

TP1 : Aide à la décision

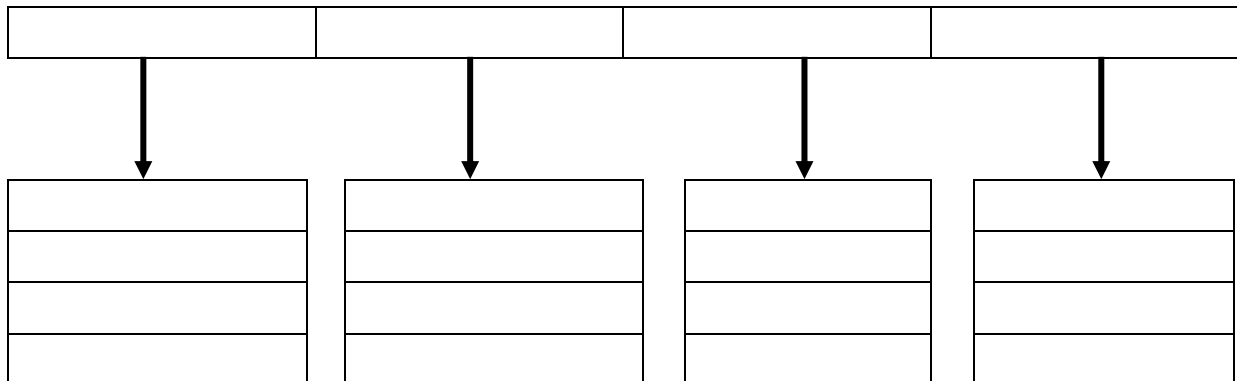
Rapport TP1

CHEUCLE Pierre – HUYNH Pascal – ODIER Valentin – WAKIM Marie

03/04/2012

Structure

La structure du sudoku sera la suivante : un vector de vector d'entiers. (Vector<Vector<int>>).



Nous trouverons donc dans ces cases les états du sudoku c'est-à-dire 1, 2, 3 ou 4 pour un sudoku 2X2. Le schéma ci-dessus correspond à un sudoku 2X2, pour un 3X3 nous aurons 5 cases supplémentaires en lignes et en colonnes. (soient 9 lignes et 9 colonnes en tout)

Cette structure sera initialisée à l'exécution du programme.

Nous ne créons pas réellement l'arbre, nous faisons un parcours virtuel de celui-ci grâce à notre fonction récursive de résolution (expliquée ci-après). Cette méthode nous permet de gagner en performance mémoire par rapport à un arbre où chaque nœud stockerait la grille du sudoku.

Algorithme de résolution

L'utilisation d'une fonction récursive nous a paru le meilleur moyen de remplir la grille de sudoku, tant au point de vue performance que temporel.

A partir de la grille nous prenons la première case (celle en haut à gauche).

Début de l'algorithme :

```
Fonction resolution (case, ligne, colonne, carré){
    Si pas de case suivante //on est à la fin du sudoku
        Retourner vrai
```

```
Si case a déjà une valeur fixée
    resolution(case suivante, ligne, colonne, carré)
Sinon
    Etudier les possibilités avec les contraintes de
    ligne, de colonne et de carré
    S'il y a possibilités
        Tant qu'il y a de possibilités
            resolution (case suivante, ligne, colonne,
            carré)
        Sinon pas de possibilités
            retourner null
    retourner etat de la case
}
```

Donc il n'y a pas de possibilités dans deux cas :

- Le choix est faux, il faut donc remonter notre arbre.
- Le tableau est plein, SUCCES.

Exécution de l'algorithme sur cet exemple :

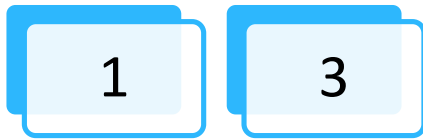
4	2		
			1
2			

Nous commençons par la 1^{ère} case en haut à gauche :

La case n'a pas de valeur fixe : nous étudions donc les possibilités en utilisant les contraintes liées à la ligne, à la colonne et au carré.

Nous obtenons donc 2 possibilités : 1 et 3.

Nous rappelons récursivement la fonction avec 1 en paramètre puis avec 3.



Nous expliquerons le cas pour 1 (le cas pour 3 étant le même).

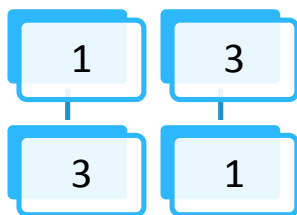
Dans la fonction avec 1 en paramètre :

La valeur de la case est fixée, nous passons à la case suivante. (De même pour 3)

Pour la case suivante 1ere ligne, 2^{ème} colonne : la valeur de la case n'est pas fixée, nous étudions les possibilités.

