Graphes et jeux

Rappel : tester à l'aide de Utop

— `dune utop` – lance utop
— `open Tp.Trie ;;` – charge le module `Tp.Trie` i.e les fonctions contenues dans le fichier `trie.ml`
— `nouveau (fun x->x) (fun x->x) ;;` – par exemple pour tester la fonction `nouveau`
— `exit 0;;` – pour quitter

Consignes générales

— Chaque fonction demandée doit être accompagnée de son contrat.
— Vous devez également fournir des tests pour les fonctions demandées.
— Le code rendu doit être fonctionnel, il vaut mieux traiter moins de questions mais bien que toutes les questions mais avec des fonctions qui ne marchent pas correctement.
— Vous êtes libres de choisir la manière de structurer votre rendu en terme de fichiers mais vous devrez accompagner votre rendu d'un fichier readme expliquant rapidement cette structure et dans quel fichier est implémenté quelle fonction.

1 Ensemble

Dans cette partie, on cherche à représenter des ensemble à partir de listes.

▷ **Exercice 1 (Type ensemble)** *Définir le type ensemble, qui doit pouvoir contenir un type quelconque d'éléments.*

▷ **Exercice 2 (Fonctions de base)** *Écrire les fonctions ajouter, rechercher et retirer qui permettent respectivement d'ajouter, de rechercher ou de retirer un élément d'un ensemble.*

▷ **Exercice 3 (Filtrer)** *Écrire une fonction filtrer qui renvoi la liste des éléments d'un ensemble satisfaisant une condition passée en paramètre. Une condition est représentée par une fonction prenant en paramètre un élément de l'ensemble et renvoyant un booléen.*

2 Graphe

Un graphe est représenté par un ensemble de nœud et un ensemble d'arête. Dans un graphe un nœud peut être représenté par n'importe quel type, tuples, entiers, chaines de caractères, etc. Une arête est un couple de nœuds indiquant qu'ils sont reliés entre eux.

▷ **Exercice 4 (Type graphe)** *Définir le type graphe.*

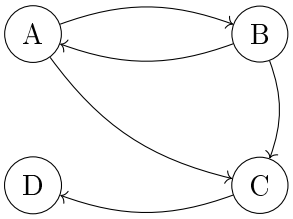

Figure 1 – Exemple de graphe

▷ **Exercice 5 (Fonctions de base)** *Écrire les fonctions permettant d'ajouter et retirer des nœuds et des arrêtes d'un graphe. Attention retirer un nœud implique de retirer toutes les arrêtes partant et allant vers ce nœud.*

3 Jeu

Un jeu est un cas particulier de graphe dans lequel les configurations possible du jeu sont représentés par un nœud composé :

- de deux booléens indiquant respectivement si la configuration est gagnante pour le joueur 1 et gagnante pour le joueur 2
- un entier, valant 1 ou 2 représentant quel joueur doit jouer le prochain coup
- une étiquette représentant l'état du jeu

Un exemple de jeu que l'on peut représenter par un graphe et le jeu des allumettes. On a un certain nombre d'allumette posé sur la table. Lorsque c'est son tour un joueur peut choisir de retirer une ou deux allumettes de la table. Le joueur qui prend la dernière allumette perd et l'autre gagne. Dans le cas de ce jeu, l'état du jeu est représenté par un entier donnant le nombre d'allumettes restantes. La figure 3 montre le graphe représentant ce jeu avec 4 allumettes. En plus du graphe, un jeu a besoin de définir une configuration initiale. Un jeu est donc un couple constitué d'une configuration initiale et d'un graphe.

