TP - Jeux de Chomp

Règles du jeu de Chomp

Le jeu de Chomp se joue à l'aide d'une tablette de chocolat rectangulaire dont le coin supérieur gauche est empoisonné (Figure 1) : chaque joueur choisit à tour de rôle un carré et le mange, ainsi que tous les morceaux situés à la droite et en dessous du carré choisi. Bien évidemment, le joueur qui n'a plus d'autre choix que de manger le carré empoisonné a perdu.

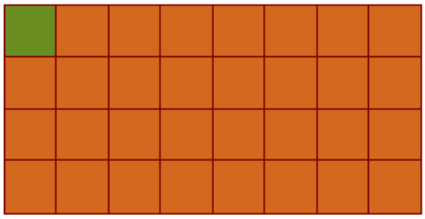


FIGURE 1 – La configuration initiale du jeu de Chomp.

Figure 2 un exemple de partie perdue par Adam (J1), qui a commencé à jouer en premier. Il s'agit d'une partie jouée sur une tablette de 3 lignes par 4 colonnes

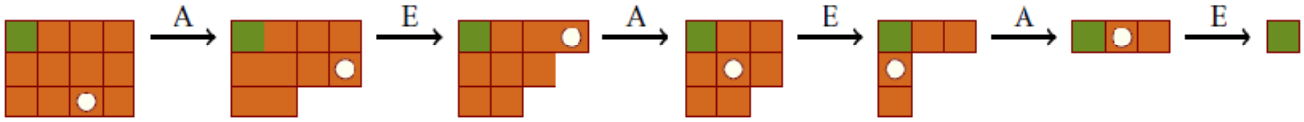
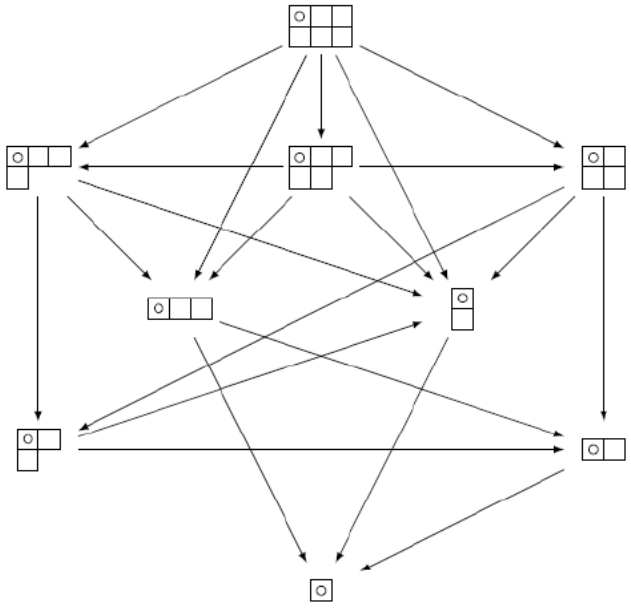


FIGURE 2 – Un exemple de partie du jeu de Chomp.

Arène du jeu de Chomp pour une tablette de 2 lignes × 3 colonnes

Nous allons commencer par un exemple simple : Tablette avec 2 lignes et 3 colonnes. Le graphe associé à ce jeu est le suivant :



Ce graphe à 9 sommets. Chaque sommet peut donc être numéroté avec un indice qui est un entier de 0 à 8. On a alors le graphe ci-dessous :

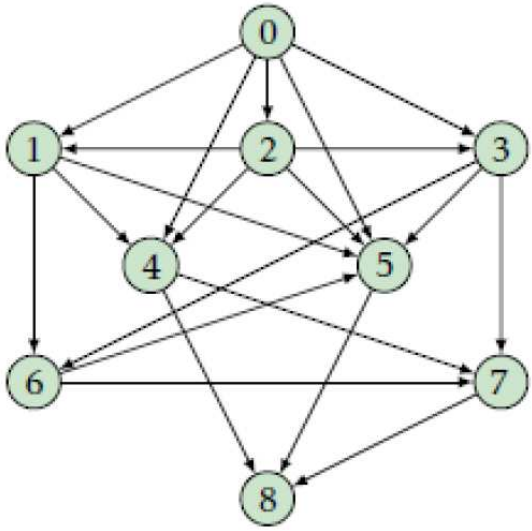


FIGURE 3 – L'arène associée au jeu de Chomp (2, 3).

