Marcel Délèze

SUDOKU, avec étapes intermédiaires

■ Mode d'emploi

1. Modifier la grille des données (recopier la grille vierge sur la grille de données et entrer les données).
2. Exécuter le programme *Mathematica* (Noyau / Evaluer le cahier).
3. Vérifier que la grille des données soit correcte (première grille de la liste des résultats).
4. Prendre connaissance des résultats. Le résultat final se trouve sur la dernière ligne.

   Usuellement, il devrait y avoir une et une seule grille complète sur la dernière ligne.

   Si la grille sur la dernière ligne est incomplète (ou les grilles de la dernière ligne sont incomplètes), le problème n'a pas de solution.

   Si plusieurs grilles complètes apparaissent sur la dernière ligne, il y a plusieurs solutions.

■ Entrée des données

Grille vierge (à recopier sur la grille "donnee" pour entrer une nouvelle grille):

```
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □;
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
□ □ □ □ □ □ □ □ □
```

Grille de données (à remplir):

```
          □ □ □ 9 □ □ □ □ 6
          □ □ □ 7 □ □ □ □ 1
          □ □ 5 □ 2 8 3 □ □
          □ 9 8 1 □ □ □ □ □
donnee = □ 7 □ □ □ □ □ 3 □;
          □ □ □ □ □ 4 9 8 □
          □ □ 3 5 4 □ 6 □ □
          1 □ □ □ □ 3 □ □ □
          2 □ □ □ □ 9 □ □ □
```

Représentation interne des grilles:

```
donnee = Apply[sdk, donnee /. {□ → Null}]
```

```
sdk[{Null, Null, Null, 9, Null, Null, Null, Null, 6},
 {Null, Null, Null, 7, Null, Null, Null, Null, 1},
 {Null, Null, 5, Null, 2, 8, 3, Null, Null}, {Null, 9, 8, 1, Null, Null, Null, Null, Null},
 {Null, 7, Null, Null, Null, Null, Null, 3, Null},
 {Null, Null, Null, Null, Null, 4, 9, 8, Null}, {Null, Null, 3, 5, 4, Null, 6, Null, Null},
 {1, Null, Null, Null, Null, 3, Null, Null, Null},
 {2, Null, Null, Null, Null, 9, Null, Null, Null}]
```

Affichage d'une grille:

```
affiche[grille_sdk] := FrameBox[GridBox[Apply[List, grille] /. {Null → "·"},
    RowLines → {False, False, True, False, False, True, False, False},
    ColumnLines → {False, False, True, False, False, True, False, False}]] // DisplayForm
```

```
affiche[g_List] := Map[affiche, g]
```

Résolution

Détermination des chiffres possibles

Extractions de lignes, de colonnes et de carrés:

```
ligne[grille_sdk, i_] := grille[[i]]

colonne[grille_sdk, j_] := Transpose[Apply[List, grille]][[j]]

carre[grille_sdk, i_, j_] :=
 Module[{ib, jb, ic, jc}, ib = 3 Quotient[i - 1, 3] + 1; jb = 3 Quotient[j - 1, 3] + 1;
  Flatten[Table[grille[[ib + ic, jb + jc]], {ic, 0, 2}, {jc, 0, 2}], 1]]
```

Pour une case vide, les chiffres possibles (qu'on pourrait essayer) sont ceux qui ne se trouvent ni dans la ligne, ni dans la colonne, ni dans le carré:

```
case[grille_sdk, i_, j_] := grille[[i, j]] /; IntegerQ[grille[[i, j]]]

case[grille_sdk, i_, j_] :=
 Complement[Range[1, 9], ligne[grille, i], colonne[grille, j], carre[grille, i, j]]

possible[grille_sdk] := Apply[sdk, Table[case[grille, i, j], {i, 1, 9}, {j, 1, 9}]];
```

