SUDOKU

1 Le jeu

Le sudoku (du japonais *sû*, chiffre, et de *doku*, unique) est un jeu en forme de grille correspondant le plus souvent à un tableau carré de neuf cases de côté. Un grille est ainsi composée de neuf lignes, de neuf colonnes, et de neuf carrés de trois cases de côté appelés région (voir les numérotations en figure 1 ci-dessous). Le but du jeu est de remplir cette grille avec des chiffres allant de 1 à 9 de sorte que, chaque ligne, chaque colonne et chaque région, ne contienne qu'une et une seule fois tous les chiffres de 1 à 9. Notons qu'il existe environ 6.67×10^{21} grilles différentes que l'on peut réduire à un peu moins de six milliards du fait de permutations. Un jeu consiste en une grille partiellement remplie qu'il s'agit de compléter en respectant les contraintes énoncées ci-dessus. Les chiffres ne sont utilisés que par convention et n'importe quel ensemble de 9 signes distincts (comme par exemple des lettres ou des couleurs) peut être utilisé sans changer les règles du jeu.

	0	1	2	3	4	5	6	7	8
0		I			II	1		8	
1	2		4		7			3	5
2		7			5	6	2	1	
3		6		2	III	5		IV	1
4	3								6
5	1			6		7		2	
6		3	1	8	6			7	
7	7	2			1		8		4
8		5		7	V	VI		VII	

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

FIGURE 1 – Un jeu et sa solution

Dans la presse, on trouve des grilles classées selon leur degré de difficulté et n'admettant qu'une et une seule solution. On considère ici trois niveaux de difficulté dont le premier est le niveau *facile*. On dira qu'une grille est facile si elle peut être résolue à l'aide de la méthode dite *des candidats uniques* détaillée au paragraphe 3. Toute grille ne pouvant être résolue à l'aide de cette méthode sera dite de niveau *difficile*, si elle admet une unique solution, ou de niveau *terrible* si elle admet plusieurs solutions. Dans le paragraphe 4, on présente une méthode de *backtracking* permettant de résoudre n'importe quelle grille et en particulier les grilles difficiles et terribles. Le travail demandé est de programmer ces deux méthodes afin de trouver la solution à n'importe quel jeu mais aussi d'en identifier le niveau de difficulté. La qualité de la programmation et, par conséquent, le temps d'exécution (*CPU user time*) pour résoudre un jeu sera l'un des critères de notation (mais pas le seul) de ce projet.

2 Définitions

2.1 Cases ouvertes et cases fermées

On appelle *case fermée* toute case de la grille contenant un chiffre et *case ouverte* toute case vide de la grille. Le but du jeu est donc de fermer toutes les cases ouvertes d'une grille en respectant, bien entendu, les contraintes.

2.2 Cases voisines

Deux cases sont dites *voisines* si elles se trouvent sur une même ligne ou sur une même colonne ou dans une même région. La règle du jeu exige que deux cases voisines doivent contenir des chiffres différents.

2.3 Candidats

On appelle *candidat* d'une case ouverte, tout chiffre pouvant être utilisé pour remplir cette case. En figure 2 ci-dessous, on donne les candidats de chaque case ouverte de la grille de la figure 1. Par exemple, la case $(0, 0)$ admet les candidats 5, 6 et 9, alors que la case $(0, 1)$ n'admet qu'un seul candidat, le chiffre 9.

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

FIGURE 2 – Les candidats des cases ouvertes

3 Méthode des candidats uniques

3.1 Candidat unique

Considérons la case (i, j) où i est le numéro de ligne de la grille et j le numéro de colonne. On dira qu'un chiffre c est un *candidat unique* de la case (i, j) si et seulement si une au moins des quatre conditions suivantes est vérifiée :

(i) Le chiffre c est l'unique candidat de la case (i, j).

(ii) Le chiffre c est l'un des candidats de la case (i, j) mais n'est pas candidat dans une autre case de la ligne i.

(iii) Le chiffre c est l'un des candidats de la case (i, j) mais n'est pas candidat dans une autre case de la colonne j.

