Algorithmique et complexité M1

Université de Bourgogne, UFR Sciences et Techniques, Département I.E.M.

Projet « Algorithmique et complexité »

Groupe
Elio GENCE
Guillaume BELDILMI

Sujet n°4 : Eternity II

2023-2024

Contents

1.1 Présentation du projet

Eternity II est un jeu de puzzle inventé par Christopher Monckton en 2007. Le but du jeu est de placer des pièces de puzzle de manière à ce que les couleurs des bords des pièces soient identiques aux couleurs des bords adjacents. Les pièces sont carrées et chaque côté est coloré. Les pièces sont placées sur une grille de taille $n \times n$.

Notre projet était de créer un solveur capable de résoudre des puzzles d'Eternity II de taille 12 par 12 avec une dizaine de couleurs. Nos seules contraintes étaient que notre programme devait être écrit en Ocaml et que la résolution devait se faire par une méthode de recherche arborescente avec retour en arrière (backtrack).

Pour cela, nous avons divisé le projet en plusieurs étapes : tout d'abord, nous avons créé une structure de données pour représenter les pièces et la grille. Ensuite, nous avons créé un ensemble de fonctions permettant de générer un puzzle aléatoire, de mélanger les pièces du puzzle, et enfin de résoudre le puzzle. A cela s'ajoute une autre fonction permettant d'afficher le puzzle afin d'avoir un retour sur l'exécution du programme.

1.2 Organisation

Premièrement, nous avons réalisé un prototype conformément à l'ISO-1664 dans un langage plus abordable, Python. Ce prototype nous a permis de mieux comprendre le problème et de définir l'architecture de notre programme.

Ensuite, nous avons transcrit ce prototype en Ocaml, en corrigeant quelques erreurs et en ajoutant des fonctionnalités précédemment omises après avoir pris en compte le retour du corps enseignant.

Nous avons ensuite testé notre programme sur des puzzles progressivement de plus en plus grands.

Enfin, nous avons peaufiné notre programme en simplifiant le code et en corrigeant l'affichage du puzzle afin de rendre le résultat plus facile à manipuler.

1.3 Explication de l'algorithme utilisé

Nous avons utilisé un algorithme naïf de backtracking pour résoudre le puzzle. L'idée est de placer chaque pièce sur la grille en progressant selon le sens d'écriture. Si une pièce ne peut pas être placée, on revient à la pièce précédemment placée et on essaie une autre pièce. Si aucune pièce ne peut être placée, on revient à la pièce précédente, et ainsi de suite jusqu'à ce que le puzzle soit résolu. Une fois le puzzle résolu, on affiche le résultat et puis le programme poursuit son exécution afin de trouver d'autres solutions.

D'autres algorithmes plus performants existent, optimisants notamment l'ordre de placement des pièces (commençant par les pièces de bords puis progressant vers le centre en zigzag ou en spirale), mais nous avons choisi de rester sur un algorithme simple pour des raisons de simplicité et de temps.

Nous avons pour notre faire 2 versions différentes : une avec rotation de pièce et une autre sans.

1.3.1 Analyse de la complexité

Nous allons faire l'analysé de la complexité sur le code sans Rotation Lors d'une itération le nombre d'opération à faire est minimime (quelques test et opérations). Nous allons donc simplifier le tout et pour le bien de nos calcul nous alons supposer qu'une itération est égal à O(1).

Partons d'une situation impossible : tout les pièces sont exactements similaire (en ignorant également le concept de bordure). Ceci est notre pire scénario ! L'algorithme va donc tester toute les pièces à toutes les emplacement possible N = Taille du puzzle largeur/longueur N*N = Nombre de pièce En d'autres terme l'algorithme va parcourir N*N fois la première case puis (N*N)*(N*N) la seconde case etc etc Cela nous donne donc une complexité de O(N!)

Maintenant le meilleur scénario : toutes les pièces sont uniques et une seul solution est possible Alors dans cette situation l'algorithme va être bien plus rapide : Il va parcourir N*N la première case N*N la seconde case (car seulement une seule piece valide à la première case) etc etc Nous donnant donc une complexite de O(N)

Cela veux donc dire que la complexité va varier de O(N) à O(N!). Le facteur qui va décider de la complexité de l'algorithme est le nombre de couleur, plus celui ci sera élevé et plus les pièces seront uniques. En d'autre termes : moins nous avons de couleurs, plus de test sont nécéssaire à être effectuer pour verifier toutes les solutions et plus le code sera couteux/long à exécuter.

Pour le code avec Rotation il faut quadruplé chaque itération ce qui donne une complexité variable entre O(4N) et O(4N!)

1.3.2 Explication détaillés (code sans Rotation)

Pour notre code, nous pouvons le diviser en 5 parties : - définition d'une pièce 1.1 - creation des structures et initialisation 1.2 - test de pièce lors du backtrack 1.3 - backtrack 1.3 - affichage du résultat 1.3

Listing 1.1: Déclaration pièce (sources/tile.ml)

```
type tile = {
  id: int; (* permet de savoir l'emplacement d'origine *)
  top: int;
  right: int;
  bottom: int;
  left: int;
} ;;
```

Nous utilisons dans notre code 2 structures : board et tiles. Board est un tableau de tableau de pièce, il représente le plateau de jeu sur lequel on va venir placer les pièces pour essayer de résoudre le puzzle. Tiles quant à lui est un tableau de pièce, il va servir à stocker la liste de pièce encore non placé sur le board. (à noter que si une pièce identifiant est égal à -1 alors on considère l'emplacement comme vide)

L'initialisation du board ce fait avec un parcours de haut en bas et de gauche à droite/-droite à gauche une ligne sur deux (afin de simuler un parcours en zigzag dans les algorithmes de backtrack optimal) Lors de la création d'un pièce nous avons donc besoin de vérifier seulement 3 paramètres : - s'il est adjacent à une bordure (si c'est le cas le/les cotés adjacent prennent

la valeur 0) - la valeur du coté inférieur de la pièce supérieur - la valeur du coté gauche de la pièce à droite (ou inverse pour un parcours de droite à gauche) - le reste des coté prendront une valeur aléatoire parmis celle disponible (défini au tout début du code pour une modification rapide)

Listing 1.2: Initialisation du board (sources/init_board.ml)

```
let init_board size =
  let board = create_board size in
  for i = 0 to size - 1 do
    if i mod 2 = 0 then
      for j = 0 to size-1 do
        board.(i).(j) <-
          {
            top = if i = 0 then 0 else board.(i-1).(j).bottom;
            bottom = if i = size-1 then 0 else Random.int nbColor + 1;
            left = if j = 0 then 0 else board.(i).(j - 1).right;
            right = if j = size - 1 then 0 else Random.int nbColor + 1;
            id = i*size+j;
            flag=true;
          };
      done
    else
      for j = size-1 downto 0 do
        board.(i).(j) <-
          {
            top = board.(i-1).(j).bottom;
            bottom = if i = size-1 then 0 else Random.int nbColor + 1;
            left = if j = 0 then 0 else Random.int nbColor + 1;
            right = if j = size - 1 then 0 else board.(i).(j + 1).left;
            id = i*size+j;
            flag=true;
          };
      done;
  done;
  board;;

(* Création de gauche àdroite puis droite àgauche une ligne sur deux pour simuler le
   parcours optimal non fait pour le backtrack *)
```

Pour ce qui est du test des tuiles lors du backtrack voici le code et ce qui est testé

