Lic. Info. & MI 2ème année — Université d'Orléans
SLA4IF02 - Prog. fonctionnelle — Année 2018-2019

Devoir de programmation : le jeu du Hashiwo kakero.
À rendre le 5/5/2019

Le problème est le suivant : pour promouvoir l'unité et le progrès économique et technologique de l'archipel sur lequel vous régnez, vous proposez la construction d'un réseau de ponts permettant à vos sujets de circuler librement aux quatre coins du territoire. Dans ce but, vous devrez prendre en compte les besoins très particuliers de chaque île et respecter certaines contraintes techniques. Votre grande sagesse sera-t-elle à même de résoudre ce problème ?

Présentation du jeu :

Le jeu s'appelle *Hashiwo kakero* (qu'on peut traduire par "construire des ponts") est un puzzle inventé et publié par le célèbre magazine japonais *Nikoli*, qui est aussi à l'origine notamment du *Sudoku* et du *Kakuro*. Le but de ce projet est d'élaborer un programme capable de résoudre les puzzles de ce type.

Chaque puzzle se présente sous la forme d'une grille d'une certaine taille et bien souvent de forme carrée. Sur certains croisements de cette grille, des ronds sont placés pour matérialiser les îles de l'archipel. L'importance de chaque île est notée à l'intérieur du rond concerné.

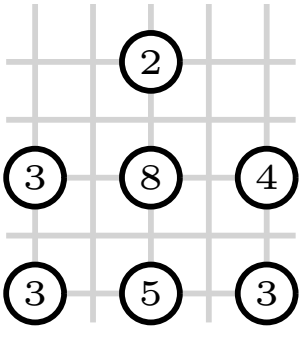
Pour résoudre le puzzle vous devez former un réseau connexe (i.e. qui permet à un habitant de n'importe quelle île de rejoindre n'importe quelle autre île) en plaçant des ponts. Ceux-ci doivent respecter quelques contraintes :

- un pont doit partir d'une île et arriver à une autre sans faire de virage ;
- chaque pont doit être horizontal ou vertical ;
- entre deux îles données, il peut y avoir deux, un ou aucun pont ;
- un pont ne peut croiser ni une île, ni un autre pont.

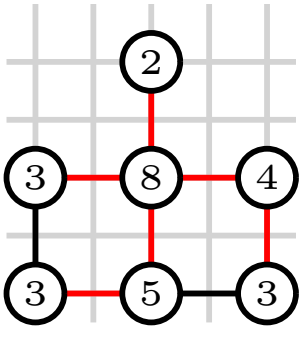
Chaque île doit être reliée au réseau par un nombre de ponts égal à son importance.

La solution de tout puzzle est unique, et vous pouvez considérer (car c'est le cas en général) que deux îles ne sont jamais placées à deux croisements consécutifs dans une direction horizontale ou verticale.

Ci-dessous, un exemple de puzzle et sa solution. Le trait noir représente un pont unique et le trait rouge représente l'existence de deux ponts.



puzzle 1



sa solution

Méthodes de résolution :

Il existe quelques astuces pour résoudre ce genre de puzzle. Ce sont les mêmes astuces qui sont utiles pour un humain, et qui servent à construire un algorithme de résolution par ordinateur.

1. Tout d'abord, il y a un grand nombre de situations qui obligent la construction de ponts dans certaines directions. Lorsqu'une île est dans le coin ou sur le bord de la carte, les possibilités sont limitées. Aussi, une île très importante, même au beau milieu de l'archipel nécessitera forcément au moins un pont dans chacune des quatre directions possibles.

   Au fur et à mesure que vous construisez des ponts, les situations se débloquent pour les autres îles. Un pont peut bloquer l'accès entre deux îles et ainsi conduire à une situation dont la solution est évidente. Une bonne partie des puzzles, et même des puzzles entiers, peuvent être résolus en appliquant cette seule stratégie en cascade.
2. Par contre, pour les puzzles plus difficiles, il faut se servir du fait que les îles doivent toutes être reliées entre elles. Ainsi, s'il ne reste plus qu'un pont à construire à partir d'une certaine île, mais qu'il y a deux destinations possibles, vous pouvez en éliminer une si vous savez que cela formera un sous-réseau qui ne pourra jamais être relié au reste.

Programmation :

A priori, votre solveur consiste à appliquer successivement la première astuce donnée ci-dessus à chaque île, puis à recommencer s'il y a eu des changements du réseau mais que le puzzle n'a pas encore été résolu. Si la première astuce ne produit aucun changement, vous pouvez tenter la deuxième pour débloquer la situation. Vous pouvez aussi y ajouter vos propres techniques de résolution.

