TP10 – Des jeux pour réviser

Projet de programmation M1

08 Décembre 2015

1 Le jeu du 2048

Le jeu du 2048 se joue sur une grille 4×4. Les cases sont soit vides soit contiennent une puissance de 2 (supérieur ou égal à 2). Le joueur choisit de déplacer toutes les cases vers la droite, la gauche, le haut ou le bas. Si deux cases ayant la même valeur se rencontrent, elles fusionnent et leur valeur s'additionne. Lorsque le joueur a joué, l'ordinateur choisit aléatoirement et uniformément, une case libre et lui donne la valeur 2 avec probabilité 9/10 et 4 avec probabilité 1/10. Si le joueur ne peut plus bouger de blocs, il perd. Son but est de créer un bloc de valeur 2048.

Par exemple, si on a la grille suivante :

$$\begin{pmatrix} 2 & 4 & 8 & . \\ 2 & 2 & 4 & 4 \\ 2 & 4 & 8 & 16 \\ 2 & . & . & . \end{pmatrix}$$

Et qu'on déplace les cases vers la droite, on aura :

$$\begin{pmatrix} . & 2 & 4 & 8 \\ . & . & 4 & 8 \\ 2 & 4 & 8 & 16 \\ . & . & . & 2 \end{pmatrix}$$

Dans le cas où on a trois valeurs identiques qui se rencontrent, on fusionne toujours en premier celles qui sont le plus proche du bord vers lequel on déplace les cases. Par exemple, si on déplace la ligne (2 2 2 0) vers la droite, on obtiendra (0 0 2 4) et non pas (0 0 4 2). Si on déplace cette même ligne vers la gauche en revanche, on obtiendra (4 2 0 0).

On va implémenter un jeu de 2048. On veut pouvoir y jouer sur des grilles de taille arbitraire.

1. Écrire une fonction `move_right(int *t, int length)` qui déplace une ligne (dans le tableau `t`) de longueur `length` vers la droite comme expliqué ci-dessus. **N'oubliez pas de tester votre fonction**.
2. Définissez 4 constantes : `UP, DOWN, LEFT, RIGHT`. Écrire une fonction `move_grid(int grid[n][m], int n, int m, int direction)` qui déplace les cases dans la direction donnée en argument. On pourra utiliser `move_right` en l'appelant sur les bons tableaux (par exemple, pour déplacer vers la gauche la ligne l, on peut déplacer vers la droite une ligne qui contient le mirroir de l et réécrire le résultat dans la bonne ligne et le bon sens).
3. Implémenter le jeu complet.

2 Le Sudoku

On représente une grille de Sudoku par une grille d'entier 9 par 9. Si une case est vide, elle aura la valeur 0. Sinon, sa valeur sera le chiffre inscrit dans la grille.

1. Écrire une fonction `int check_value(int m[9][9], int v, int i, int j)` qui vérifie si on peut écrire le chiffre v à la case (i, j).

2. Écrire une fonction `int check_grid(int m[9][9])` qui vérifie si les valeurs remplies jusqu'ici sont cohérentes (chaque chiffre apparaît au plus une fois dans chaque ligne, colonnes, bloc de taille 3×3).
3. Écrire une fonction `int solve(int **m)` qui renvoie 1 si la grille a une solution et remplit m en conséquence et qui renvoie 0 sinon. On procédera par bruteforce, c'est-à-dire qu'on essaiera toutes les solutions possibles (ou presque) avec une fonction récursive `int solveaux(int m[9][9], int i0, j0)`. Si la case (i_0, j_0) est remplie, on appelle récursivement sur la case suivante. Si la case est vide, on remplit la case avec la plus petite valeur possible et on essaiera de résoudre récursivement en passant à la case suivante. Si l'appel récursif échoue, alors on tentera de remplir la case vide avec la valeur suivante. Sinon, on sait qu'on a fini de remplir la grille. Si on a testé la fonction avec toutes les valeurs possibles sans succès, on remet la case (i_0, j_0) à 0 et on renvoie 0.