Listing 1.3: Test de compatibilité des pièces (fonction imbriqués)(sources/test_tile.ml)

```
let test_tile t i j =
       let res = ref true in

       (* Piece àgauche existante *)
       if j != 0 && board.(i).(j - 1).right != tiles.(t).left then
         res := false
       (* Bordure gauche impossible en intérieur *)
       else if j != 0 && tiles.(t).left = 0 then
         res := false
       (* Bordure droite impossible en intérieur *)
       else if j != (size - 1) && tiles.(t).right = 0 then
```

```
12          res := false
13        (* Bordure gauche *)
14        else if j = 0 && (tiles.(t).left != 0) then
15          res := false
16        (* Bordure droite *)
17        else if j = (size - 1) && (tiles.(t).right != 0) then
18          res := false
19
20        (* Piece supérieur existante *)
21        else if i != 0 && board.(i - 1).(j).bottom != tiles.(t).top then
22          res := false
23        (* Bordure supérieur impossible en intérieur *)
24        else if i != 0 && tiles.(t).top = 0 then
25          res := false
26        (* Bordure inférieur impossible en intérieur *)
27        else if i != (size - 1) && tiles.(t).bottom = 0 then
28          res := false
29        (* Bordure supérieur *)
30        else if i = 0 && (tiles.(t).top != 0) then
31          res := false
32        (* Bordure inférieur *)
33        else if i = (size - 1) && (tiles.(t).bottom != 0) then
34          res := false;
35
36        (* Parcours de droite àgauche, de haut en bas donc non nécéssaire
37        de vérifier les pièces en dessous et àdroite
38        mais nécéssaire de vérifier les bordures *)
39
40        !res;
41        in
```

Le backtrack la partie la plus important est également la partie la plus simple, le test s'occupant du principal tout ce qui reste à faire maintenant est de gérer les boucles de recherches et les appels récurssifs.

Listing 1.4: Backtrack (fonction principale) (sources/backtrack.ml)

```
1  let rec solve_backtrack board tiles i j = (* i et j représentent la position du board que l'
       on va tester *)
2    if board.(i).(j).id = -1 then (* Vérification qu'aucune pièce n'es déja placé *)
3      for t = 0 to (Array.length tiles) -1 do
4        if tiles.(t).id != -1 then (* Vérification que la pièce est valide *)
5          if test_tile t i j then begin (* Test de compatibilité de la pièce *)
6            (* Placement de la pièce et retrait de la liste des pièces disponibles *)
7            board.(i).(j) <- tiles.(t);
8            tiles.(t) <- { id = -1; top = 0; right = 0; bottom = 0; left = 0; flag=true };
9            (* backtrack *)
10           if j < size - 1 then
11             solve_backtrack board tiles i (j+1)
12           else if i < size - 1 then
13             solve_backtrack board tiles (i+1) 0
14           else
15             print_board board;
16           (* On retire la pièce et ajout dans la liste des
```

```
        (* pièce disponible pour chercher les prochaines solutions *)
        tiles.(t) <- board.(i).(j);
        board.(i).(j) <- { id = -1; top = 0; right = 0; bottom = 0; left = 0; flag=true };
      end
  done
 (* failsafe pour le backtrack *)
 else if j < size - 1 then
  solve_backtrack board tiles i (j+1)
 else if i < size - 1 then
  solve_backtrack board tiles (i+1) 0
 else
  print_board board;
;;
```

Enfin il ne nous reste plus que l'affichage qui est composé de plusieurs boucles imbriqués afin d'obtenir un affichage facilement lisible

Listing 1.5: Affichage du résultat (board)(sources/print.ml)

```
let print_board = fun board ->
 Array.iter (fun row ->
    Array.iter (fun tile ->
        Printf.printf "+--------------";
      ) row;
    Printf.printf "+\n";
    Array.iter (fun tile ->
        Printf.printf "| %02d " tile.top;
      ) row;
    Printf.printf "|\n";
    Array.iter (fun tile ->
        Printf.printf "|%02d %04d %02d" tile.left tile.id tile.right;
      ) row;
    Printf.printf "|\n";
    Array.iter (fun tile ->
        Printf.printf "| %02d " tile.bottom;
      ) row;
    Printf.printf "|\n";
  ) board;
 Printf.printf "+";
 Array.iter (fun _ ->
    Printf.printf "--------------+";
  ) board;
 Printf.printf "\n" ;;
```

Explication détaillés (code avec Rotation)

L'algorithme avec rotation est très similaire cependant dans le backtrack 1.4 et dans le test de pièce 1.3 on rajoute un nouveau paramètre (que l'on nomme k) variant entre 0 et 3 afin de pouvoir tester chaque rotation et ainsi obtenir de nouvelle solution.

Listing 1.6: Paramètre k pour savoir le sens de rotation de la pièce (sources/rotation.ml)

```
if k=0 then
```

		Taille*Taille					
Couleurs		4	6	8	10	11	12
	4	0,050~0,054	00,050~0,074	3,2~45s	+2m	/	/
	8	0,050~0,054	0,058~0,062	0,060~0,065	0,060~0,065	1~5s	+2m
	10	0,050~0,054	0,050~0,054	0,050~0,054	0,052~0,055	0,053~0,066	0,057~0,087

Figure 1.1: Temps sans rotation

```
      board.(i).(j) <- tiles.(t)
    else if k = 1 then
      board.(i).(j) <- rotate_tile_left tiles.(t)
    else if k = 2 then
      board.(i).(j) <- rotate_tile_180 tiles.(t)
    else
      board.(i).(j) <- rotate_tile_right tiles.(t);
```

Ainsi toutes les solutions possibles seront obtenus mais cela entraine 2 problème : La durée -> le code est bien plus long car bien plus de solution à tester Les solutions -> Le nombre de solution obtenus peux en réalité être divisé par 4 car nous trouvons des solutions "doublons" que l'on peut retrouver simplement en tournant le plateau de puzzle. Une solution possible à se problème serait de fixer un coin et de faire le backtrack à partir de ce moment la.

1.4 Résultats et analyses

1.4.1 Temps

Voici les temps observé pour un code sans rotation :

Comme on peux le constater la durée moyenne varie selon le nombre de couleur et la taille (logique). Cependant le temps varie de façon différente. Tandis que le temps se rallonge lorsque l'on augmente la taille, il diminue au contraire quand le nombre de couleur augmente. C'est logique car le nombre de cas à tester et de solution possible diminue lorsque le nombre de couleurs augmentent, si toute les pièces sont très atypiques alors il deviens plus dur de les substituer à l'inverse si les pièces se ressemblent (aka le nombre de couleur est faible) alors le programme va devoir tester bien plus de possibilité de résolution car les pièces vont s'imbriquer plus facilement les unes dans les autres.

Voici les temps observé pour un code avec rotation :

		Taille*Taille					
Couleurs		4	6	8	10	11	12
	4	0,063~0,090	++6m	/	/	/	/
	8	0,062~0,075	0,070~0,090	++2m	/	/	/
	10	0,062~0,075	0,070~0,090	0,3-1s	++2m	/	/
	15	0,062~0,075	0,070~0,090	0,07~0,130	5-10s	++2m	/
	20	0,062~0,075	0,070~0,090	0,070~0,095	0,2~0,8s	0,4~1s	1m4

Nous observons ici le même phénomène qu'avec le code sans rotation cependant les temps sont bien plus car, contrairement au précédent code, chaque pièce doit être tester 4 fois (une dans chaque sens) rendant la recherche plus fastidieuse mais également en augmentant le nombre de solution possible !

1.4.2 Affichage

Afin de simplifier l'exploitation des fichiers de résultats après l'exécution de notre programme (comptage, suppression d'éventuels doublons, ou encore d'autres traitements à base de grep ou autre...), nous affichons systématiquement la balise <deb> au début de chaque grille et sautons une ligne vide une fois cette dernière terminée.

