INP ENSEEIHT

Projet : Graphes et donjons

Motivation

Vous participez à la création d'un jeu vidéo. Le concept est simple : le joueur doit parvenir à trouver un trésor caché dans un donjon. Ce donjon est composé d'un ensemble de salles dont une est l'entrée et une est la salle du trésor, ainsi que de couloirs entre certaines salles. Un couloir peut être soit pratiquable, soit bloqué par une porte verrouillée d'une certaine couleur. Les clés sont disposées dans les salles du donjon. Dès qu'une clé d'une certaine couleur est possédée, elle permet d'ouvrir toutes les portes de la même couleur. Il est possible qu'une même clé soit disponible dans plusieurs salles.

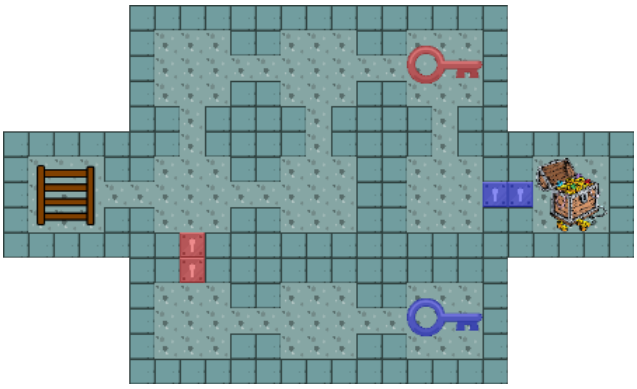

FIGURE 1 – Un exemple de donjon. Les murs sont en carré, le sol en bleu. L'échelle représente l'entrée, le trésor représente la sortie. Chaque clé permet d'ouvrir la porte de sa couleur.

Un de vos collègues est chargé de concevoir de nombreux donjons différents. Il vous a donné le programme avec lequel il crée les donjons : double-cliquez simplement sur `creationDonjon.html` pour ouvrir une page dans votre navigateur et avoir accès à son interface de création de donjons. Vous pouvez notamment charger les donjons créés par votre collègue avec le bouton situé en bas de la page.

Pour pouvoir livrer le jeu dans les délais, votre collègue a créé des donjons rapidement sans s'assurer qu'ils soient faisables. Vous vous en apercevez en ouvrant `donjon_impossible3s1c0i.ml` : le chemin est bloqué par une porte rouge, mais la clé rouge est située derrière cette même porte. Plutôt que de vérifier les donjons un à un à la main, vous décidez de coder un programme effectuant cette vérification. Vous réalisez que la structure de graphes se prête bien à la modélisation d'un donjon : les salles en sont les sommets tandis que les couloirs représentent ses arêtes. De plus, comme vous aimez le challenge, vous vous mettez en tête d'écrire ce programme en CAML.

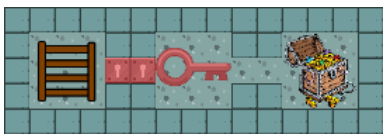

FIGURE 2 – Un exemple de donjon non faisable. Le joueur ne peut pas progresser jusqu'au trésor car la clé rouge est située après la porte rouge.

Partie 1 : Une première intuition pour les donjons à clés

Dans cette partie, on souhaite implémenter une fonction `faisable g vi vf cles` renvoyant un booléen qui indique s'il existe un chemin dans le graphe `g` permettant d'aller d'un sommet `vi` à un sommet `vf` connaissant la liste `cles` des couples (clé, portes ouvertes par cette clé). Puis, si un donjon est faisable, on souhaite obtenir un parcours menant de l'entrée à la sortie, en passant par les clés nécessaires via une fonction `parcours g vi vf cles`.