(iv) Le chiffre c est l'un des candidats de la case (i, j) mais n'est pas candidat dans une autre case de la région où se trouve la case (i, j).

Considérons les candidats représentés en figure 2. On constate, par exemple, que :

- Les cases $(0, 1)$, $(1, 3)$ et $(2, 8)$ ne possèdent chacune qu'un et un seul candidat (le 9 à chaque fois) qui, de fait, est un candidat unique (condition (i)).
- Le chiffre 6 est un candidat unique de la case $(1, 6)$ car 6 n'est candidat dans aucune autre case de le ligne 1 (condition (ii)).
- Le chiffre 5 est un candidat unique de la case $(0, 0)$ car 5 n'est candidat dans aucune autre case de la colonne 0 (la condition (iii)).
- Le chiffre 4 est un candidat unique de la case $(0, 6)$ car 4 n'est candidat dans aucune autre case de la région II (condition (iv)).
- Le chiffre 1 est un candidat unique de la case $(1, 1)$ qui vérifie les conditions, (ii), (iii) et (iv).

3.2 Méthode de résolution

Si une case ouverte (i, j) admet un candidat unique, elle peut être fermée : on peut remplir cette case avec ce candidat unique. On peut ensuite ôter ce candidat de l'ensemble des candidats de chaque voisine encore ouverte de la case (i, j).

Considérons, par exemple, le candidat unique 9 de la case $(0, 1)$. Si on ferme cette case avec le chiffre 9, le candidat 9 devra ensuite être éliminé des ensembles de candidats des cases ouvertes de la ligne 0, des cases ouvertes de la colonne 1 et des cases ouvertes de la région 0. Cette élimination fait alors apparaître deux nouveaux candidats uniques : le 7 en $(0, 8)$ et le 8 en $(2, 0)$.

Une méthode de résolution consiste donc à fermer successivement toutes les cases ouvertes admettant un candidat unique. Les grilles les plus simples peuvent être résolues de cette façon, c'est le cas de la grille de la figure 2 sur laquelle on pourra s'exercer.

Si une grille n'est pas de niveau facile, cette méthode permettra seulement d'éliminer tous les candidats uniques et de diminuer ainsi le nombre de cases ouvertes. On appliquera ensuite la méthode de résolution générale utilisant un algorithme de backtracking.

4 Le backtracking

4.1 Le principe

L'algorithme de backtracking (*retour sur trace*) est un algorithme qui permet de tester toutes les combinaisons possibles de candidats parmi lesquelles se trouvent la ou les solutions du jeu :

- Partant d'une première case ouverte de la grille, on la remplit avec l'un de ses candidats (le plus petit par exemple). Ensuite, on remplit successivement, et tant que cela est possible, les autres cases ouvertes de la grille en s'assurant à chaque fois que les contraintes du jeu soient respectées.

- Lorsqu'on arrive sur une case ouverte et que tous ses candidats ont été testés sans succès, on revient vers la case précédemment fermée pour choisir un candidat non testé avant de repartir de nouveau vers l'avant. Il est parfois nécessaire de faire *un retour sur trace* de plusieurs cases avant de pouvoir repartir de l'avant.

Le backtracking est donc une stratégie récursive qui permet de tester tous les remplissages possibles d'une grille et cela de manière systématique. Cette méthode permet de trouver la ou les solutions d'un jeu si elles existent. Si aucune solution n'est trouvée, c'est que la grille n'en admet pas.

4.2 Exemple

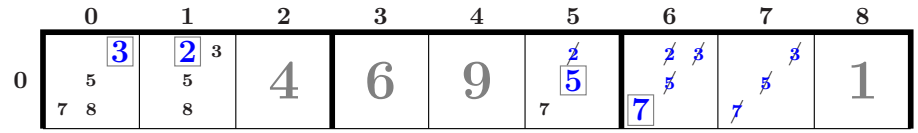
Considérons la grille ci-dessous qui ne contient aucun candidat unique. Nous allons appliquer l'algorithme de backtracking en commençant par la case ouverte $(0, 0)$ et en avançant ligne par ligne. On convient également de tester les candidats dans l'ordre croissant.