Voici un exemple de puzzle de taille 5 par 5 avec 5 couleurs résolu par notre programme sans rotation :

Listing 1.7: Résultat sans Rotation (sources/resNR_5x5.txt)

```
Initial board
<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0000 05|05 0001 04|04 0002 03|03 0003 02|02 0004 00|
| 01 | 01 | 02 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 01 | 02 | 05 | 04 |
|00 0005 05|05 0006 05|05 0007 05|05 0008 01|01 0009 00|
| 02 | 01 | 01 | 01 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 01 | 01 | 01 | 02 |
|00 0010 02|02 0011 02|02 0012 02|02 0013 05|05 0014 00|
| 01 | 05 | 05 | 02 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 05 | 05 | 02 | 03 |
|00 0015 03|03 0016 01|01 0017 05|05 0018 02|02 0019 00|
| 03 | 01 | 04 | 05 | 01 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 04 | 05 | 01 |
|00 0020 01|01 0021 04|04 0022 04|04 0023 05|05 0024 00|
```

```
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+



Shuffled board
<deb>
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 05 | 02 | 01 |
|02 0019 00|00 0005 05|04 0023 05|05 0018 02|00 0015 03|
| 01 | 02 | 00 | 05 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 01 | 04 | 00 | 00 |
|01 0009 00|01 0021 04|04 0022 04|00 0000 05|04 0002 03|
| 02 | 00 | 00 | 01 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 01 | 01 | 00 | 05 |
|02 0011 02|02 0013 05|05 0006 05|02 0004 00|05 0008 01|
| 05 | 02 | 01 | 04 | 01 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 05 | 03 | 05 | 01 |
|05 0014 00|03 0016 01|00 0020 01|01 0017 05|02 0012 02|
| 03 | 01 | 00 | 04 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 02 | 02 | 00 | 00 |
|05 0024 00|05 0007 05|00 0010 02|03 0003 02|05 0001 04|
| 00 | 01 | 01 | 05 | 01 |
+--------------+--------------+--------------+--------------+--------------+



Solving: may the time be with you
<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0000 05|05 0001 04|04 0002 03|03 0003 02|02 0004 00|
| 01 | 01 | 02 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 01 | 02 | 05 | 04 |
|00 0005 05|05 0006 05|05 0007 05|05 0008 01|01 0009 00|
| 02 | 01 | 01 | 01 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 01 | 01 | 01 | 02 |
|00 0010 02|02 0011 02|02 0012 02|02 0013 05|05 0014 00|
| 01 | 05 | 05 | 02 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 05 | 05 | 02 | 03 |
|00 0015 03|03 0016 01|01 0017 05|05 0018 02|02 0019 00|
| 03 | 01 | 04 | 05 | 01 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 04 | 05 | 01 |
|00 0020 01|01 0021 04|04 0022 04|04 0023 05|05 0024 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+
```

```
<deb>
+-------------+-------------+-------------+-------------+-------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0000 05|05 0001 04|04 0002 03|03 0003 02|02 0004 00|
| 01 | 01 | 02 | 05 | 04 |
+-------------+-------------+-------------+-------------+-------------+
| 01 | 01 | 02 | 05 | 04 |
|00 0005 05|05 0006 05|05 0007 05|05 0008 01|01 0009 00|
| 02 | 01 | 01 | 01 | 02 |
+-------------+-------------+-------------+-------------+-------------+
| 02 | 01 | 01 | 01 | 02 |
|00 0010 02|02 0012 02|02 0011 02|02 0013 05|05 0014 00|
| 01 | 05 | 05 | 02 | 03 |
+-------------+-------------+-------------+-------------+-------------+
| 01 | 05 | 05 | 02 | 03 |
|00 0015 03|03 0016 01|01 0017 05|05 0018 02|02 0019 00|
| 03 | 01 | 04 | 05 | 01 |
+-------------+-------------+-------------+-------------+-------------+
| 03 | 01 | 04 | 05 | 01 |
|00 0020 01|01 0021 04|04 0022 04|04 0023 05|05 0024 00|
| 00 | 00 | 00 | 00 | 00 |
+-------------+-------------+-------------+-------------+-------------+



Solving: end of time
```

Voici un exemple de puzzle de taille 5 par 5 avec 5 couleurs résolu par notre programme avec rotation : 1.11 Comme dis lors de l'explication, nous obtenons tous les résultats dans 4 rotations possibles donc il faut diviser par 4 le nombre de résultat obtenus pour avoir le nombre réel de solutions possibles (un code prennant en compte ce détail sera ajouter en annexe 1.10

1.5 Conclusion

Finalement, et pour conclure, nous avons réussi à implémenter ce solveur. Nous pensons avoir compris en profondeur les subtilités algorithmique derrière ce dernier et leur importance, et ce, malgré les améliorations possibles au résultat de notre travail. Nous restons donc relativement satisfait de notre projet et remercions les profs de TP (Boris Bordeaux et Diarra Ibrahim) qui sont resté disponible pour répondre à toutes nos questions (notamment sur comment pour effectuer le parcours de façon optimal par exemple).

1.6 Annexe

Listing 1.8: Code sans Rotation (sources/code.ml)

```
Random.self_init ();;

let size = 12 ;;
```

```
4  let nbColor = 10 ;;
5
6  type tile = {
7    id: int;
8    top: int;
9    right: int;
10   bottom: int;
11   left: int;
12 } ;;
13
14 let create_board size =
15   Array.init size (fun _ ->
16       Array.init size (fun _ ->
17           { id=(-1); top=0; right=0; left=0; bottom=0}
18         )
19     ) ;;
20
21 let init_board size =
22   let board = create_board size in
23   for i = 0 to size - 1 do
24     if i mod 2 = 0 then
25       for j = 0 to size-1 do
26         board.(i).(j) <-
27           {
28             top = if i = 0 then 0 else board.(i-1).(j).bottom;
29             bottom = if i = size-1 then 0 else Random.int nbColor + 1;
30             left = if j = 0 then 0 else board.(i).(j - 1).right;
31             right = if j = size - 1 then 0 else Random.int nbColor + 1;
32             id = i*size+j;
33           };
34       done
35     else
36       for j = size-1 downto 0 do
37         board.(i).(j) <-
38           {
39             top = board.(i-1).(j).bottom;
40             bottom = if i = size-1 then 0 else Random.int nbColor + 1;
41             left = if j = 0 then 0 else Random.int nbColor + 1;
42             right = if j = size - 1 then 0 else board.(i).(j + 1).left;
43             id = i*size+j;
44           };
45       done;
46   done;
47   board;;
48
49 let print_board = fun board ->
50   Array.iter (fun row ->
51       Array.iter (fun tile ->
52           Printf.printf "+--------------";
53         ) row;
54       Printf.printf "+\n";
55       Array.iter (fun tile ->
56           Printf.printf "| %02d " tile.top;
57         ) row;
```

```
      Printf.printf "|\n";
      Array.iter (fun tile ->
          Printf.printf "|%02d %04d %02d" tile.left tile.id tile.right;
        ) row;
      Printf.printf "|\n";
      Array.iter (fun tile ->
          Printf.printf "| %02d " tile.bottom;
        ) row;
      Printf.printf "|\n";
    ) board;
  Printf.printf "+";
  Array.iter (fun _ ->
      Printf.printf "--------------+";
    ) board;
  Printf.printf "\n" ;;


let shuffle_board board =
  for i = 0 to size - 1 do
    for j = 0 to size - 1 do
      let random_i = Random.int size in
      let random_j = Random.int size in
      let temp_tile = board.(i).(j) in
      board.(i).(j) <- board.(random_i).(random_j);
      board.(random_i).(random_j) <- temp_tile;
    done;
  done;
  board ;;

let init_tiles board=
  let tiles = Array.make (size*size) ({ id=(-1); top=0; right=0; left=0; bottom=0}) in
  for i = 0 to size - 1 do
    for j = 0 to size - 1 do
      tiles.((i)*size+j) <- board.(i).(j);
    done;
  done;
  tiles ;;

let rec solve_backtrack board tiles i j =

  let test_tile t i j =
    let res = ref true in
    if j != 0 && board.(i).(j - 1).right != tiles.(t).left then
      res := false
    else if j != 0 && tiles.(t).left = 0 then
      res := false
    else if j != (size - 1) && tiles.(t).right = 0 then
      res := false
    else if i != 0 && board.(i - 1).(j).bottom != tiles.(t).top then
      res := false
    else if i != 0 && tiles.(t).top = 0 then
      res := false
    else if i != (size - 1) && tiles.(t).bottom = 0 then
      res := false
```

```
    else if i = 0 && (tiles.(t).top != 0) then
      res := false
    else if i = (size - 1) && (tiles.(t).bottom != 0) then
      res := false
    else if j = 0 && (tiles.(t).left != 0) then
      res := false
    else if j = (size - 1) && (tiles.(t).right != 0) then
      res := false;
    !res;
  in

  if board.(i).(j).id = -1 then
    for t = 0 to (Array.length tiles) -1 do
      if tiles.(t).id != -1 then
        if test_tile t i j then begin
          board.(i).(j) <- tiles.(t);
          tiles.(t) <- { id = -1; top = 0; right = 0; bottom = 0; left = 0};
          if i < size - 1 then
            solve_backtrack board tiles (i+1) j
          else if j < size - 1 then
            solve_backtrack board tiles 0 (j+1)
          else
            print_board board;
          tiles.(t) <- board.(i).(j);
          board.(i).(j) <- { id = -1; top = 0; right = 0; bottom = 0; left = 0};
        end
    done
  else if i < size - 1 then
    solve_backtrack board tiles (i+1) j
  else if j < size - 1 then
    solve_backtrack board tiles 0 (j+1)
  else
    print_board board;
;;


(* main *)

let b = init_board size ;;
Printf.printf "Initial board\n" ;;
print_board b ;;

Printf.printf "\n\nShuffled board\n" ;;
let b = shuffle_board b ;;
print_board b ;;

Printf.printf "\n\nSolving: may the time be with you\n";;
let tiles = init_tiles b;;
let board = create_board size;;
solve_backtrack board tiles 0 0;
Printf.printf "\n\nSolving: end of time\n" ;
```