L'idée est la suivante : si il existe un chemin entre `vi` et `vf`, alors le donjon est faisable, sinon on regarde l'ensemble des clés accessibles, c'est-à-dire les clés atteignables à partir de `vi`. On retire ces clés de `cles` et on ajoute à `g` les arêtes correspondant aux portes qu'elles ouvrent. Si aucune clé n'est accessible, alors le donjon n'est pas faisable, sinon on appelle alors récursivement `faisable` sur le nouveau graphe enrichi des arêtes que l'on vient d'ajouter, sur les mêmes positions initiale et finale et sur la liste des clés restantes.

Attention : lors de l'ajout d'arêtes dans un graphe, le graphe est modifié de façon définitive. Afin de ne pas avoir à le réinitialiser chaque fois que vous relancez un test, une fonction `true_copy g` vous est proposée. En effet, la fonction `copy` de la librairie de graphe ne copie pas les noms des sommets et des arêtes, seulement les labels.

1. Complétez la fonction `v_reach g v` pour qu'elle renvoie la liste des sommets de `g` qui sont atteignables depuis `v`.
2. Complétez la fonction `cles_accessibles g vi cles` pour qu'elle renvoie la liste des couples (sommet contenant une clé, liste des portes ouvertes par cette clé) telle que les sommets contenant une clé soient atteignables depuis `vi`.
3. Complétez la fonction `ouvre_porte g cle` où `cle` est un couple (sommet contenant une clé, liste des portes ouvertes par cette clé) pour qu'elle retourne une copie de `g` à laquelle elle ajoute les arêtes correspondant aux portes ouvertes par la clé.
4. Complétez la fonction `faisable g vi vf cles` pour qu'elle renvoie un booléen indiquant si le donjon est faisable.
5. Complétez la fonction `parcours_salles_principales g vi vf cles` pour qu'elle renvoie la liste des salles principales par lesquelles passer dans l'ordre pour parvenir au trésor si le donjon est faisable. S'il n'est pas faisable, on renverra la liste vide `[]`. Les salles principales sont l'entrée, la liste des salles successives où prendre les clés, et la sortie.
6. Compléter la fonction `parcours g vi vf cles` pour qu'elle renvoie la liste des salles à parcourir dans l'ordre pour atteindre le trésor. On pourra s'appuyer sur le fait que les salles principales successives sont connexes et utiliser `shortest_path` sur chaque paire de salles principales successives.

Partie 2 : Un cas plus général, les donjons avec interrupteurs

Une variante de la clé est l'interrupteur. Un donjon peut contenir des couloirs bloqués par des pointes au sol qui peuvent être levées ou baissées. Lorsque les pointes sont baissées, il est possible de passer. En revanche, lorsqu'elles sont levées, le couloir est impratiquable. Un interrupteur d'une certaine couleur permet de faire alterner les pointes baissées et les pointes levées de la même couleur. Il ne disparaît pas lorsqu'il est activé et peut donc être activé plusieurs fois.

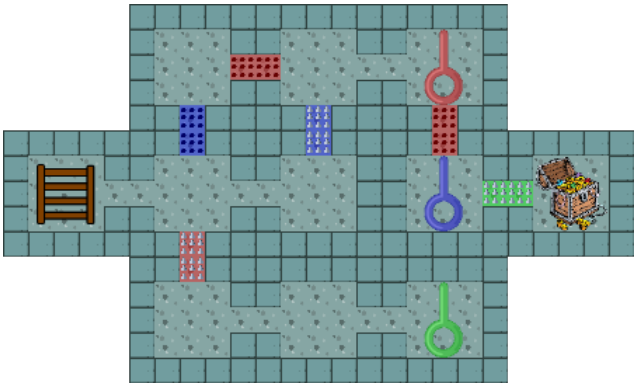

FIGURE 3 – Un exemple de donjon avec interrupteurs. Les cases colorées avec des points noirs représentent des pics baissés et sont donc pratiquables. Les cases colorées avec des pointes grises sont non pratiquables. Pour accéder au trésor, il faut aller activer l'interrupteur bleu, puis le rouge, puis le vert, puis de nouveau le rouge. Notez qu'au départ, il n'y a pas de pointes vertes baissées, l'interrupteur vert correspond donc à une clé.