▷ **Exercice 6 (Type Jeu)** *Définir le type Jeu.*

▷ **Exercice 7 (Chemin)** *Écrire une fonction qui prend en paramètre un jeu et une liste de configuration et détermine si la liste des états décrit une suite de configurations possible dans le jeu.*

▷ **Exercice 8 (Parties possibles)** *Écrire une fonction qui prend en paramètre un jeu qui se termine forcément (son graphe ne contient pas de cycle) et qui renvoie la liste des parties possibles, une partie étant une liste de configurations se terminant par la victoire de l'un ou l'autre des joueurs.*

On cherche à déterminer si le joueur 1 a une stratégie gagnante ou non. Pour cela, un algorithme simple consiste à construire petit à petit l'ensemble des configurations depuis lesquelles on peut gagner en 1 coup, puis en 2 coups, etc. Pour obtenir l'ensemble des configurations depuis lesquelles on peut gagner en n+1 coups à partir de l'ensemble des configurations depuis lesquelles gagner en n coups il suffit d'ajouter à ce dernier ensemble :

- toutes les configurations pour lesquelles le joueur 2 doit jouer telles que tous les coups du joueur 2 le fasse arriver dans cet ensemble ;

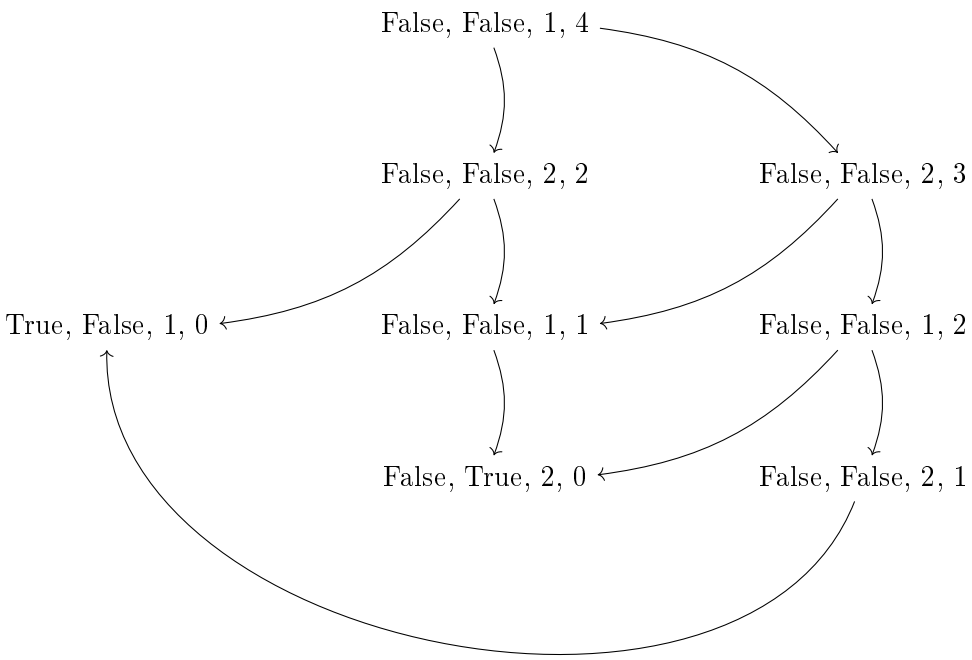

Figure 2 – Graphe représentant le jeu des allumettes

— toutes les configurations pour lesquelles le joueur 1 doit jouer telles qu'au moins un coup possible du joueur 1 le fasse arriver dans cet ensemble ;

Lorsque cet ensemble devient stable, cela veut dire que l'on a exploré toutes les configurations depuis lesquelles le joueur 1 a une stratégie pour être certain de gagner. Le joueur 1 a alors une stratégie gagnante si et seulement si la configuration initiale est dans cet ensemble.

▷ **Exercice 9 (Stratégie gagnante)** *Écrire une fonction qui prend en paramètre un jeu et qui renvoi un booléen indiquant si le joueur 1 a une stratégie gagnante ou pas.*

4 Module et interface

On remarque que tous ce qui a été fait précédemment utilise le type ensemble mais qu'il n'est pas nécessaire que ce type soit implémenté par des listes pour pouvoir utiliser des graphes. Remarque : il vaut mieux rendre cette partie

▷ **Exercice 10 (Interface Ensemble)** *Écrire une l'interface Ensemble et le module InterfaceEnsemble qui l'implémente.*

▷ **Exercice 11 (Foncteur Graphe et Jeu)** *Écrire un foncteur Graphe et un foncteur Jeu qui dépende d'un module réalisant Ensemble et qui implémente les fonctions précédentes pour les graphes et les jeux.*