Listing 1.9: Code avec Rotation (sources/codeRotation.ml)

```
Random.self_init ();;

let size = 4 ;;
let nbColor = 3 ;;

type tile = {
  id: int;
  top: int;
  right: int;
  bottom: int;
  left: int;
} ;;

let rotate_tile_right t:tile =
  {
    id=t.id;
    top = t.left;
    bottom = t.right;
    left = t.bottom;
    right = t.top;
  } ;;

let rotate_tile_180 t:tile =
  {
    id=t.id;
    top = t.bottom;
    bottom = t.top;
    left = t.right;
    right = t.left;
  } ;;


let rotate_tile_left t:tile =
  {
    id=t.id;
    right = t.bottom;
    top = t.right;
    bottom = t.left;
    left = t.top;
  } ;;

let create_random_tile () tile =
  {
    id=0;
    top = Random.int nbColor;
    bottom = Random.int nbColor;
    left = Random.int nbColor;
    right = Random.int nbColor;
  } ;;


let create_board size =
  Array.init size (fun _ ->
      Array.init size (fun _ ->
```

```
        { id=(-1); top=0; right=0; left=0; bottom=0}
      )
  ) ;;

let init_board size =
  let board = create_board size in
  for i = 0 to size - 1 do
    if i mod 2 = 0 then
      for j = 0 to size-1 do
        board.(i).(j) <-
          {
            top = if i = 0 then 0 else board.(i-1).(j).bottom;
            bottom = if i = size-1 then 0 else Random.int nbColor + 1;
            left = if j = 0 then 0 else board.(i).(j - 1).right;
            right = if j = size - 1 then 0 else Random.int nbColor + 1;
            id = i*size+j;
          };
      done
    else
      for j = size-1 downto 0 do
        board.(i).(j) <-
          {
            top = board.(i-1).(j).bottom;
            bottom = if i = size-1 then 0 else Random.int nbColor + 1;
            left = if j = 0 then 0 else Random.int nbColor + 1;
            right = if j = size - 1 then 0 else board.(i).(j + 1).left;
            id = i*size+j;
          };
      done;
  done;
  board;;

let print_board = fun board ->
  Array.iter (fun row ->
      Array.iter (fun tile ->
          Printf.printf "+--------------";
        ) row;
      Printf.printf "+\n";
      Array.iter (fun tile ->
          Printf.printf "|  %02d  " tile.top;
        ) row;
      Printf.printf "|\n";
      Array.iter (fun tile ->
          Printf.printf "|%02d %04d %02d" tile.left tile.id tile.right;
        ) row;
      Printf.printf "|\n";
      Array.iter (fun tile ->
          Printf.printf "|  %02d  " tile.bottom;
        ) row;
      Printf.printf "|\n";
    ) board;
  Printf.printf "+";
  Array.iter (fun _ ->
      Printf.printf "--------------+";
```

```
    ) board;
  Printf.printf "\n" ;;


let shuffle_board board =
  for i = 0 to size - 1 do
    for j = 0 to size - 1 do
      let random_i = Random.int size in
      let random_j = Random.int size in
      let temp_tile = board.(i).(j) in
      board.(i).(j) <- board.(random_i).(random_j);
      board.(random_i).(random_j) <- temp_tile;
    done;
  done;
  board ;;

let init_tiles board=
  let tiles = Array.make (size*size) ({ id=(-1); top=0; right=0; left=0; bottom=0}) in
  for i = 0 to size - 1 do
    for j = 0 to size - 1 do
      tiles.((i)*size+j) <- board.(i).(j);
    done;
  done;
  tiles ;;

let rec solve_backtrack board tiles i j =

  let test_tile t i j k =
    let res = ref true in

    let test_tile = ref tiles.(t) in

    if k = 1 then
      test_tile := rotate_tile_left !test_tile
    else if k = 2 then
      test_tile := rotate_tile_180 !test_tile
    else if k = 3 then
      test_tile := rotate_tile_right !test_tile;

    if j != 0 && board.(i).(j - 1).right != !test_tile.left then
      res := false
    else if j != 0 && !test_tile.left = 0 then
      res := false
    else if j != (size - 1) && !test_tile.right = 0 then
      res := false
    else if i != 0 && board.(i - 1).(j).bottom != !test_tile.top then
      res := false
    else if i != 0 && !test_tile.top = 0 then
      res := false
    else if i != (size - 1) && !test_tile.bottom = 0 then
      res := false
    else if i = 0 && (!test_tile.top != 0) then
      res := false
    else if i = (size - 1) && (!test_tile.bottom != 0) then
```

```
      res := false
    else if j = 0 && (!test_tile.left != 0) then
      res := false
    else if j = (size - 1) && (!test_tile.right != 0) then
      res := false;
    !res;
  in

  if board.(i).(j).id = -1 then
    for t = 0 to (Array.length tiles) -1 do
      if tiles.(t).id != -1 then
        for k = 0 to 3 do
          if test_tile t i j k then
            begin
              if k=0 then
                board.(i).(j) <- tiles.(t)
              else if k = 1 then
                board.(i).(j) <- rotate_tile_left tiles.(t)
              else if k = 2 then
                board.(i).(j) <- rotate_tile_180 tiles.(t)
              else
                board.(i).(j) <- rotate_tile_right tiles.(t);

              tiles.(t) <- { id = -1; top = 0; right = 0; bottom = 0; left = 0 };
              if j < size - 1 then
                solve_backtrack board tiles i (j+1)
              else if i < size - 1 then
                solve_backtrack board tiles (i+1) 0
              else
                print_board board;

              if k=0 then
                tiles.(t) <- board.(i).(j)
              else if k = 1 then
                tiles.(t) <- rotate_tile_right board.(i).(j)
              else if k = 2 then
                tiles.(t) <- rotate_tile_180 board.(i).(j)
              else
                tiles.(t) <- rotate_tile_left board.(i).(j);
              board.(i).(j) <- { id = -1; top = 0; right = 0; bottom = 0; left = 0 };
            end
        done
    done
  else if j < size - 1 then
    solve_backtrack board tiles i (j+1)
  else if i < size - 1 then
    solve_backtrack board tiles (i+1) 0
  else
    print_board board;
;;


(* main *)