Contrairement à la partie précédente où prendre une clé ne faisait qu'ajouter des arêtes et ne retirait aucun des chemins déjà accessibles, ici certains chemins disparaissent lorsque l'on active un interrupteur, tandis que d'autres apparaissent.

Notez également qu'on peut modéliser les clés à l'aide des interrupteurs, il suffit pour cela de créer un interrupteur dont toutes les pointes associées sont levées à l'origine. Ainsi, on ne considèrera plus les clés, mais uniquement des interrupteurs, qui généralisent les clés.

Dans cette partie, on cherche à savoir si un donjon est faisable et à expliciter le plus court chemin à parcourir s'il l'est. Vous êtes libres de coder les fonctions de votre choix pour y parvenir. En cas de manque d'idées, on vous propose l'approche suivante :

— Pour chacune des 2^n combinaisons d'états des n différentes couleurs d'interrupteurs disposés dans le donjon, créer une copie du graphe en lui ajoutant/retirant les arêtes correspondant aux interrupteurs activés ou non. Par exemple pour un donjon avec trois interrupteurs rouges et un interrupteur bleu, faire 4 graphes correspondant respectivement aux cas (R = 0, B = 0), (R = 0, B = 1), (R = 1, B = 0) et (R = 1, B = 1).
— Ajouter des arêtes de transition entre graphes correspondant à l'appui sur un interrupteur : par exemple pour un donjon avec trois interrupteurs rouges et un interrupteur bleu, ajouter des arêtes menant de chaque interrupteur rouge du graphe (R = i, B = j) vers l'interrupteur rouge correspondant du graphe (R = $1 - i$, B = j) et une arête menant de l'interrupteur bleu du graphe (R = i, B = j) à l'interrupteur bleu du graphe (R = i, B = $1 - j$).
— Chercher un chemin avec l'algorithme de Dijkstra dans le graphe global ainsi obtenu menant de l'entrée à l'une des sorties. On pourra si besoin créer un sommet de sortie commun et une arête reliant chaque sortie copiée à ce sommet.

