Projet IPI2 1A sem 2 - 2016-2017 - Ensiie Evry
Pierre Tellier
pierre.tellier@ensiie.fr

Color Flood

Votre travail consiste à réaliser, en équipe de 4 personnes, une application ludique, et le solveur associé. Le jeu à programmer aura les fonctionnalités de ce jeu :
http://unixpapa.com/floodit/
En plus de la possibilité de jouer, votre application devra permettre de trouver la solution qui demande le plus petit nombre de coups.

Description du problème.

Le jeu se présente sous la forme d'une grille carrée, de taille 12x12, 18x18 (taille par défaut) ou 24x24. Chaque case de la grille est initialisée aléatoirement par une des 6 couleurs disponibles : bleu, vert, rouge, jaune, marron et gris. L'objectif est de faire en sorte que toutes les cases de la grille deviennent identiques, en un nombre d'essais limité.
La partie se déroule en modifiant la couleur de la case située dans le coin en haut à gauche de la grille (origine) et de toutes celles qui appartiennent à la même composante 4-connexe que cette case : l'ensemble incluant la case origine et constitué des cases qui possèdent la même couleur et sont voisines les unes des autres par un côté. Le nombre de coups autorisés restant est alors diminué de 1. On répète cette opération. Pour simplifier l'affichage, on pourra représenter les couleurs par les caractères 'B', 'V', 'R', 'J', 'M' et 'G'.

Contraintes

- Vous devrez effectuer le développement en langage C ;
- Les fonctions du lot A devront faire l'objet de tests unitaires ;
- Les codes seront compilés avec les options -Wall -Wextra, la compilation ne produira aucun warning ;
- Un makefile sera fourni avec chaque lot ;
- Les codes sources seront gérés avec un dépôt Git (gitlab) ;
- L'absence de fuites de mémoire de vos lots devra être vérifiée avec Valgrind ;
- Les codes seront commentés au format Doxygen et une documentation devra pouvoir être générée à la demande grâce à une cible dans le Makefile ;
- Les consignes pour la constitution des équipes seront communiquées lors de la séance de lancement.
- Le suivi de projet sera réalisé à l'aide de Trello ou Framaboard.

Travail à réaliser.

A. lot A. Préparation des données.

La première partie de votre travail consiste à écrire et tester le code qui vous permettra d'initialiser et manipuler vos structures de données :
- Allocation dynamique d'une grille carrée de couleurs de taille variable ;
- Libération de l'espace mémoire occupé par une grille ;
- Initialisation de la grille à partir de valeurs aléatoires ;
- Initialisation de la grille à partir de valeurs contenues dans un fichier (pour la répétition des tests). Pour cela vous devrez être capable de créer des fichiers ;
- Remplacement de la couleur de la case de coordonnées (x,y) par une couleur c ;
- Identification de la composante 4-connexe incluant la case origine ;
- Changement de couleur de toutes les cases de la composante connexe incluant la case origine ;
- Test de la présence d'une même valeur dans toutes les cases.

B. Réalisation d'un jeu avec affichage en mode texte dans le terminal

Vous programmerez une application qui permet d'exécuter une partie de « Color Flood ». Elle devra posséder au minimum les fonctionnalités suivantes :

- lecture au clavier de la taille de grille et du nombre de coups autorisé ;
- boucle de jeu :
  - affichage dans le terminal de la grille et du nombre de coups restants ;
  - sélection d'une couleur ;
  - application des transformations de la grille ;
  - test de fin de partie (victoire, défaite, partie non terminée).

C. Réalisation d'un solveur.

Vous programmerez une application qui détermine la séquence de couleurs à sélectionner afin de terminer une partie en un nombre minimal de coups. Votre première façon de procéder sera par essais successifs. C'est à dire que vous testerez pour une situation donnée toutes les possibilités. Pour chaque situation atteinte vous referez de même. L'algorithme est de la forme :

```
solveur(grille g, entier n, pile solution, int profondeur) {
   // déclarations des variables locales i, g2, …
   for (i=0; i<6; i=i+1) { // pour toutes les couleurs possibles
      solution = empiler(solution,i);
      g2 = propageCouleur(g, i);
      if (terminaison(g2))    uneSolutionTrouvee(solution);
      else solveur(g2, n, solution, profondeur+1);
      depiler(&solution);
   }
}
```

Vous optimiserez votre solveur en évitant de continuer d'explorer les branches dont on sait qu'elles permettront de découvrir des solutions qui demandent un nombre de coups supérieur à la meilleure solution trouvée depuis le début de la résolution.

D. Réalisation d'un jeu complet, avec initialisation aléatoire des grilles, utilisation du solveur pour déterminer le nombre de coups autorisés, et optimisation du solveur.

Le lot D consiste à produire l'application complète, qui génère aléatoirement une grille et indique le nombre de coups pour trouver la solution, déterminé par le solveur.
Dans ce lot, vous proposerez aussi une autre approche pour votre solveur. Vous ne parcourrez plus toutes les solutions « en profondeur d'abord », mais parcourrez l'arbre d'exploration des solutions dans un ordre déterminé à l'aide d'une fonction heuristique. Pour cela, il vous faudra mettre en oeuvre une structure d'arbre de grilles… Ce point sera développé lors de la séance de présentation du projet et pendant les séances encadrées.

LIVRABLES

- Tous les codes source commentés, et documentés au format Doxygen.
- Un rapport de projet (algorithmes, répartition du travail, planning, …).
- Un fichier Makefile permettant de compiler le programme.
- Des jeux de tests unitaires avec le lot A.
- Un README avec les instructions d'installation et utilisation (OBLIGATOIRE).
- Date de livraison lot A : 24 avril 2017
- Date de livraison lot B : 2 mai 2017
- Date de livraison lot C : 15 mai 2017
- Date de livraison lot D : 22 mai 2017

Barème

- ✦ Un code qui ne passe pas à la compilation entraîne une note nulle !
- ✦ Tout retard entraîne des pénalités, 1 point par jour de retard.
- ✦ Absence de warnings à la compilation : 1 point.
- ✦ Qualité de l'écriture du code : lisibilité, présentation, documentation/commentaires, indentation, identificateurs, taille des fonctions, … : 2 points
- ✦ Organisation du code : modules, fichiers d'entêtes, makefile, … : 2 points.
- ✦ Absence de fuites mémoire (vérifiée avec Valgrind) : 1 point.
- ✦ Rapport : installation, utilisation, structures, algorithmes, organisation du travail, …: 1 point.
- ✦ Validation du point A : 4 points (dont 2 pour les tests).
- ✦ Validation du point B : 2 points
- ✦ Validation du point C : 4 points
- ✦ Validation du point D : 3 points
- ✦ Total : 20 points.

Dates des séances encadrées :

- 31 mars 2017 - Démarrage du projet
- 21 avril 2017
- 28 avril 2017
- 12 mai 2017
- 19 mai 2017

Documents utiles disponibles sur pedago-ens.ensiie.fr dans /pub/IPI2 :

- ce sujet
- lecture de caractères à la volée
- exemple d'utilisation de la librairie graphique SDL
- support de la présentation en amphi.
- …