```

```
let b = init_board size ;;
Printf.printf "Initial board\n" ;;
print_board b ;;

Printf.printf "\n\nShuffled board\n" ;;
let b = shuffle_board b ;;
print_board b ;;

Printf.printf "\n\nSolving: may the time be with you\n";;
let tiles = init_tiles b;;
let board = create_board size;;
solve_backtrack board tiles 0 0;
Printf.printf "\n\nSolving: end of time\n" ;
```

Listing 1.10: Code avec Rotation et première tuile Locked (sources/codeRotation1stTileLocked.ml)

```
Random.self_init ();;

let size = 4 ;;
let nbColor = 5 ;;

type tile = {
  id: int;
  top: int;
  right: int;
  bottom: int;
  left: int;
} ;;

let rotate_tile_right t:tile =
  {
    id=t.id;
    top = t.left;
    bottom = t.right;
    left = t.bottom;
    right = t.top;
  } ;;

let rotate_tile_180 t:tile =
  {
    id=t.id;
    top = t.bottom;
    bottom = t.top;
    left = t.right;
    right = t.left;
  } ;;

let rotate_tile_left t:tile =
  {
    id=t.id;
    right = t.bottom;
    top = t.right;
    bottom = t.left;
```

```
    left = t.top;
  } ;;

let create_board size =
  Array.init size (fun _ ->
      Array.init size (fun _ ->
          { id=(-1); top=0; right=0; left=0; bottom=0}
        )
    ) ;;

let init_board size =
  let board = create_board size in
  for i = 0 to size - 1 do
    if i mod 2 = 0 then
      for j = 0 to size-1 do
        board.(i).(j) <-
          {
            top = if i = 0 then 0 else board.(i-1).(j).bottom;
            bottom = if i = size-1 then 0 else Random.int nbColor + 1;
            left = if j = 0 then 0 else board.(i).(j - 1).right;
            right = if j = size - 1 then 0 else Random.int nbColor + 1;
            id = i*size+j;
          };
      done
    else
      for j = size-1 downto 0 do
        board.(i).(j) <-
          {
            top = board.(i-1).(j).bottom;
            bottom = if i = size-1 then 0 else Random.int nbColor + 1;
            left = if j = 0 then 0 else Random.int nbColor + 1;
            right = if j = size - 1 then 0 else board.(i).(j + 1).left;
            id = i*size+j;
          };
      done;
  done;
  board;;

let print_board = fun board ->
  Array.iter (fun row ->
      Array.iter (fun tile ->
          Printf.printf "+--------------";
        ) row;
      Printf.printf "+\n";
      Array.iter (fun tile ->
          Printf.printf "| %02d " tile.top;
        ) row;
      Printf.printf "|\n";
      Array.iter (fun tile ->
          Printf.printf "|%02d %04d %02d" tile.left tile.id tile.right;
        ) row;
      Printf.printf "|\n";
      Array.iter (fun tile ->
          Printf.printf "| %02d " tile.bottom;
```

```
92        ) row;
93      Printf.printf "|\n";
94    ) board;
95  Printf.printf "+";
96  Array.iter (fun _ ->
97      Printf.printf "--------------+";
98    ) board;
99  Printf.printf "\n" ;;
100
101
102 let shuffle_board board =
103   for i = 0 to size - 1 do
104     for j = 0 to size - 1 do
105       if not (i = 0 && j = 0) then
106         begin
107           let random_i = (Random.int (size-1)) +1 in
108           let random_j = (Random.int (size-1)) +1 in
109           let temp_tile = board.(i).(j) in
110           board.(i).(j) <- board.(random_i).(random_j);
111           board.(random_i).(random_j) <- temp_tile;
112         end;
113     done;
114   done;
115   board ;;
116
117 let init_tiles board =
118   let tiles = Array.make ((size*size)) ({ id=(-1); top=0; right=0; left=0; bottom=0}) in
119   for i = 0 to size - 1 do
120     for j = 0 to size - 1 do
121       tiles.((i)*size+j) <- board.(i).(j);
122     done;
123   done;
124   tiles ;;
125
126 let rec solve_backtrack board tiles i j =
127
128   let test_tile t i j k =
129     let res = ref true in
130
131     let test_tile = ref tiles.(t) in
132
133     if k = 1 then
134       test_tile := rotate_tile_left !test_tile
135     else if k = 2 then
136       test_tile := rotate_tile_180 !test_tile
137     else if k = 3 then
138       test_tile := rotate_tile_right !test_tile;
139
140     if j != 0 && board.(i).(j - 1).right != !test_tile.left then
141       res := false
142     else if j != 0 && !test_tile.left = 0 then
143       res := false
144     else if j != (size - 1) && !test_tile.right = 0 then
145       res := false
```

```
    else if i != 0 && board.(i - 1).(j).bottom != !test_tile.top then
      res := false
    else if i != 0 && !test_tile.top = 0 then
      res := false
    else if i != (size - 1) && !test_tile.bottom = 0 then
      res := false
    else if i = 0 && (!test_tile.top != 0) then
      res := false
    else if i = (size - 1) && (!test_tile.bottom != 0) then
      res := false
    else if j = 0 && (!test_tile.left != 0) then
      res := false
    else if j = (size - 1) && (!test_tile.right != 0) then
      res := false;
    !res;
  in

  if board.(i).(j).id = -1 then
    for t = 1 to (Array.length tiles) -1 do
      if tiles.(t).id != -1 then
        for k = 0 to 3 do
          if test_tile t i j k then
            begin
              if k=0 then
                board.(i).(j) <- tiles.(t)
              else if k = 1 then
                board.(i).(j) <- rotate_tile_left tiles.(t)
              else if k = 2 then
                board.(i).(j) <- rotate_tile_180 tiles.(t)
              else
                board.(i).(j) <- rotate_tile_right tiles.(t);

              tiles.(t) <- { id = -1; top = 0; right = 0; bottom = 0; left = 0 };
              if j < size - 1 then
                solve_backtrack board tiles i (j+1)
              else if i < size - 1 then
                solve_backtrack board tiles (i+1) 0
              else
                print_board board;

              if k=0 then
                tiles.(t) <- board.(i).(j)
              else if k = 1 then
                tiles.(t) <- rotate_tile_right board.(i).(j)
              else if k = 2 then
                tiles.(t) <- rotate_tile_180 board.(i).(j)
              else
                tiles.(t) <- rotate_tile_left board.(i).(j);
              board.(i).(j) <- { id = -1; top = 0; right = 0; bottom = 0; left = 0 };
            end
        done
    done
  else if j < size - 1 then
    solve_backtrack board tiles i (j+1)
```

```
  else if i < size - 1 then
    solve_backtrack board tiles (i+1) 0
  else
    print_board board;
;;


(* main *)

let b = init_board size ;;
Printf.printf "Initial board\n" ;;
print_board b ;;

Printf.printf "\n\nShuffled board\n" ;;
let b = shuffle_board b ;;
print_board b ;;

Printf.printf "\n\nSolving: may the time be with you\n";;
let tiles = init_tiles b;;
let board = create_board size;;
board.(0).(0) <- tiles.(0);;
tiles.(0) <- { id = -1; top = 0; right = 0; bottom = 0; left = 0 };;
solve_backtrack board tiles 0 0;;
Printf.printf "\n\nSolving: end of time\n" ;;
```

