INF421 PROGRAMMING PROJECT
HAMILTONIAN PATHS AND RIKUDO SOLVER

VINCENT PILAUD
vincent.pilaud@lix.polytechnique.fr

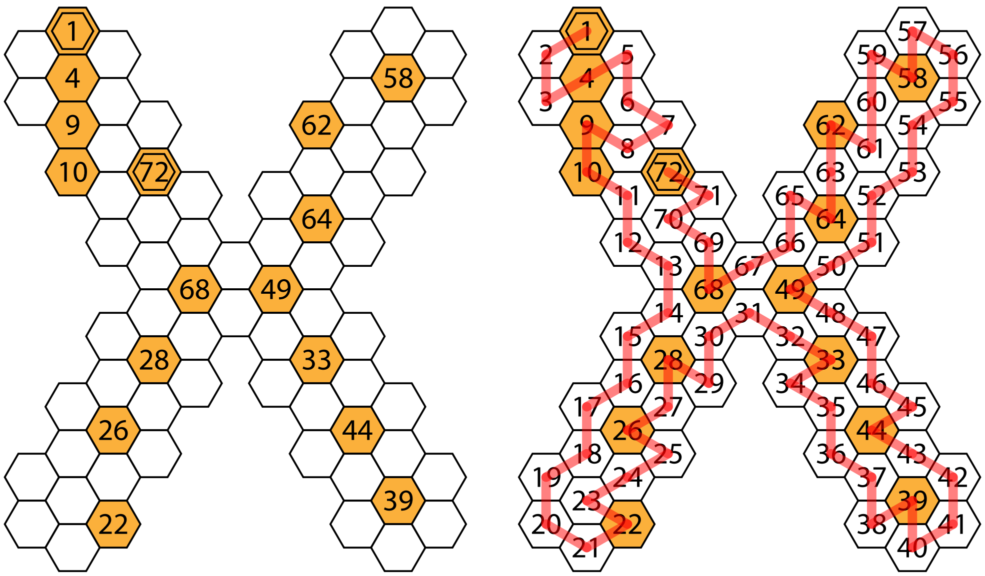

Figure 1. A riXkudo problem (left) with a solution (right).

1. Rikudo

Rikudo is a recent game[1] whose rules are as follows (see http://www.rikudo.fr). We are given a tiling in the plane, with n hexagonal tiles, where some tiles have been labeled by numbers in $\{1, \dots, n\}$ and some edges have been marked by a diamond. The goal of the player is to find a path of adjacent cells, starting from the cell numbered 1 and ending in the cell numbered n, in such a way that if a cell is numbered by i, it is visited at step i, and if an edge is marked by a diamond, then its two adjacent cells are visited consecutively. In other words, the player looks for a sequence $c_1, c_2, \dots, c_n$ so that

- for all i, the cells c_i and c_{i+1} are adjacent cells (adjacent rule),
- for all i, the cell c_i is either empty or labeled by i (labeling rule),
- if the cells c_i and c_j are adjacent along an edge marked with a diamond, then $i = j + 1$ or $i = j - 1$ (diamond rule).

This game is illustrated on Figure 2, as it appears on the website http://www.rikudo.fr. A more appropriate version of the game is illustrated in Figure 1.

2. Hamiltonian path

We consider a directed graph $G = (V, A)$ with vertex set V and oriented edge set $A \subseteq V^2$. A *Hamiltonian path* in G is a bijective ordering $v_1, \dots, v_n$ of the vertices such that $(v_i, v_{i+1}) \in A$ for all i. Finding a Hamiltonian path in a graph is a famous NP-complete problem. In this section, we discuss two computational methods to find a Hamiltonian path in a small graph G.

[1]Invented in 2015 in the physics department of Polytechnique!

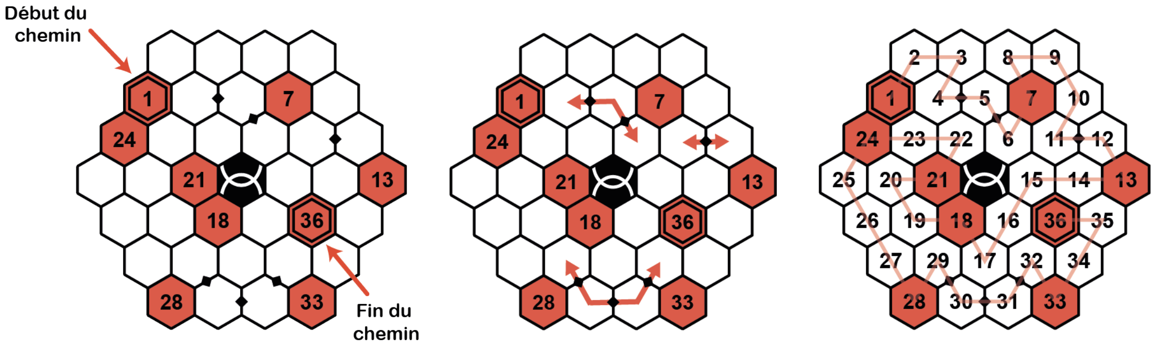

FIGURE 2. Rikudo solving: A rikudo grid (left), its diamond rules (middle), and its solution (right). Images from http://www.rikudo.fr

2.1. **Using a SAT solver.** Our first method is to transform the Hamiltonian path problem to a SAT problem. An instance of the *SAT problem* (SAT for satisfiability) consists in deciding whether a formula over boolean variables $x_1, \dots, x_m$ connected with operators of conjunction $\wedge$ (for "and"), disjunction $\vee$ (for "or") and negation $\neg$ (for "not") is satisfiable. The formula is in *conjunctive normal form* when it is written as a conjunction ($\wedge$) of *clauses*, each of which is a disjunction ($\vee$) of literals (x_i or $\neg x_i$).