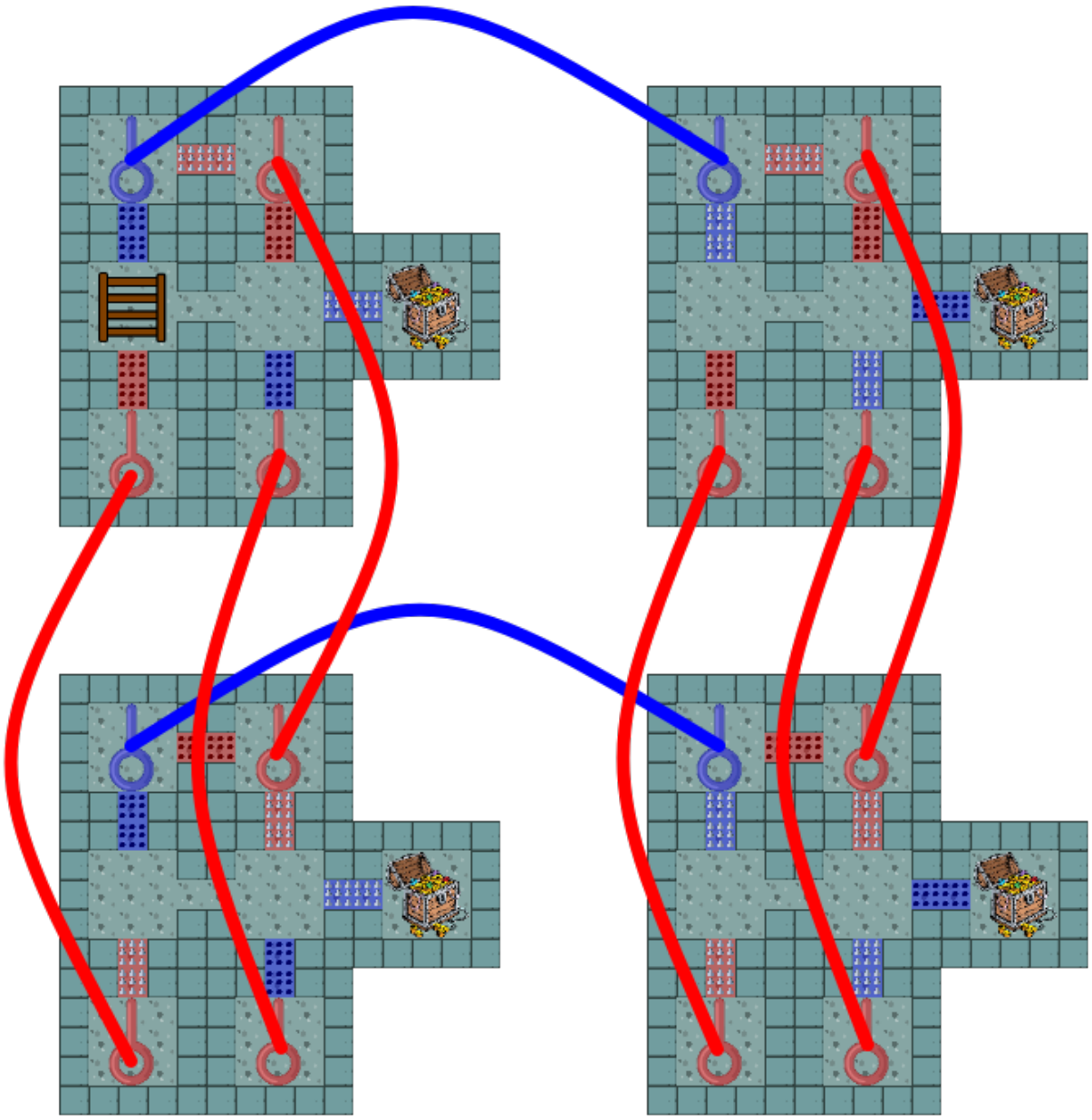

FIGURE 4 – Illustration de l'algorithme proposé pour résoudre les donjons avec interrupteurs. A partir d'un donjon initial (avec l'échelle d'entrée) considéré comme l'état du donjon lorsque aucun interrupteur n'a été activé, les graphes des 2^2 possibilités d'états des 2 couleurs d'interrupteurs sont créés, pour (R = 0, B = 0), (R = 0, B = 1), (R = 1, B = 0) et (R = 1, B = 1). Des arêtes sont ensuite ajoutées au niveau des salles contenant des interrupteurs. Notez que lorsqu'un interrupteur apparaît plusieurs fois (ici le rouge), il n'est pas utile de distinguer trois variables R1, R2 et R3 chacune égale à 0 ou 1, une unique variable R modifiée par chaque interrupteur suffit. L'équivalent en graphes est présenté à la page suivante.