Listing 1.11: Résultat avec Rotation (sources/resR_5x5.txt)

```
Initial board
<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0000 05|05 0001 03|03 0002 02|02 0003 04|04 0004 00|
| 02 | 05 | 04 | 02 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 05 | 04 | 02 | 02 |
|00 0005 04|04 0006 05|05 0007 05|05 0008 05|05 0009 00|
| 04 | 05 | 02 | 05 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 05 | 02 | 05 | 05 |
|00 0010 01|01 0011 05|05 0012 02|02 0013 01|01 0014 00|
| 03 | 04 | 02 | 01 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 01 | 03 |
|00 0015 01|01 0016 01|01 0017 04|04 0018 01|01 0019 00|
| 05 | 02 | 04 | 01 | 01 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 02 | 04 | 01 | 01 |
|00 0020 04|04 0021 03|03 0022 03|03 0023 02|02 0024 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+



Shuffled board
```

```
<deb>
+--------------+--------------+--------------+--------------+--------------+
| 05 | 02 | 04 | 05 | 02 |
|00 0020 04|05 0009 00|00 0010 01|01 0011 05|00 0005 04|
| 00 | 05 | 03 | 04 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 01 | 00 | 02 | 04 |
|02 0013 01|03 0023 02|02 0003 04|01 0017 04|05 0007 05|
| 01 | 00 | 02 | 04 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 00 | 04 | 03 | 02 | 00 |
|03 0002 02|03 0022 03|00 0015 01|05 0008 05|04 0004 00|
| 04 | 00 | 05 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 00 | 01 | 00 | 02 | 05 |
|00 0000 05|02 0024 00|05 0001 03|04 0021 03|04 0006 05|
| 02 | 00 | 05 | 00 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 04 | 03 | 05 | 01 |
|05 0012 02|01 0016 01|01 0019 00|01 0014 00|04 0018 01|
| 02 | 02 | 01 | 03 | 01 |
+--------------+--------------+--------------+--------------+--------------+



Solving: may the time be with you
<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0020 05|05 0015 03|03 0010 04|04 0005 02|02 0000 00|
| 04 | 01 | 01 | 04 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 01 | 01 | 04 | 05 |
|00 0003 02|02 0016 04|04 0011 05|05 0006 05|05 0001 00|
| 02 | 01 | 05 | 05 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 01 | 05 | 05 | 03 |
|00 0002 04|04 0017 02|02 0012 02|02 0007 04|04 0022 00|
| 03 | 04 | 02 | 05 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 05 | 03 |
|00 0023 01|01 0018 01|01 0013 05|05 0008 02|02 0021 00|
| 02 | 01 | 01 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 01 | 01 | 05 | 04 |
|00 0024 01|01 0019 03|03 0014 05|05 0009 02|02 0004 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0020 05|05 0015 03|03 0010 04|04 0005 02|02 0000 00|
| 04 | 01 | 01 | 04 | 05 |
```

```
+--------------+--------------+--------------+--------------+--------------+
| 04 | 01 | 01 | 04 | 05 |
|00 0021 02|02 0016 04|04 0011 05|05 0006 05|05 0001 00|
| 03 | 01 | 05 | 05 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 05 | 05 | 03 |
|00 0022 04|04 0017 02|02 0012 02|02 0007 04|04 0002 00|
| 03 | 04 | 02 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 05 | 02 |
|00 0023 01|01 0018 01|01 0013 05|05 0008 02|02 0003 00|
| 02 | 01 | 01 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 01 | 01 | 05 | 04 |
|00 0024 01|01 0019 03|03 0014 05|05 0009 02|02 0004 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0020 05|05 0015 03|03 0022 03|03 0019 01|01 0024 00|
| 04 | 01 | 04 | 01 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 01 | 04 | 01 | 02 |
|00 0021 02|02 0016 04|04 0017 01|01 0018 01|01 0023 00|
| 03 | 01 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 02 | 04 | 03 |
|00 0014 01|01 0013 02|02 0012 05|05 0011 01|01 0010 00|
| 05 | 05 | 02 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 05 | 02 | 05 | 04 |
|00 0009 05|05 0008 05|05 0007 05|05 0006 04|04 0005 00|
| 02 | 02 | 04 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 02 | 04 | 05 | 02 |
|00 0004 04|04 0003 02|02 0002 03|03 0001 05|05 0000 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0020 05|05 0014 03|03 0010 04|04 0005 02|02 0000 00|
| 04 | 01 | 01 | 04 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 01 | 01 | 04 | 05 |
|00 0003 02|02 0016 04|04 0011 05|05 0006 05|05 0001 00|
| 02 | 01 | 05 | 05 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 01 | 05 | 05 | 03 |
|00 0002 04|04 0017 02|02 0012 02|02 0007 04|04 0022 00|
| 03 | 04 | 02 | 05 | 03 |
```

```
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 05 | 03 |
|00 0023 01|01 0018 01|01 0013 05|05 0008 02|02 0021 00|
| 02 | 01 | 01 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 01 | 01 | 05 | 04 |
|00 0024 01|01 0019 03|03 0015 05|05 0009 02|02 0004 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0020 05|05 0014 03|03 0010 04|04 0005 02|02 0000 00|
| 04 | 01 | 01 | 04 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 01 | 01 | 04 | 05 |
|00 0021 02|02 0016 04|04 0011 05|05 0006 05|05 0001 00|
| 03 | 01 | 05 | 05 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 05 | 05 | 03 |
|00 0022 04|04 0017 02|02 0012 02|02 0007 04|04 0002 00|
| 03 | 04 | 02 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 05 | 02 |
|00 0023 01|01 0018 01|01 0013 05|05 0008 02|02 0003 00|
| 02 | 01 | 01 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 01 | 01 | 05 | 04 |
|00 0024 01|01 0019 03|03 0015 05|05 0009 02|02 0004 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0020 05|05 0014 03|03 0022 03|03 0019 01|01 0024 00|
| 04 | 01 | 04 | 01 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 01 | 04 | 01 | 02 |
|00 0021 02|02 0016 04|04 0017 01|01 0018 01|01 0023 00|
| 03 | 01 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 02 | 04 | 03 |
|00 0015 01|01 0013 02|02 0012 05|05 0011 01|01 0010 00|
| 05 | 05 | 02 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 05 | 02 | 05 | 04 |
|00 0009 05|05 0008 05|05 0007 05|05 0006 04|04 0005 00|
| 02 | 02 | 04 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 02 | 04 | 05 | 02 |
|00 0004 04|04 0003 02|02 0002 03|03 0001 05|05 0000 00|
| 00 | 00 | 00 | 00 | 00 |
```

```
+-------------+-------------+-------------+-------------+-------------+

<deb>
+-------------+-------------+-------------+-------------+-------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0004 02|02 0009 05|05 0015 03|03 0021 04|04 0020 00|
| 04 | 05 | 01 | 02 | 05 |
+-------------+-------------+-------------+-------------+-------------+
| 04 | 05 | 01 | 02 | 05 |
|00 0003 02|02 0008 05|05 0013 01|01 0016 01|01 0014 00|
| 02 | 05 | 02 | 04 | 03 |
+-------------+-------------+-------------+-------------+-------------+
| 02 | 05 | 02 | 04 | 03 |
|00 0002 04|04 0007 02|02 0012 02|02 0017 04|04 0022 00|
| 03 | 05 | 05 | 01 | 03 |
+-------------+-------------+-------------+-------------+-------------+
| 03 | 05 | 05 | 01 | 03 |
|00 0001 05|05 0006 05|05 0011 04|04 0018 01|01 0019 00|
| 05 | 04 | 01 | 01 | 01 |
+-------------+-------------+-------------+-------------+-------------+
| 05 | 04 | 01 | 01 | 01 |
|00 0000 02|02 0005 04|04 0010 03|03 0023 02|02 0024 00|
| 00 | 00 | 00 | 00 | 00 |
+-------------+-------------+-------------+-------------+-------------+

<deb>
+-------------+-------------+-------------+-------------+-------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0004 02|02 0009 05|05 0015 03|03 0019 01|01 0024 00|
| 04 | 05 | 01 | 01 | 02 |
+-------------+-------------+-------------+-------------+-------------+
| 04 | 05 | 01 | 01 | 02 |
|00 0003 02|02 0008 05|05 0013 01|01 0018 01|01 0023 00|
| 02 | 05 | 02 | 04 | 03 |
+-------------+-------------+-------------+-------------+-------------+
| 02 | 05 | 02 | 04 | 03 |
|00 0002 04|04 0007 02|02 0012 02|02 0017 04|04 0022 00|
| 03 | 05 | 05 | 01 | 03 |
+-------------+-------------+-------------+-------------+-------------+
| 03 | 05 | 05 | 01 | 03 |
|00 0001 05|05 0006 05|05 0011 04|04 0016 02|02 0021 00|
| 05 | 04 | 01 | 01 | 04 |
+-------------+-------------+-------------+-------------+-------------+
| 05 | 04 | 01 | 01 | 04 |
|00 0000 02|02 0005 04|04 0010 03|03 0014 05|05 0020 00|
| 00 | 00 | 00 | 00 | 00 |
+-------------+-------------+-------------+-------------+-------------+

<deb>
+-------------+-------------+-------------+-------------+-------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0004 02|02 0009 05|05 0015 03|03 0019 01|01 0024 00|
| 04 | 05 | 01 | 01 | 02 |
+-------------+-------------+-------------+-------------+-------------+
```

```
| 04 | 05 | 01 | 01 | 02 |
|00 0021 02|02 0008 05|05 0013 01|01 0018 01|01 0023 00|
| 03 | 05 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 05 | 02 | 04 | 03 |
|00 0022 04|04 0007 02|02 0012 02|02 0017 04|04 0002 00|
| 03 | 05 | 05 | 01 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 05 | 05 | 01 | 02 |
|00 0001 05|05 0006 05|05 0011 04|04 0016 02|02 0003 00|
| 05 | 04 | 01 | 01 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 04 | 01 | 01 | 04 |