Given a directed graph $G = (V, A)$, a source vertex $s \in V$ and a target vertex $t \in V$, we can transform the problem to find a Hamiltonian path in G connecting s to t into a SAT problem as follows. For each vertex $v \in V$ and each index $i \in [n]$, we consider a boolean variable $x_{i,v}$ which remembers if the ith vertex of the path is v. These variables should satisfy the following constraints:

- each vertex $v \in V$ appears precisely once in the path,
- each index $i \in [n]$ is occupied precisely once,
- consecutive vertices along the path are adjacent in the graph G,
- the first vertex should be the source s and the last vertex should be the target t.

To solve a SAT problem, we can use libraries already developed for java, for example Sat4j (see http://www.sat4j.org for the documentation). A minimal example of use of this library is provided at http://www.lix.polytechnique.fr/~pilaud/enseignement/TP/DIX/INF421/1718/.

Task 1. *Write a method that translates a Hamiltonian path problem to a SAT solver and solves it using the library of your choice. Evaluate the computation time on examples such as complete graphs, cycles, grid graphs, ...*

Remark 1. (i) Conversely, any SAT problem can be transformed into a Hamiltonian path problem. Implementing this classical reduction is a good optional mini-project.
(ii) Adapt the SAT problem to find a Hamiltonian cycle rather than a Hamiltonian path?

2.2. **Backtracking algorithm.** Our second method to solve the Hamiltonian path problem is to design an adapted backtracking algorithm. The idea is simple: start from the source vertex s, and at each step consider all vertices that are adjacent to the current vertex and not yet visited. If such a vertex could be added and completed to a Hamiltonian path to t, then return this path ; otherwise, remove this vertex and try the next vertex.

Task 2. *Implement this backtracking method to solve the Hamiltonian path problem. Evaluate the computation time on examples such as complete graphs, cycles, grid graphs, ...*

Remark 2. (i) Adapt your algorithm so that it counts the number of Hamiltonian paths from s to t in G. How many Hamiltonian paths are there between two opposite corners of a grid graph of size $n \times n$ for small values of n? (Can you explain this phenomenon?) How many solutions do you find for the problem of Figure 1?
(ii) Adapt your algorithm for a given k, it returns an empty path if there are less than k paths between s and t in G, and one of these Hamiltonian paths if there are more than k.

3. Adding constraints

Coming back to the initial Rikudo problem, we now consider a directed graph $G = (V, A)$ with vertex set V and oriented edge set $A \subseteq V^2$ together with

- a partial injective map $\lambda : [n] \to V$ (partial means that some $\lambda(i)$ are not defined), representing the labels, and
- a collection $E_\diamond \subseteq \binom{V}{2}$ of pairs of vertices, representing the diamonds.

A Hamiltonian path $s = v_1, \dots, v_n = t$ connecting the source s to the target t is *valid* if $v_i = \lambda(i)$ for all i in the domain of definition of λ, and $\{v_i, v_j\} \in E_\diamond$ implies $i = j + 1$ or $i = j - 1$.

Task 3. *Adapt your implementations of Tasks 1 and 2 to look for a valid Hamiltonian path. Compare the two algorithms on examples such as complete graphs, cycles, grid graphs, ...*

Remark 3. To complete your project, you can consider additional constraints as those described on the page http://www.rikudo.fr/v2.0/dossier-presse, or even invent new constraints and rules. Try to adapt both the SAT and the backtracking algorithms to satisfy these contraints.

4. Creating puzzles

The goal of the remaining of the project is to create rikudo puzzles. We require that the puzzles have a unique solution and we will try to create puzzles with nice shapes.

4.1. **From a graph.** At the moment, we start from a graph $G = (V, A)$ with a source vertex s and a target vertex t. Our goal is to output a rikudo instance (ie. a partial labeling λ and a diamond collection $E_\diamond$, together with all other additional constraints you might have invented in Remark 3) which admits a unique solution but also provides the minimal information for this solution.

Task 4. *Write a method that decides, given a partial labeling λ and a diamond collection $E_\diamond$, whether this information*

- *determines a unique solution to the rikudo problem,*
- *is minimal for this property.*

Is the information provided in Figure 1 minimal? How many minimal information would have lead to precisely the same solution?

Task 5. *Given a graph G with a source vertex s and a target vertex t, design one or more strategy to create rikudo instances on G with precisely the minimal information to admit a unique solution. Use your code to provide instances on the graphs of Figures 1 and 2. Up to which value of n can you create a rikudo instance on the $n \times n$ grid?*

Remark 4. To complete your project, you can implement functions that output nice representations of your rikudo problems, like the ones in Figures 1 and 2. A basic graphical interface is provided at http://www.lix.polytechnique.fr/~pilaud/enseignement/TP/DIX/INF421/1718/.

4.2. **From a picture.** We now want to create puzzles with nice shapes. As you probably have observed, it is quite tedious to create a puzzle with a nice shape.

Task 6. *Given a black and white picture and a chosen resolution, provide a method that creates a rikudo puzzle whose underlying graph is a subset of the grid that looks similar to the initial picture.*

Some binary images and a class to manipulate binary images are provided at http://www.lix.polytechnique.fr/~pilaud/enseignement/TP/DIX/INF421/1718/.

Remark 5. In general, feel free to invent any extension of the tasks presented in this project (why not a web interface... even if this does not enter anymore in the INF421 phylosophy). Be imaginative, you are a puzzle creator!