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

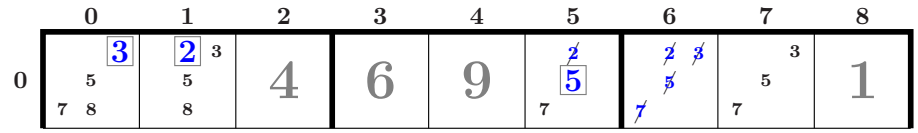
FIGURE 3 – Une grille sans candidat unique

Les premières étapes de l'algorithme sont les suivantes :

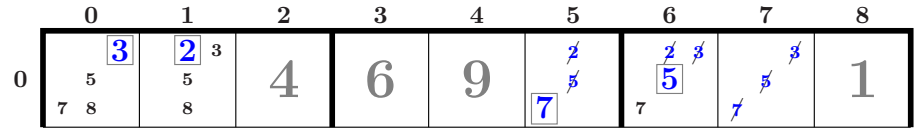
▷ On remplit successivement la case $(0,0)$ avec le candidat 3, la case $(0,1)$ avec le candidat 2, la case $(0,5)$ avec le candidat 5 (puisque 2 ne convient pas) et la case $(0,6)$ avec le candidat 7 (puisque 2, 3 et 5 ne conviennent pas). Arrivé en $(0,8)$, aucun candidat ne convient : blocage.



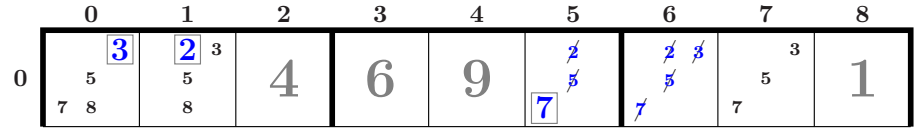
▷ On revient en $(0,6)$ où tous les candidats ont été testés : blocage.



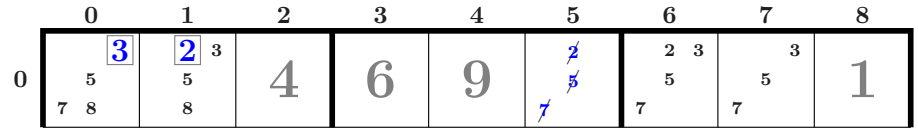
▷ On revient en $(0,5)$ pour passer au candidat 7 et on repart vers la ces $(0,6)$ que l'on remplit avec le candidat 5 (2 et 3 ne conviennent pas). Arrivé en $(0,7)$, aucun candidat ne convient : blocage.



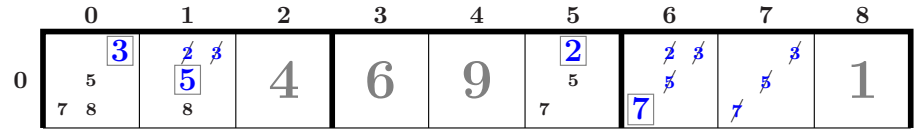
▷ On revient en $(0,6)$ où le candidat restant, le chiffre 7, ne convient pas : blocage.



▷ On revient en $(0,5)$ où tous les candidats ont été testés : blocage.



▷ On revient en $(0,1)$ où le prochain candidat à tester est le chiffre 5. On avance ensuite en remplissant successivement la case $(0,5)$ avec le candidat 2 puis la case $(0,6)$ avec le candidat 7. Arrivé en $(0,7)$, aucun candidat ne convient : blocage.



L'algorithme se poursuit ensuite jusqu'à ce que toutes les combinaisons possibles de candidats aient été testées. Notons qu'il reste à détecter les instants où, pendant le déroulement de l'algorithme, la grille est entièrement remplie.

5 Programmation

Les types, variables et fonctions présentés dans ce paragraphe devront être IMPÉRATIVEMENT utilisés dans votre programme.

5.1 Idée générale

Le programme à réaliser exécutera d'abord la méthode des candidats uniques. Si celle-ci se termine avec succès, le programme sera arrêté. Dans le cas contraire, le programme se poursuivra en exécutant la méthode de backtracking. Trois issues sont alors possibles : soit la grille admet une unique solution, soit elle admet plusieurs solutions, soit elle n'admet aucune solution. Les résultats fournis par le programme seront : une solution (si elle existe) et le niveau du jeu, le nombre total de solutions.