|00 0000 02|02 0005 04|04 0010 03|03 0014 05|05 0020 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0004 02|02 0009 05|05 0014 03|03 0021 04|04 0020 00|
| 04 | 05 | 01 | 02 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 05 | 01 | 02 | 05 |
|00 0003 02|02 0008 05|05 0013 01|01 0016 01|01 0015 00|
| 02 | 05 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 05 | 02 | 04 | 03 |
|00 0002 04|04 0007 02|02 0012 02|02 0017 04|04 0022 00|
| 03 | 05 | 05 | 01 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 05 | 05 | 01 | 03 |
|00 0001 05|05 0006 05|05 0011 04|04 0018 01|01 0019 00|
| 05 | 04 | 01 | 01 | 01 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 04 | 01 | 01 | 01 |
|00 0000 02|02 0005 04|04 0010 03|03 0023 02|02 0024 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0004 02|02 0009 05|05 0014 03|03 0019 01|01 0024 00|
| 04 | 05 | 01 | 01 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 05 | 01 | 01 | 02 |
|00 0003 02|02 0008 05|05 0013 01|01 0018 01|01 0023 00|
| 02 | 05 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 05 | 02 | 04 | 03 |
|00 0002 04|04 0007 02|02 0012 02|02 0017 04|04 0022 00|
| 03 | 05 | 05 | 01 | 03 |
+--------------+--------------+--------------+--------------+--------------+
```

```
| 03 | 05 | 05 | 01 | 03 |
|00 0001 05|05 0006 05|05 0011 04|04 0016 02|02 0021 00|
| 05 | 04 | 01 | 01 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 04 | 01 | 01 | 04 |
|00 0000 02|02 0005 04|04 0010 03|03 0015 05|05 0020 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0004 02|02 0009 05|05 0014 03|03 0019 01|01 0024 00|
| 04 | 05 | 01 | 01 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 05 | 01 | 01 | 02 |
|00 0021 02|02 0008 05|05 0013 01|01 0018 01|01 0023 00|
| 03 | 05 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 05 | 02 | 04 | 03 |
|00 0022 04|04 0007 02|02 0012 02|02 0017 04|04 0002 00|
| 03 | 05 | 05 | 01 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 05 | 05 | 01 | 02 |
|00 0001 05|05 0006 05|05 0011 04|04 0016 02|02 0003 00|
| 05 | 04 | 01 | 01 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 04 | 01 | 01 | 04 |
|00 0000 02|02 0005 04|04 0010 03|03 0015 05|05 0020 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0000 05|05 0001 03|03 0002 02|02 0003 04|04 0004 00|
| 02 | 05 | 04 | 02 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 05 | 04 | 02 | 02 |
|00 0005 04|04 0006 05|05 0007 05|05 0008 05|05 0009 00|
| 04 | 05 | 02 | 05 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 05 | 02 | 05 | 05 |
|00 0010 01|01 0011 05|05 0012 02|02 0013 01|01 0015 00|
| 03 | 04 | 02 | 01 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 01 | 03 |
|00 0023 01|01 0018 01|01 0017 04|04 0016 02|02 0021 00|
| 02 | 01 | 04 | 01 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 01 | 04 | 01 | 04 |
|00 0024 01|01 0019 03|03 0022 03|03 0014 05|05 0020 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+
```

```
352
353 <deb>
354 +-------------+-------------+-------------+-------------+-------------+
355 | 00 | 00 | 00 | 00 | 00 |
356 |00 0000 05|05 0001 03|03 0002 02|02 0003 04|04 0004 00|
357 | 02 | 05 | 04 | 02 | 02 |
358 +-------------+-------------+-------------+-------------+-------------+
359 | 02 | 05 | 04 | 02 | 02 |
360 |00 0005 04|04 0006 05|05 0007 05|05 0008 05|05 0009 00|
361 | 04 | 05 | 02 | 05 | 05 |
362 +-------------+-------------+-------------+-------------+-------------+
363 | 04 | 05 | 02 | 05 | 05 |
364 |00 0010 01|01 0011 05|05 0012 02|02 0013 01|01 0015 00|
365 | 03 | 04 | 02 | 01 | 03 |
366 +-------------+-------------+-------------+-------------+-------------+
367 | 03 | 04 | 02 | 01 | 03 |
368 |00 0014 01|01 0016 01|01 0017 04|04 0018 01|01 0019 00|
369 | 05 | 02 | 04 | 01 | 01 |
370 +-------------+-------------+-------------+-------------+-------------+
371 | 05 | 02 | 04 | 01 | 01 |
372 |00 0020 04|04 0021 03|03 0022 03|03 0023 02|02 0024 00|
373 | 00 | 00 | 00 | 00 | 00 |
374 +-------------+-------------+-------------+-------------+-------------+
375
376 <deb>
377 +-------------+-------------+-------------+-------------+-------------+
378 | 00 | 00 | 00 | 00 | 00 |
379 |00 0000 05|05 0001 03|03 0002 02|02 0003 04|04 0004 00|
380 | 02 | 05 | 04 | 02 | 02 |
381 +-------------+-------------+-------------+-------------+-------------+
382 | 02 | 05 | 04 | 02 | 02 |
383 |00 0005 04|04 0006 05|05 0007 05|05 0008 05|05 0009 00|
384 | 04 | 05 | 02 | 05 | 05 |
385 +-------------+-------------+-------------+-------------+-------------+
386 | 04 | 05 | 02 | 05 | 05 |
387 |00 0010 01|01 0011 05|05 0012 02|02 0013 01|01 0014 00|
388 | 03 | 04 | 02 | 01 | 03 |
389 +-------------+-------------+-------------+-------------+-------------+
390 | 03 | 04 | 02 | 01 | 03 |
391 |00 0023 01|01 0018 01|01 0017 04|04 0016 02|02 0021 00|
392 | 02 | 01 | 04 | 01 | 04 |
393 +-------------+-------------+-------------+-------------+-------------+
394 | 02 | 01 | 04 | 01 | 04 |
395 |00 0024 01|01 0019 03|03 0022 03|03 0015 05|05 0020 00|
396 | 00 | 00 | 00 | 00 | 00 |
397 +-------------+-------------+-------------+-------------+-------------+
398
399 <deb>
400 +-------------+-------------+-------------+-------------+-------------+
401 | 00 | 00 | 00 | 00 | 00 |
402 |00 0000 05|05 0001 03|03 0002 02|02 0003 04|04 0004 00|
403 | 02 | 05 | 04 | 02 | 02 |
404 +-------------+-------------+-------------+-------------+-------------+
405 | 02 | 05 | 04 | 02 | 02 |
```

```
|00 0005 04|04 0006 05|05 0007 05|05 0008 05|05 0009 00|
| 04 | 05 | 02 | 05 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 05 | 02 | 05 | 05 |
|00 0010 01|01 0011 05|05 0012 02|02 0013 01|01 0014 00|
| 03 | 04 | 02 | 01 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 01 | 03 |
|00 0015 01|01 0016 01|01 0017 04|04 0018 01|01 0019 00|
| 05 | 02 | 04 | 01 | 01 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 02 | 04 | 01 | 01 |
|00 0020 04|04 0021 03|03 0022 03|03 0023 02|02 0024 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0000 05|05 0001 03|03 0022 03|03 0021 04|04 0004 00|
| 02 | 05 | 04 | 02 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 05 | 04 | 02 | 02 |
|00 0005 04|04 0006 05|05 0007 05|05 0008 05|05 0009 00|
| 04 | 05 | 02 | 05 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 05 | 02 | 05 | 05 |
|00 0010 01|01 0011 05|05 0012 02|02 0013 01|01 0015 00|
| 03 | 04 | 02 | 01 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 01 | 03 |
|00 0014 01|01 0016 01|01 0017 04|04 0018 01|01 0019 00|
| 05 | 02 | 04 | 01 | 01 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 02 | 04 | 01 | 01 |
|00 0020 04|04 0003 02|02 0002 03|03 0023 02|02 0024 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0000 05|05 0001 03|03 0022 03|03 0021 04|04 0004 00|
| 02 | 05 | 04 | 02 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 05 | 04 | 02 | 02 |
|00 0005 04|04 0006 05|05 0007 05|05 0008 05|05 0009 00|
| 04 | 05 | 02 | 05 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 04 | 05 | 02 | 05 | 05 |
|00 0010 01|01 0011 05|05 0012 02|02 0013 01|01 0014 00|
| 03 | 04 | 02 | 01 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 01 | 03 |
```

```
|00 0015 01|01 0016 01|01 0017 04|04 0018 01|01 0019 00|
| 05 | 02 | 04 | 01 | 01 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 02 | 04 | 01 | 01 |
|00 0020 04|04 0003 02|02 0002 03|03 0023 02|02 0024 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0024 02|02 0023 03|03 0010 04|04 0005 02|02 0000 00|
| 01 | 01 | 01 | 04 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 01 | 01 | 04 | 05 |
|00 0019 01|01 0018 04|04 0011 05|05 0006 05|05 0001 00|
| 03 | 01 | 05 | 05 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 05 | 05 | 03 |
|00 0022 04|04 0017 02|02 0012 02|02 0007 04|04 0002 00|
| 03 | 04 | 02 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 05 | 02 |
|00 0015 01|01 0016 01|01 0013 05|05 0008 02|02 0003 00|
| 05 | 02 | 01 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 02 | 01 | 05 | 04 |
|00 0020 04|04 0021 03|03 0014 05|05 0009 02|02 0004 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0024 02|02 0023 03|03 0010 04|04 0005 02|02 0000 00|
| 01 | 01 | 01 | 04 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 01 | 01 | 04 | 05 |
|00 0019 01|01 0018 04|04 0011 05|05 0006 05|05 0001 00|
| 03 | 01 | 05 | 05 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 05 | 05 | 03 |
|00 0022 04|04 0017 02|02 0012 02|02 0007 04|04 0002 00|
| 03 | 04 | 02 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 04 | 02 | 05 | 02 |
|00 0014 01|01 0016 01|01 0013 05|05 0008 02|02 0003 00|
| 05 | 02 | 01 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 02 | 01 | 05 | 04 |
|00 0020 04|04 0021 03|03 0015 05|05 0009 02|02 0004 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