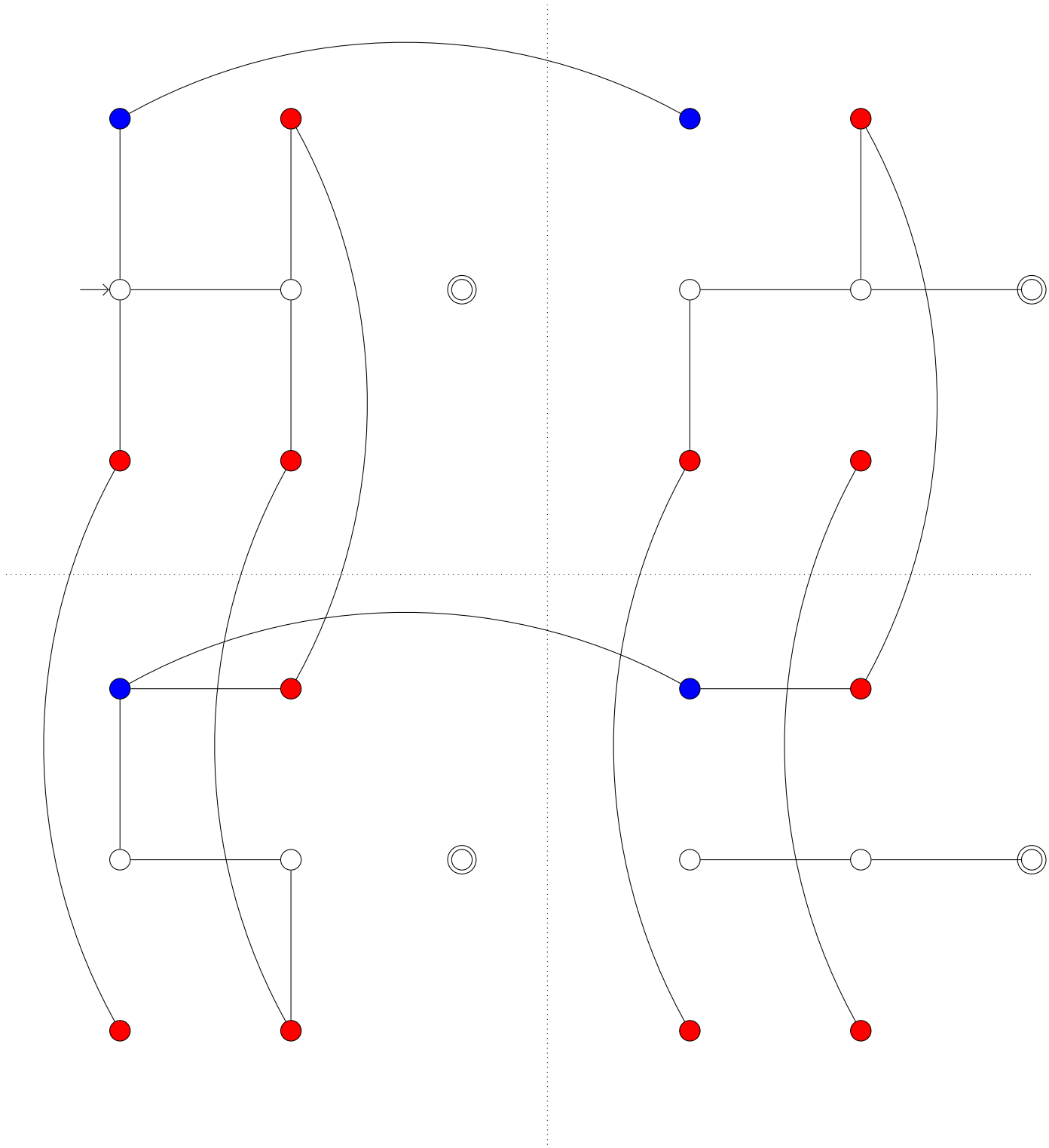

FIGURE 5 – Représentant du graphe global correspondant au donjon de la figure 4. Ce graphe comporte toutes les combinaisons d'états possibles. La flèche représente le sommet initial, les sommets avec un double cercle représentent les sommets finaux.

Evaluation du projet

Le projet sera réalisé en binôme et devra être rendu pour le **dimanche 20 janvier 23 heures** sous Moodle.

Vous devez rendre un code commenté, auquel vous ajouterez les tests de vos fonctions, ainsi qu'un rapport retranscrivant votre compréhension du projet.

Deux séances de TP, au cours desquelles vous pourrez poser des questions, sont prévues.

Barême indicatif :

— Partie 1 : 8 points

— Partie 2 : 7 points

— Rapport : 5 points