INFO-F103/INFO-F105
Algorithmique 1 et Langages de programmation 1
Projet Arbre : *SplitGame*

Naïm Qachri – naim.qachri@ulb.ac.be

année académique 2012-2013

1 Énoncé

Pour ce projet sur les arbres, nous vous proposons d'implanter une version simplifiée et en ligne de commande du jeu *Splice*[1] [2]. Ce dernier est un puzzle-game dans lequel vous devez manipuler une structure arborescente donnée en début de jeu afin d'obtenir un autre structure arborescente qui vous est aussi indiquée au début du jeu

Pour obtenir la structure finale, vous aurez droit à réaliser un nombre limité de déplacements (défini pour le puzzle). La structure correspondra à un arbre binaire. Vous pouvez accrocher n'importe quel sous-arbre à un nœud tant que ce dernier a un fils libre. Chaque nœud est marqué d'un nombre qui lui aura été attribué en début de partie (ou au cours de la partie, voir plus bas). Le joueur choisira à chaque tour le numéro du nœud à déplacer, le numéro du nœud auquel ce dernier veut se rattacher. L'utilisateur terminera son tour en indiquant si le rattachement se fera en tant que fils gauche ou droit. Le tout sera séparé par des « - ». Nous fournissons ci-dessous un exemple de déplacement en Figure 1. La boucle d'exécution du jeu affichera les deux arbres et ensuite proposera une invite de commande dans laquelle entrer la chaîne de caractères donnant la commande de jeu suivante.

Exemples de commande utilisateur :

```
5-10-FD
12-3-FG
```

Ces commandes correspondent donc à deux déplacements. Le premier est un déplacement du nœud avec label `5` vers le fils droit du nœud avec label `10`. Le second déplacement est un déplacement du nœud avec label `12` vers le fils gauche du nœud avec label `3`.

Les nombres ne sont là que pour aider l'utilisateur à identifier les nœuds, seule la structure finale doit être identique à celle attendue. La Figure 2 montre deux résultats finaux identiques. Les arbres de départ et d'arrivée ainsi que le nombre d'essais est fixé par le développeur même.

Certains nœuds spéciaux, fixés dans la structure de départ avant une partie, peuvent subir des actions particulières décrites ci-après, ces nœuds sont dits « déclenchables ». Le joueur peut déclencher tous les nœuds spéciaux atteignables depuis la racine pour peu que le chemin depuis la racine ne passe pas par un autre nœud spécial (Figure 3). Les types de nœuds spéciaux sont identifiables par une lettre : X, S, D. Ces nœuds permettent de déclencher un type d'action. Une fois déclenchée, les nœuds résultant de l'action deviennent normaux (sans type particulier).

1. http ://www.cipherprime.com/games/splice
2. Démo disponible sur steam pour Mac/Win/Linux à l'adresse http ://store.steampowered.com/app/209790/ ?l=french

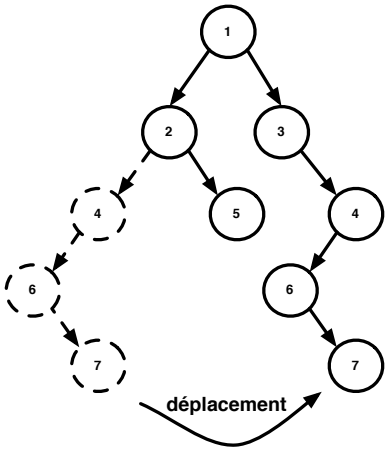

FIGURE 1 – Exemple de déplacement d'un nœud à un autre correspondant à la commande 4-3-FD

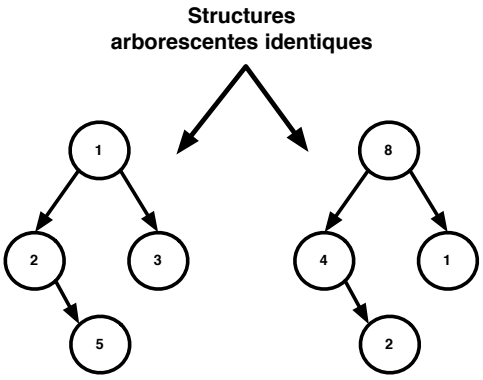

FIGURE 2 – L'arbre de gauche est identique à l'arbre présenté à droite

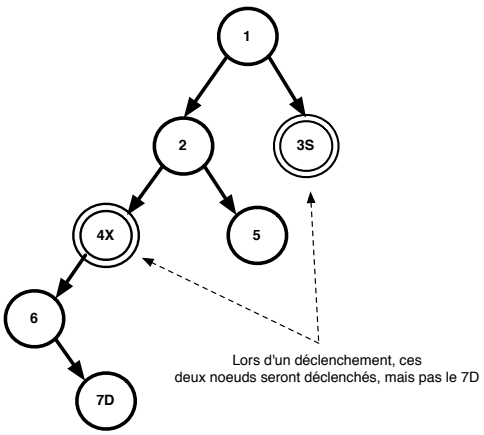

FIGURE 3 – Structure arborescente montrant les nœuds déclenchables simultanément

Le joueur active les nœuds spéciaux sous la forme d'une vague de déclenchement à l'aide de la commande `trigger` en lieu et place d'un déplacement.

Les nœuds de types X (eXtend) créent une extension d'un nœud (Figure 4). Un nouveau nœud sera créé auquel on rattachera en tant que fils gauche le sous-arbre à partir du nœud X compris.