Réduction parmi les chiffres possibles

Réduction
Si un chiffre possible pour une case n'apparaît pas ailleurs dans la ligne, alors la case peut se réduire à ce chiffre;
Si un chiffre possible pour une case n'apparaît pas ailleurs dans la colonne, alors la case peut se réduire à ce chiffre;
Si un chiffre possible pour une case n'apparaît pas ailleurs dans le carré, alors la case peut se réduire à ce chiffre.

```
Apply[And, Table[Count[donnee[g, i], 1, {1}] < 2, {i, 1, 9}]]
```

```
True
```

```
verifie[g_sdk] := Module[{i, j, k}, And[
   Apply[And, Table[
     Apply[And, Table[Count[ligne[g, i], k, {1}] < 2, {k, 1, 9}]],
     {i, 1, 9}]],
   Apply[And, Table[
     Apply[And, Table[Count[colonne[g, j], k, {1}] < 2, {k, 1, 9}]],
     {j, 1, 9}]],
   Apply[And, Table[Count[carre[g, 2, 2], k, {1}] < 2, {k, 1, 9}]],
   Apply[And, Table[Count[carre[g, 2, 5], k, {1}] < 2, {k, 1, 9}]],
   Apply[And, Table[Count[carre[g, 2, 8], k, {1}] < 2, {k, 1, 9}]],
   Apply[And, Table[Count[carre[g, 5, 2], k, {1}] < 2, {k, 1, 9}]],
   Apply[And, Table[Count[carre[g, 5, 5], k, {1}] < 2, {k, 1, 9}]],
   Apply[And, Table[Count[carre[g, 5, 8], k, {1}] < 2, {k, 1, 9}]],
   Apply[And, Table[Count[carre[g, 8, 2], k, {1}] < 2, {k, 1, 9}]],
   Apply[And, Table[Count[carre[g, 8, 5], k, {1}] < 2, {k, 1, 9}]],
   Apply[And, Table[Count[carre[g, 8, 8], k, {1}] < 2, {k, 1, 9}]]]]
```

```
reduis[grille_sdk] := Module[{g, t, i, j}, g = grille;
  Do[If[Length[g[[i, j]]] == 1, g[[i, j]] = First[g[[i, j]]]],
   {i, 1, 9}, {j, 1, 9}];
  Do[If[Head[grille[[i, j]]] === List,
    t = Intersection[grille[[i, j]], Complement[Range[1, 9],
       Flatten[Delete[ligne[grille, i], j]]]]; If[Length[t] > 0, g[[i, j]] = First[t]]],
   {i, 1, 9}, {j, 1, 9}];
  Do[If[Head[grille[[i, j]]] === List,
    t = Intersection[grille[[i, j]], Complement[Range[1, 9],
       Flatten[Delete[colonne[grille, j], i]]]]; If[Length[t] > 0, g[[i, j]] = First[t]]],
   {i, 1, 9}, {j, 1, 9}];
  Do[If[Head[grille[[i, j]]] === List,
    t = Intersection[grille[[i, j]], Complement[Range[1, 9],
       Flatten[Delete[carre[grille, i, j], 3 Mod[i - 1, 3] + Mod[j - 1, 3] + 1]]]];
    If[Length[t] > 0, g[[i, j]] = First[t]]],
   {i, 1, 9}, {j, 1, 9}];
  Apply[sdk, Table[If[IntegerQ[g[[i, j]]], g[[i, j]], Null], {i, 1, 9}, {j, 1, 9}]]]
```

■ Détermination des grilles qui comportent un chiffre de plus

Tester les impasses.

Former la grille suivante, ou plusieurs grilles lorsqu'il y a plusieurs possibilités, comme suit:

* choisir une case pour laquelle la liste des chiffres à essayer est de longueur minimale;
* pour chaque chiffre à essayer, former une grille comportant ce chiffre

```
suivant[grille_sdk] := Module[{p, e, n, nPoss, i, j, k, li, lj},
  n = Count[grille, _Integer, {2}];
  Which[
   (* si *) n == 81, (*fini*)
   grille,
   (* si *) p = possible[grille];
   MemberQ[p, {}, {2}] ⋁ Not[verifie[grille]], (* impasse *)
   {},
   (* si *) e = reduis[p]; Count[e, _Integer, {2}] > n, (* un seul successeur *)
   e,
   (* si *) True, (* plusieurs successeurs *)
   nPoss = 10;
   Do[If[1 ≤ Length[p[[i, j]]] < nPoss,
     li = i; lj = j; nPoss = Length[p[[i, j]]]],
    {i, 1, 9}, {j, 1, 9}];
   Table[ReplacePart[grille, p[[li, lj]][[k]], {li, lj}], {k, 1, nPoss}]
  ]]

suivant[g_List] := Flatten[Map[suivant, g]]
```

■ Itération

```
avanceQ[x_sdk] := Count[x, _Integer, {2}] < 81

avanceQ[x_List] := Apply[Or, Map[avanceQ, x]]

resous[grille_] := Module[{x, acc},
  x = grille; acc = {x};
  While[avanceQ[x],
   x = suivant[x]; If[UnsameQ[x, {}], AppendTo[acc, x]]];
  acc]

r = resous[donnee];

Length[r]
```

```
11
```

```
TableForm[affiche[r]]
```

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

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

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

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

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

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

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

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

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

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

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