Types de données : Le puzzle doit être représenté par une liste de couples, dont la première composante est une coordonnée et la deuxième est l'importance de l'île placée à cette coordonnée.

```
1 type coordinate = int * int ; ;
2 type importance = int ; ;
3 type puzzle = ( coordinate * importance ) list ; ;
```

La solution sera une matrice (représentée par une liste de listes) de ce qui se trouve à chaque croisement de la grille. Chaque sous-liste représente une ligne de la solution avec le croisement le plus à gauche en premier. Les sous-listes doivent être ordonnées de haut en bas.

```
1 type bridge = { isVertical : bool ; isDoubled : bool } ; ;
2 type cell = | Nothing | Island of importance | Bridge of bridge ; ;
3 type solution = cell list list ; ;
```

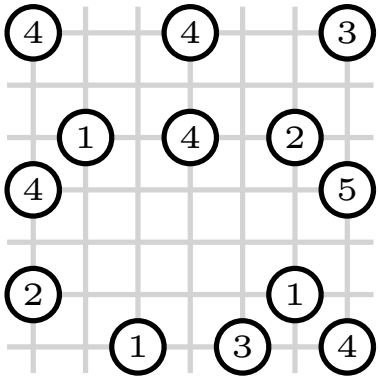
Avec cette représentation, un croisement qui comporte deux ponts horizontaux est représenté par la valeur `Bridge {isVertical = false; isDoubled = true}`.

En plus de ces deux types, vous pourrez définir d'autres types si besoin.

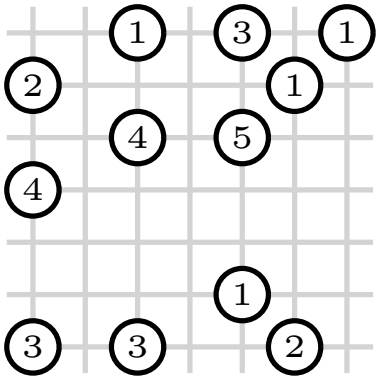
On demande de :

1. définir une fonction qui applique la première astuce à chaque île du puzzle ;
2. définir la fonction qui applique la deuxième astuce à l'île qui est la plus adaptée.
   En effet, en regardant les exemples, l'application de la deuxième astuce donne plus ou moins de réussite selon l'île choisie.
3. À l'aide des fonctions précédentes et éventuellement d'autres fonctions qui représentent vos propres techniques de résolution, définir la fonction `solve : puzzle -> solution` qui résout un puzzle donné ;
4. tester votre solveur sur les exemples donnés ci-après, des puzzles tirés d'autres sources ou construits par vous-même.

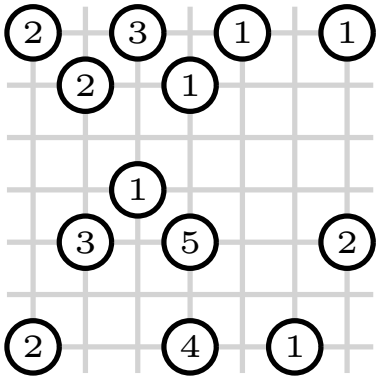
Exemples de puzzles



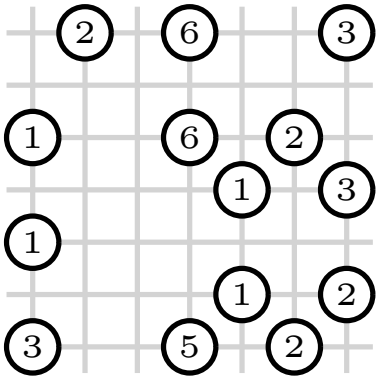
puzzle 2



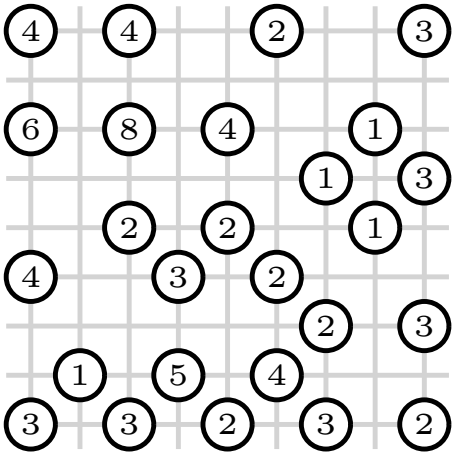
puzzle 3



puzzle 4



puzzle 5



puzzle 6

Travail demandé :

Le projet est à réaliser par binôme. Il sera accompagné d'un dossier contenant impérativement la description des choix faits, la description des types et des fonctions. Pour chaque fonction, on donnera impérativement l'interface complète (dans le code en commentaire et dans le rapport pour les fonctions présentées).

Le dossier fournira également des cas de tests accompagnés des résultats attendus et retournés.

Le dossier doit également comporter un mode d'emploi de l'outil développé. Il doit aussi indiquer la répartition du travail dans le groupe. Rapport et programme ".ml" (suffisamment commenté) devront être archivés dans un "**(.zip)**" et déposés sur Celene. Un dépôt sera créé pour ce devoir. Vous devrez impérativement indiquer dans votre dossier le nom de votre **chargé de TP**.