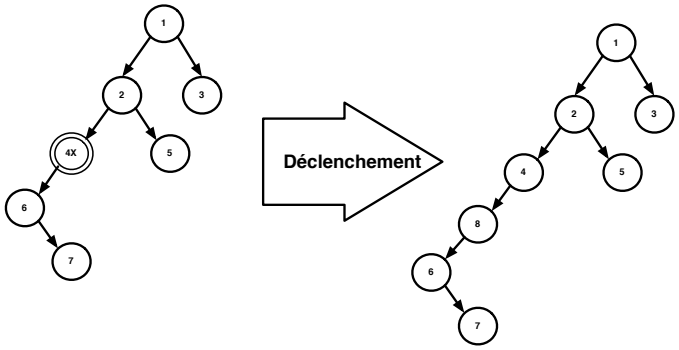

FIGURE 4 – Exemple de déclenchement d'un nœud spécial X

Les nœuds de types S (Split) dupliquent le sous-arbre dont ils sont la racine. Les deux sous-arbres seront rattachés en tant que fils droit et fils gauche à un nouveau nœud racine, qui lui-même s'accrochera en lieu et place de l'ancien nœud de type S (Figure 5).

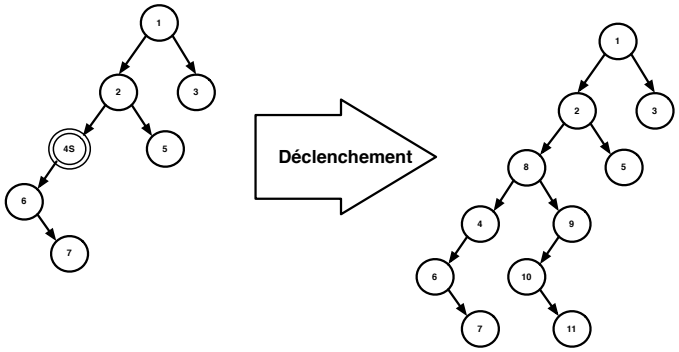

FIGURE 5 – Exemple de déclenchement d'un nœud spécial S

Les nœuds de type D (Deletion) détruisent le nœud ainsi que l'entièreté de ses sous-arbres droit et gauche (Fig. 6).

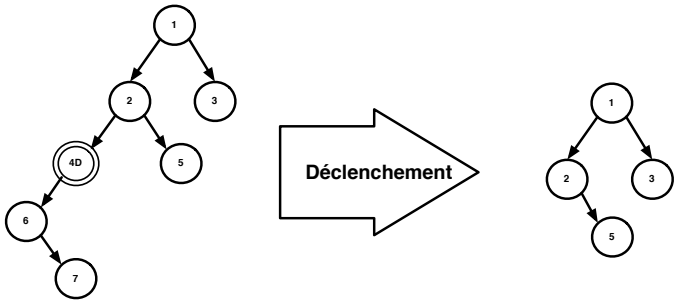

FIGURE 6 – Exemple de déclenchement d'un nœud spécial D

Pour l'affichage, les arbres auront une profondeur maximale connue stockée dans leur instance afin de pouvoir les afficher. Cet affichage devra être formaté afin d'être lisible par le joueur. Par exemple, vous pouvez vous

aider d'une matrice de chaînes de caractères de taille $hauteur \times 2^{hauteur}$.

Le joueur devra toujours avoir les arbres de départ et d'arrivée affichés en console avant d'effectuer le mouvement suivant.

2 Partie Algorithmique

Nous vous demandons d'implanter le jeu sous la forme d'une classe dans laquelle on passera trois paramètres au constructeur : l'arbre de départ et l'arbre d'arrivée ainsi que le nombre de mouvements. Cette classe devra s'appeler `SplitGame` et s'occupera entièrement du jeu. Elle devra impérativement implanter une méthode `run(self)` lançant et gérant une partie. Votre classe devra comporter une découpe du problème en méthodes cohérentes afin de manipuler, afficher, et récupérer les commandes du joueur.

Plusieurs méthodes seront appelés dans la méthode `run(self)` qui effectuera la boucle de jeu. Nous vous donnons la signature de quelques unes d'entre elles :

- `interact(self)` va recevoir la commande entrée dans l'invite de commande qui effectuera soit un déplacement soit un déclenchement ;
- `trig(self)` effectue l'activation des noeuds spéciaux ;
- `move(self,from,to,which)` effectue le déplacement du nœud `from` pour le rattacher au nœud `to` et `which` indique de quel côté rattacher le nœud.

Vous pouvez également vous inspirer du fichier `main` fourni pour le C++ pour établir la signature d'autres méthodes dont vous aurez besoin. Nous vous fournirons un exemple d'arbre de départ et d'arrivée avec sa solution afin de tester votre programme peu de temps après la sortie de cet énoncé.

3 Partie C++

Pour le projet du cours de langages de programmation I, nous vous demandons de ne pas implanter l'entièreté du jeu, mais simplement la classe `Tree` du jeu ainsi que certaines des méthodes de `SplitGame`. Pour vous aider, nous vous fournissons un fichier de `main` reprenant les signatures de ce que vous avez besoin d'implanter. Nous insistons sur l'emploi des mots clés `const`, `static` ainsi que sur l'emploi des notions de programmation orienté objets (cette dernière demande s'applique également à la version `Python`). Bien que les arbres sont fixés par le développeur pour les parties de jeux, nous sous demandons, uniquement pour la partie `C++`, d'avoir une méthode `generateRandomTree()` pour la classe `Tree` qui génèrera un arbre binaire aléatoire de hauteur aléatoire.

4 Consignes Générales

Les consignes sont celles reprises dans le document `http://www.ulb.ac.be/di/consignes_projets_INFO1.pdf`. Les dates de remise sont le lundi 30 avril 2013 pour la version `Python` pour le cours d'algorithmique, le lundi 6 mai 2013 pour la version en `C++` pour le cours de langages de programmation I.