TP6 Halving Game

Auteurs : Karam ELNASORY · Erkin Tunc BOYA · Semih DOYNUK

Lien Git : https://github.com/ErkinTunc/ARP-Recherche-Non-Inform

Exercice 1

Q1: Modélisation Constructive

- Etape 1: Espace d'Etat

  $s = \{(n,p) | n \in N, 0 \leq n \leq N, p \in \{1,2\}\}$

  - **n** → nombre actuelle sur le table
  - **p** → Joueur actuel
- Etape 2: Etat-Initial
  - $s_0 = (N, 1)$ ← joeur 1 nombre départ
- Etape 3: Actions

  Action → *sub* , *div*

  - sub = $n - 1$
  - div = $\lfloor n/2 \rfloor$
- Etape 4: Fonction de Successeurs
  - $succ(s, c)$
  - $succ(s, sub) = (n - 1, 3 - p)$
  - $succ(s, div) = (\lfloor N2 \rfloor, 3 - p)$
- Etape 5: Ensemble d'états but $T \subseteq S$
  - $S_t \in \quad \{(0,p) \Leftrightarrow (0,1), (0,2)\}$ ← état final
- Etape 6: Fonction de coût
  - $u(s,p)$: +1 pour le joueur gagnant, -1 pour le joueur perdant.

Question 2

- Le taille de l'espace d'états en fonction de nombre N est $2(N+1)$ car on peut atteindre de 0 a N

Question 3

On suppose que le joueur 1 commone le Jeu

- Le premier joueur vise a maximiser cette fonction $u(s,1)$
- Le deuxieme joueur vise a minimiser cette fonction $u(s,2)$

Implementation

```
public Action minmax_policy (State s){
    Action best=null;
    int bestValue;

    if (s.player == 1) {
        bestValue = Integer.MIN_VALUE;
        for (Action a : s.actions()) {
            int v = minmax_value(s.succ(a));
            if (v > bestValue) {
                bestValue = v;
                best = a;
            }
        }
    }
    else {
        bestValue = Integer.MAX_VALUE
        for (Action a : s.actions()) {
            int v = minmax_value(s.succ(a));
            if (v<bestValue) {
                bestValue = v;
                best = a;
            }
        }
    }
    return bestAction;
}

private int minimax_value(State s) {
        if (s.is_goal()) {
            return s.utility();
        }


        if (s.player == 1) {
            int maxVal = Integer.MIN_VALUE;
            for (Action a : s.actions()) {
                int v = minimax_value(s.succ(a));
                maxVal = Math.max(maxVal, v);
            }
            return maxVal;
        } else {
            int minVal = Integer.MAX_VALUE;
            for (Action a : s.actions()) {
                int v = minimax_value(s.succ(a));
                minVal = Math.min(minVal, v);
            }
            return minVal;
        }
    }
```

Compilation et exécution

Compilation :
javac -d build src/*.java

Execution :
java -cp build Launcher