Adrien Holvoet & Carole Merle

Rapport ACT : TP Hexapawn

Question 1

Pour calculer les successeurs d'une configuration, il y a deux cas possibles :

- Tous les successeurs sont positifs : On prend la valeur maximale
- Successeurs négatifs et positifs : On prend la valeur maximale des négatifs

Question 2

Nous avons choisi de représenter une configuration par une structure nommée *Configuration* caractérisée par :

- char joueurAQuiEstLeTour
- unsigned int n,m //dimension du plateau
- vector<vector<char>> plateau // tableau 2D correspondant à la position de nos pions sur le plateau

```
struct Configuration {

    char joueurAQuiEstLeTour;
    unsigned int n, m;
    vector<vector<char> > plateau;

    Configuration(vector<vector<char>> p, unsigned int l, unsigned int c, char
 pion) {
        joueurAQuiEstLeTour = pion; //P = blanc et p = noir
        plateau = p;
        n = l;
        m = c;
    }
};
```

On calcule l'évaluation d'une configuration de manière naïve de la manière suivante :

Nous regardons le cas de base (=joueur perdu, ne peut pas jouer ou un pion adversaire à atteint sa base), si la configuration est dans un cas de base on retourne 0.

Si la configuration n'est pas dans le cas de base, on génère l'ensemble des sous-configurations possibles à partir de cette première et on retourne la valeur optimale des successeurs de chaque sous-configuration.

Pseudo code :

```
calculConfiguration(config) {
 int [] successeurs ;
 If(cas de base(config)) {
 return 0;
 }
 else {
        If(pionAvance) {
        new config;
        successeurs.push(calculConfiguration(config)) ;
        }
        if(pionPrendàDroite) {
        new config;
        successeurs.push(calculConfiguration(config)) ;
        }
        If(pionPrendàGauche) {
        new config ;
        successeurs.push(calculConfiguration(config)) ;
        }
return meilleurSuccesseur(successeurs) ; // retourne le meilleur successeur (formule Question 1)
 }
}
```

Question 3

Ici on utilise la mémoïzation pour stocker les configurations et leur évaluations correspondantes déjà calculées et pas la « remontée » car dans cet exercice nous n'avons pas connaissance d'une configuration de base sur laquelle commencer cette « remontée ».

Pour cette mémoïzation nous utilisons une unordered_map (équivalent d'une table de hachage) dans laquelle on stocke :

- Une clef: la configuration (struct) convertie en string.
- Une valeur associée : son évaluation correspondante

On a choisi d'utiliser cette structure de donnée car dans notre cas avec des valeurs complexes, il est plus simple d'utiliser celle-ci car la recherche et l'insertion ont une complexité en O (1).

Voici comment fonctionne cette mémoïzation : à chaque appel de la méthode récursive (calculConfiguration), nous convertissons notre configuration sous forme de string, nous vérifions par la suite si cette clef existe dans notre table de hachage. Si oui, nous retournons la valeur associée à cette clef sinon la fonction calcule l'évaluation de la configuration en cours en l'enregistrant à la fin dans la table de hachage.