Le graphe sera codé sous la forme d'un dictionnaire. Les clés sont les indices des sommets (type : int) et les Valeurs associées sont les listes des indices des sommets suivant (type : list). Ce dictionnaire `d2x3` est alors :

```
d2x3 = {0:[1,2,3,4,5] , 1:[4,5,6], 2:[1,3,4,5], 3:[5,6,7],
        4:[7,8], 5:[8], 6:[5,7], 7:[8], 8:[] }
```

Objectif du TP Déterminer une stratégie gagnante sur un jeu de Chomp 2×3 puis 3×3 voir p×q.

Résolution manuelle.

Question 1. En vous aidant de ce qui a été vu en cours sur les jeux d'accessibilité, déterminer le noyau de cette arène du jeu de Chomp avec une tablette 2×3.

Question 2. En déduire une stratégie gagnante.

Question 3. Sachant que ce jeu débute à la position d'indice 0, faut-il pour gagner prendre la main ou laisser la main ?

Résolution informatique.

Ouvrir avec Pyzo le fichier « TP_jeu_Chomp.py ». Le code Python de ce fichier contient les graphes correspondant aux jeux de Chomp 2×3 (**`d2x3`**) et 3×3 (**`d3x3`**).

Question 4. Implémenter une fonction **`sansSucceseur(G:dict)->int`** qui prend en argument le graphe **`G`** sous la forme d'un dictionnaire et renvoie l'entier **`S`** correspondant à l'indice d'un sommet sans successeur.

Question 5. Implémenter une fonction **`predecesseurs(G:dict,S :int)->list`** qui prend en argument le graphe **`G`** sous la forme d'un dictionnaire, l'indice **`S`** d'un sommet (un entier) et renvoie la liste **`Lpre`** d'entiers correspondants aux indices des prédécesseurs de **`S`**.

Question 6. Implémenter une fonction **`supprime_succ(Lsucc:list,S:int)->list`** qui prend en argument la liste de sommets successeurs **`Lsucc`** sous la forme d'une liste d'entiers, l'indice **`S`** d'un sommet (un entier) et renvoie la liste **`Lnew`** d'entiers correspondants aux indices des successeursesseurs dans laquelle le sommet **`S`** a été supprimé.

Question 7. Implémenter une fonction **`supprimeSommet(G:dict,S:int)`** qui prend en argument le graphe **`G`** sous la forme d'un dictionnaire, l'indice **`S`** d'un sommet (un entier). Qui ne renvoie rien mais qui modifie le graphe G de la manière suivante :

- ☞ Elle supprime le couple clé, valeur associé au sommet S
- ☞ Elle modifie les valeurs associées aux sommets S_{pred} prédécesseurs de S en supprimant le sommet S de la liste des successeurs des S_{pred}.

Bien sur cette fonction utilisera les fonctions des questions 5 et 6

La fonction **`noyau(G:dict)->list`** est la fonction qui va nous permettre de déterminer le noyau d'une arène. C'est-à-dire l'ensemble des positions perdantes du jeu. Elle prend donc en argument le graphe **`G`** de cette arène et renvoie la liste **`noyau`**, liste d'entier correspondant aux indices des positions perdantes du jeu.

Question 8. En vous aidant de l'algorigramme du cours sur la construction du noyau, Compléter le code de cette fonction **`noyau(G:dict)->list`**. Bien sur cette fonction utilisera les fonctions des questions précédentes.