```

```
<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0024 02|02 0023 03|03 0002 02|02 0003 04|04 0020 00|
| 01 | 01 | 04 | 02 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 01 | 04 | 02 | 05 |
|00 0019 01|01 0018 04|04 0017 01|01 0016 01|01 0015 00|
| 03 | 01 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 02 | 04 | 03 |
|00 0014 01|01 0013 02|02 0012 05|05 0011 01|01 0010 00|
| 05 | 05 | 02 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 05 | 02 | 05 | 04 |
|00 0009 05|05 0008 05|05 0007 05|05 0006 04|04 0005 00|
| 02 | 02 | 04 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 02 | 04 | 05 | 02 |
|00 0004 04|04 0021 03|03 0022 03|03 0001 05|05 0000 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0024 02|02 0023 03|03 0002 02|02 0003 04|04 0020 00|
| 01 | 01 | 04 | 02 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 01 | 04 | 02 | 05 |
|00 0019 01|01 0018 04|04 0017 01|01 0016 01|01 0014 00|
| 03 | 01 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 02 | 04 | 03 |
|00 0015 01|01 0013 02|02 0012 05|05 0011 01|01 0010 00|
| 05 | 05 | 02 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 05 | 02 | 05 | 04 |
|00 0009 05|05 0008 05|05 0007 05|05 0006 04|04 0005 00|
| 02 | 02 | 04 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 02 | 04 | 05 | 02 |
|00 0004 04|04 0021 03|03 0022 03|03 0001 05|05 0000 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0024 02|02 0023 03|03 0022 03|03 0021 04|04 0020 00|
| 01 | 01 | 04 | 02 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 01 | 04 | 02 | 05 |
|00 0019 01|01 0018 04|04 0017 01|01 0016 01|01 0015 00|
```

```
| 03 | 01 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 02 | 04 | 03 |
|00 0014 01|01 0013 02|02 0012 05|05 0011 01|01 0010 00|
| 05 | 05 | 02 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 05 | 02 | 05 | 04 |
|00 0009 05|05 0008 05|05 0007 05|05 0006 04|04 0005 00|
| 02 | 02 | 04 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 02 | 04 | 05 | 02 |
|00 0004 04|04 0003 02|02 0002 03|03 0001 05|05 0000 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+

<deb>
+--------------+--------------+--------------+--------------+--------------+
| 00 | 00 | 00 | 00 | 00 |
|00 0024 02|02 0023 03|03 0022 03|03 0021 04|04 0020 00|
| 01 | 01 | 04 | 02 | 05 |
+--------------+--------------+--------------+--------------+--------------+
| 01 | 01 | 04 | 02 | 05 |
|00 0019 01|01 0018 04|04 0017 01|01 0016 01|01 0014 00|
| 03 | 01 | 02 | 04 | 03 |
+--------------+--------------+--------------+--------------+--------------+
| 03 | 01 | 02 | 04 | 03 |
|00 0015 01|01 0013 02|02 0012 05|05 0011 01|01 0010 00|
| 05 | 05 | 02 | 05 | 04 |
+--------------+--------------+--------------+--------------+--------------+
| 05 | 05 | 02 | 05 | 04 |
|00 0009 05|05 0008 05|05 0007 05|05 0006 04|04 0005 00|
| 02 | 02 | 04 | 05 | 02 |
+--------------+--------------+--------------+--------------+--------------+
| 02 | 02 | 04 | 05 | 02 |
|00 0004 04|04 0003 02|02 0002 03|03 0001 05|05 0000 00|
| 00 | 00 | 00 | 00 | 00 |
+--------------+--------------+--------------+--------------+--------------+



Solving: end of time
```