Saé S1.02 : comparaison d'approches algorithmiques

Puissance 4 : faire jouer l'ordi

Présentation

Cette séance vise à vous faire reprogrammer un jeu de Puissance4 afin de faire jouer un *humain* contre l'ordinateur (appelé *ordi* par la suite).

Un programme de Puissance4 est disponible sur Moodle (fichier *Puissance4.c*). Il permet de faire jouer deux humains entre eux. C'est sur ce code source que vous allez travailler. Voici son programme principal :

```
int main()
{
    Grille laGrille;
    char vainqueur=INCONNU;
    int ligne, colonne;

    initGrille(laGrille);
    afficher(laGrille, PION_A);
    while (vainqueur==INCONNU && !grillePleine(laGrille)){
        jouer(laGrille,PION_A, &ligne, &colonne);
        afficher(laGrille, PION_B);
        if (estVainqueur(laGrille, ligne, colonne) ){
            vainqueur = PION_A;
        } else {
            jouer(laGrille, PION_B, &ligne, &colonne);
            afficher(laGrille, PION_A);
            if (estVainqueur(laGrille, ligne, colonne) ){
                vainqueur = PION_B;
            }
        }
    }
    finDePartie(vainqueur);
    system(EXIT_SUCCESS);
}
```

on fait jouer le joueur A...

...puis on fait jouer le joueur B

1 – Faire jouer l'ordi

Dans le jeu de Puissance4, toute la stratégie réside dans le choix de la colonne où le pion doit tomber. Dans le programme, ce choix est assuré par la fonction `choisirColonne(…)`. Pour l'instant cette fonction demande à chaque utilisateur de saisir le numéro de la colonne qu'il a choisie.

Mais si on veut "faire jouer l'ordi", il faut que le choix de la colonne soit programmé.

Exercice 1

Écrivez une fonction `choisirColonneStrategie1(...)` qui consiste à choisir la colonne la "plus à gauche". Cette fonction retournera la colonne 0, ou bien la colonne 1 si la colonne 0 est pleine, ou bien la colonne 2 si la colonne 0 et la colonne 1 sont pleines, etc.

Il faut ensuite adapter le programme pour faire jouer l'ordi : le joueur A restera le joueur humain (il aura les pions X) et le joueur B sera associé à l'ordi (il aura les pions O).

Exercice 2

1. Dupliquez la fonction `jouer(...)` afin de disposer de deux fonctions `faireJouerA(...)` et `faireJouerB(...)`. Modifiez le `main` en conséquence.

2. Dans la fonction `faireJouerB(...)`, faites en sorte que la colonne soit choisie selon la stratégie définie dans l'exercice 1.

3. Testez le programme et essayez de battre l'ordi.

2 – D'autres stratégies

Programmer une autre stratégie revient en fait à fournir une autre fonction `choisirColonne`.

Exercice 3

Programmez les trois stratégies suivantes.

- stratégie 2 : choisir la colonne la moins pleine
  Écrivez une fonction `choisirColonneStrategie2(...)` qui consiste à choisir la colonne la moins remplie, c'est-à-dire celle qui contient le moins de pions.

- stratégie 3 : choisir la colonne la plus au milieu
  Écrivez une fonction `choisircolonneStrategie3(...)` qui consiste à choisir la colonne numéro 3, celle du milieu. Si elle est pleine, choisir la colonne 2 ou la colonne 4. Si elles sont pleines toutes les deux, choisir la colonne 1 ou la 5, etc.

- stratégie 4 : choix aléatoire
  Écrivez une fonction `choisirColonneStrategie4(...)` qui consiste à choisir la colonne de manière aléatoire. On rappelle que la fonction `rand()` retourne un entier compris entre `0` et `RAND_MAX`, et qu'il faut impérativement exécuter `srand(time(NULL));` au début du programme.

À chaque fois, testez et essayez de battre l'ordi. Quelle stratégie est la plus facile à battre ? Laquelle est la moins facile à battre ?

3 – Faire jouer l'ordi contre lui-même

Et si on supprimait le joueur humain ? Aucune des stratégies proposées précédemment n'est difficile à battre, l'humain est plus fort que l'ordi à chaque fois (enfin, normalement !). Pour savoir quelle stratégie est la meilleure parmi les quatre programmées, faisons jouer l'ordi contre lui-même.

Exercice 4

Modifiez maintenant votre programme afin de faire joueur l'ordi contre lui-même. Par exemple le joueur A (qui n'est plus humain) joue avec la stratégie 1 et le joueur B joue avec la stratégie 2. Testez votre programme plusieurs fois. Quelle semble être la meilleure stratégie ?

4 – Mesurer les performances

Intéressons-nous maintenant à la "performance" des différentes stratégies. Comment savoir qu'une stratégie est meilleure qu'une autre ?

Nous proposons deux moyens triviaux pour comparer deux stratégies : le nombre de pions joués lors d'une partie (gagner en ayant joué 6 pions c'est mieux que gagner en ayant joué 12 pions) et le nombre de victoires sur un grand nombre de parties (remporter 90% des parties n'est sans doute pas que du hasard).

4.1 – Nombre de pions joués

Exercice 5

Modifiez le programme en mettant en place les compteurs nécessaires afin de pouvoir afficher, en fin de partie, le nombre de pions joués par le vainqueur.

Comparer les différentes stratégies entre elles.

4.2 – Statistiques

Exercice 6

Modifiez à nouveau votre programme afin qu'il exécute N parties (avec par exemple N=100). Vous afficherez les résultats sous la forme :

```
joueur X : 31 victoires
joueur O : 67 victoires
2 parties nulles
```

Attention, l'affichage de la grille à chaque coup joué va ralentir énormément l'exécution. Supprimez cet affichage pour n'afficher que le résultat des N parties.

Enfin, créez un tableau comme celui-ci et complétez-le avec les résultats obtenus (n'oubliez pas que c'est systématiquement le joueur A qui commence la partie).

joueur A / joueur B	stratégie 1	stratégie 2	stratégie 3	stratégie 4
stratégie 1 (la + à gauche)				
stratégie 2 (la - remplie)				
stratégie 3 (la + au milieu)				
stratégie 4 (aléatoire)				