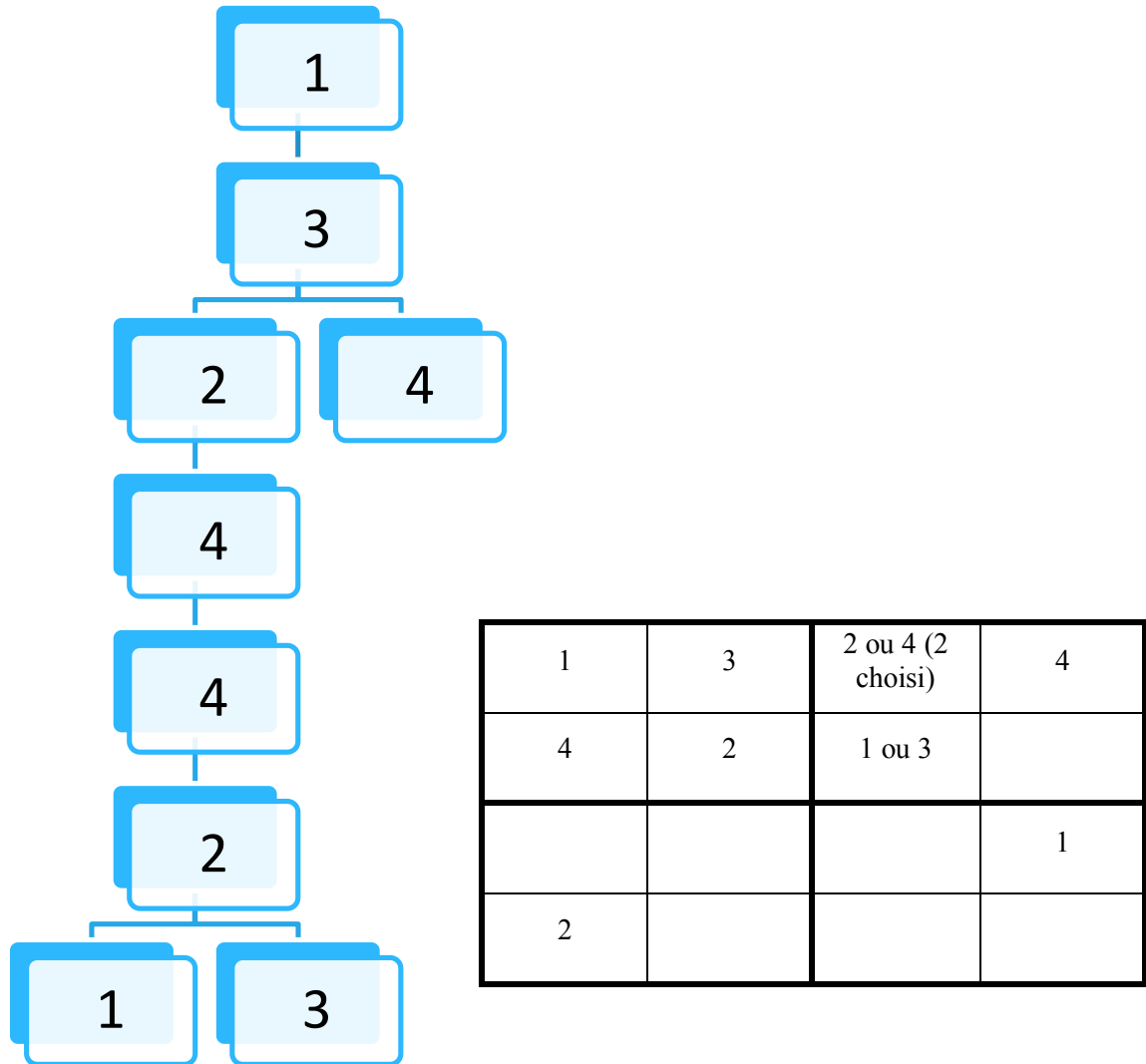
Nous obtenons 3 comme possibilité.

Nous rappelons récursivement la fonction avec 3 en paramètre.



Nous avons le schéma suivant.

Et ainsi de suite pour chaque case, à la fin nous obtenons quelque chose de cet ordre là si nous choisissons de prendre 1 comme première valeur :



Le sudoku sera résolu lorsque nous arriverons à la 16^e case dans notre récursivité. Sinon si avant la 16^{ème} case nous avons un cas d'aucune possibilité : la valeur choisie nous a conduits dans une impasse, par conséquent la branche de l'arbre n'est pas valide.