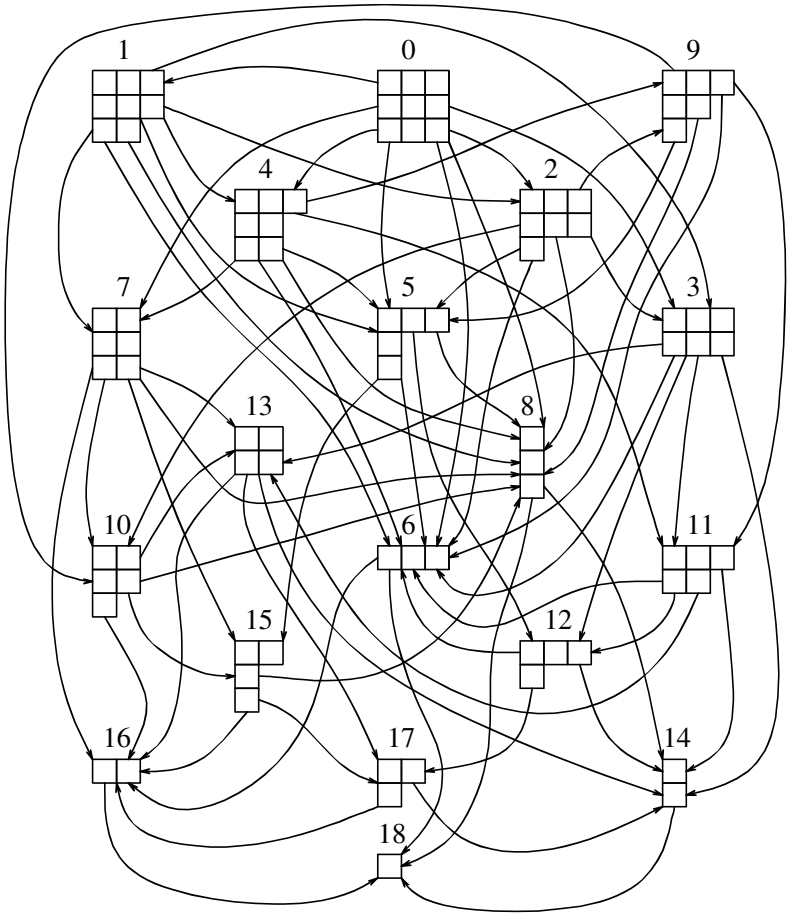
Tester la fonction avec le graphe **`d2x3`** qui doit vous renvoyer le noyau déterminé à la question 1.

Question 9. Implémenter une fonction **`strategie(G:dict)->dict`** qui prend en argument le graphe **`G`** (arène du jeu) sous la forme d'un dictionnaire, et qui renvoie une stratégie gagnante sous la forme d'un dictionnaire ou les clés sont les indices des sommets et les valeurs l'indice du sommet vers lequel il faut emmener l'adversaire pour s'assurer la victoire.

Tester la fonction avec le graphe **`d2x3`** qui doit vous renvoyer le noyau déterminé à la question 1.

Résolution pour un jeu de Chomp 3×3

On donne ci-dessous le graphe pour un jeu de Chomp 3×3. Le Graphe sous la forme d'un ductionnaire est donné dans le code Python fourni.



Question 10. Déterminer le noyau du graphe ainsi qu'une stratégie gagnante. (Manuellement ou à l'aide du code écrit dans la partie précédente).

Résolution pour un jeu de Chomp p×q

On peut constater que le graphe est déjà assez complexe pour un jeu de Chomp 3×3. Le construire avec un jeu de Chomp sur une tablette plus grande de vient difficile à faire. On vous propose donc un code (fichier « Creation Graphe et Grilles.py ») qui permet de créer ce graphe pour un jeu de Chomp p×q.

En exécutant le code Python de ce fichier et en lançant la commande **`graphe,grilles=creation_graphe_grilles(3,4)`** vous obtenez le graphe d'un jeu de Chomp. Celui-ci est stocké dans la variable **`graphe`** sous la forme d'un dictionnaire. D'autre part, la variable **`grilles`** est également un dictionnaire dont les clés sont les indices des sommets et la valeur un tableau numpy (de 0 et de 1) décrivant la position. Par exemple grilles[0] et grilles[2] vous montre les premières et deuxièmes positions de la partie décrite sur la figure 2 page 1.

Question 11. Déterminer le noyau du graphe pour un jeu de Chomp 3×4. A partir de quand Adam a-t-il commencer à mal jouer pour arriver à perdre ?

Vous avez maintenant les moyens de gagner facilement au jeu de Chomp.

Néanmoins vous constaterez qu'un jeu de Chomp p×q avec p=q=5 a un graphe de 251 sommets, pour p=q=6 on a 923 sommets, pour p=q=7 on a 3 431 sommets, pour p=q=7 on a 12 869 sommets, etc... Donc, avec l'augmentation de la taille du jeu (p et q) La complexité de l'algorithme devient rapidement trop importante pour une résolution informatique du problème.