0	0	0	1
7	0	0	0	0	0	0	0	4
8	0	0	0	3	0	6	0	0
2	0	9	0	0	5	7	0	0
0	0	0	0	8	0	2	0	5
0	6	0	0	0	9	0	0	0

Après application des règles R1 et R2 :
3 cases dévoilées.

0	0	0	5	0	0	0	1	0
6	0	5	0	2	0	0	0	0
0	0	1	8	0	0	5	0	3
0	0	4	0	6	0	0	0	1
7	0	0	0	0	0	0	0	4
8	0	2	0	3	0	6	0	0
2	0	9	0	0	5	7	0	6
0	0	0	0	8	0	2	0	5
0	6	0	0	0	9	0	0	8

Après application des règles R1 R2 et R3 :
16 cases supplémentaires dévoilées.

0	0	8	5	0	6	0	1	2
6	0	5	0	2	0	0	0	0
0	2	1	8	0	0	5	6	3
0	0	4	0	6	0	0	0	1
7	0	6	0	5	0	0	0	4
8	0	2	0	3	0	6	0	0
2	8	9	0	1	5	7	0	6
1	4	0	6	8	0	2	9	5
5	6	0	2	0	9	1	0	8

Après application des règles R1 R2 R3 et R4 :
le sudoku est résolu.

3	7	8	5	4	6	9	1	2
6	9	5	3	2	1	4	8	7
4	2	1	8	9	7	5	6	3
9	3	4	7	6	2	8	5	1
7	1	6	9	5	8	3	2	4
8	5	2	1	3	4	6	7	9
2	8	9	4	1	5	7	3	6
1	4	7	6	8	3	2	9	5
5	6	3	2	7	9	1	4	8

4.4 Exemple 4 : R1+R2+R3+R4+R5

0	0	3	0	0	0	8	0	0
7	0	0	3	0	6	0	0	2
0	0	5	4	0	8	3	0	0
0	3	8	1	0	4	2	5	0
0	0	0	0	0	0	0	0	0
0	1	7	5	0	3	9	8	0
0	0	2	6	0	7	4	0	0
8	0	0	2	0	1	0	0	9
0	0	1	0	0	0	7	0	0

Après application des règles R1 R2 R3 et R4 :
3 cases dévoilées

0	0	3	0	0	0	8	0	0
7	8	0	3	0	6	0	0	2
0	0	5	4	0	8	3	0	0
0	3	8	1	0	4	2	5	0
0	0	0	0	0	0	0	0	0
0	1	7	5	0	3	9	8	0
0	0	2	6	0	7	4	0	0
8	7	0	2	0	1	0	0	9
0	0	1	0	0	0	7	2	0

Aprés application de R1 R2 R3 R4 et R5 :
le sudoku est résolu.

6	4	3	7	2	5	8	9	1
7	8	9	3	1	6	5	4	2
1	2	5	4	9	8	3	6	7
9	3	8	1	7	4	2	5	6
4	5	6	9	8	2	1	7	3
2	1	7	5	6	3	9	8	4
5	9	2	6	3	7	4	1	8
8	7	4	2	5	1	6	3	9
3	6	1	8	4	9	7	2	5

4.5 Exemple 5 : R1+R2+R3+R4+R5+ Recherche exhaustive

1	0	3	6	7	8	0	0	2
0	4	0	0	0	0	9	0	0
0	0	0	0	9	0	0	0	0
4	0	7	0	0	0	0	1	0
0	5	8	0	0	0	2	6	0
0	3	0	0	0	0	8	0	4
0	0	0	0	3	0	0	0	0
0	0	9	7	0	0	0	2	0
7	0	0	5	4	9	0	0	6

Après application des règles R1 R2 R3 et R4 :
40 cases dévoilées

1	9	3	6	7	8	5	4	2
8	4	6	0	5	0	9	7	0
2	7	5	0	9	0	6	8	0
4	2	7	8	6	5	3	1	9
9	5	8	0	1	0	2	6	7
6	3	1	9	2	7	8	5	4
5	0	4	0	3	0	7	9	8
3	0	9	7	8	0	4	2	5
7	8	2	5	4	9	1	3	6

Après application des règles R1 R2 R3 R4 et R5 :
3 cases supplémentaires dévoilées

1	9	3	6	7	8	5	4	2
8	4	6	0	5	0	9	7	0
2	7	5	0	9	0	6	8	0
4	2	7	8	6	5	3	1	9
9	5	8	0	1	0	2	6	7
6	3	1	9	2	7	8	5	4
5	6	4	0	3	0	7	9	8
3	1	9	7	8	6	4	2	5
7	8	2	5	4	9	1	3	6

Il reste 62208 grilles à examiner : on peut lancer une recherche exhaustive, on obtient la solution :

1	9	3	6	7	8	5	4	2
8	4	6	3	5	2	9	7	1
2	7	5	1	9	4	6	8	3
4	2	7	8	6	5	3	1	9
9	5	8	4	1	3	2	6	7
6	3	1	9	2	7	8	5	4
5	6	4	2	3	1	7	9	8
3	1	9	7	8	6	4	2	5
7	8	2	5	4	9	1	3	6

4.6 Exemple 6 : R1+R2+R3+R4+R5+ ...

0	0	1	0	0	8	4	0	0
6	3	0	0	0	0	0	7	8
0	7	5	0	0	0	2	0	9
0	0	0	0	0	0	0	0	0
0	8	0	3	5	0	0	0	4
0	2	0	6	9	0	0	0	5
0	5	2	0	0	0	3	0	1
0	0	8	0	0	7	6	0	0
1	6	0	0	0	0	0	4	7

Après application des règles R1 R2 R3 et R4 :
21 cases dévoilées

2	9	1	0	0	8	4	0	0
6	3	4	0	2	0	0	7	8
8	7	5	0	0	0	2	0	9
5	1	0	0	0	4	9	2	0
9	8	0	3	5	2	0	0	4
4	2	0	6	9	1	0	0	5
7	5	2	0	0	0	3	0	1
3	4	8	0	1	7	6	0	2
1	6	9	2	0	0	0	4	7

Aprés application de R1 R2 R3 R4 et R5 :
il reste 10^{10} grilles candidates.

2	9	1	5	7	8	4	3	6
6	3	4	1	2	9	5	7	8
8	7	5	4	6	3	2	1	9
5	1	6	7	8	4	9	2	3
9	8	7	3	5	2	1	6	4
4	2	3	6	9	1	7	8	5
7	5	2	8	4	6	3	9	1
3	4	8	9	1	7	6	5	2
1	6	9	2	3	5	8	4	7