Il n'est pas demandé de réaliser d'interface graphique (mais vous êtes libres de le faire si vous le souhaitez et en avez le temps). Les affichages se feront en mode texte dans une fenêtre terminal.

5.2 Les grilles

Un grille à résoudre sera enregistrée dans un fichier texte selon un format bien précis. Chaque ligne de ce fichier sera composée de trois entiers : le premier est un numéro de ligne i (compris entre 0 et 8), le deuxième est un numéro de colonne j (compris entre 0 et 8) et le troisième est le chiffre contenu dans la case (i, j). Par exemple, les cinq premières lignes du fichier contenant la grille de la figure 3 sont les suivantes :

```
0 2 3
0 3 6
0 4 9
0 8 1
1 0 1
etc.
```

Vous trouverez dans le répertoire `COMMUN\franc\C\ProjetC` un ensemble de grilles à tester.

5.3 Les principaux types à définir

- `L_Candidats` : liste d'entiers pour représenter les ensembles de candidats d'une case.
- `T_Case` : une structure à deux champs, le numéro de ligne et le numéro de colonne d'une case.
- `L_Cases` : liste circulaire dont les éléments sont de type `T_Case` pour représenter l'ensemble des cases ouvertes.

5.4 Les principales variables à manipuler

- `int Grille[9][9]` : un tableau à deux dimensions pour représenter une grille. Un élément (i, j) de ce tableau contiendra soit un candidat si la case est fermée, soit 0 si la case est ouverte.
- `L_Candidats LC[9][9]` : un tableau à deux dimensions de listes de candidats. Un élément (i, j) de ce tableau est soit `NULL` si la case (i, j) est fermée, soit un pointeur sur la liste des candidats de la case (i, j) si celle-ci est ouverte.
- `L_Case LO` : la liste circulaire des cases ouvertes.

5.5 Les principales fonctions

Le problème devra être décomposé en taches et chaque tache donnera lieu à une fonction. Parmi les fonctions à écrire on devra trouver :

- ▷ `Lire_Grille` : fonction qui lit une grille dans un fichier texte et initialise le tableau `Grille`.
- ▷ `Affiche_Grille` : une fonction qui affiche une grille à l'écran.
- ▷ `Est_Candidat` : une fonction qui indique si oui ou non un chiffre donné est candidat pour une case donnée.
- ▷ `Init_Data` : une fonction qui permet d'initialiser le tableau `LC` des listes de candidats ainsi que la liste circulaire de cases ouvertes `LO`.
- ▷ `Admet_Unique` : une fonction qui retourne le candidat unique d'une case donnée si il existe ou la valeur 0 sinon.
- ▷ `Fermer_Case` : une fonction qui attribue définitivement un chiffre à une case de la grille et élimine ce chiffre des listes de candidats des cases voisines.
- ▷ `Fermer_Grille` : une fonction qui réalise le remplissage d'une grille par la méthode des candidats uniques.
- ▷ `Backtrack` : une fonction récursive qui réalise le backtracking.

Sans oublier toutes les fonctions permettant de manipuler les listes utilisées dans ce programme.

5.6 Et pour finir...

Ce projet est à réaliser par groupe de **4 étudiants maximum**. On attend de vous que le programme soit décomposé en plusieurs fichiers sources **judicieusement définis** et **clairement commentés**. Lors de la soutenance **en salle machine**, l'ensemble de vos fichiers sources sera tout d'abord **compilé** à l'aide de l'utilitaire `make` : un fichier `makefile` **devra donc être écrit**. Et c'est seulement **après** cette phase de compilation que l'exécutable sera testé.

A l'issue de la soutenance, vous devrez m'envoyer par mail l'ensemble de vos fichiers sources ansi que le fichier `makefile` permettant de les compiler (`franc.marchetti@univ-lorraine.fr`).

Des grilles pour tester votre projet

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

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

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

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

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

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

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

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

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