Yann.Secq@univ-lille.fr
Fabien.Delecroix@univ-lille.fr

TD n°15

Objectifs: Consolider la décomposition en fonctions, l'algorithmique et l'utilisation des tableaux à deux dimensions.

Ce sujet est quasiment identique à un exercice de DS de janvier 2017. Seules quelques questions ont été changées ou supprimées. Les questions n'ayant pas de barème ont été ajoutées.

Introduction : Match-3 (variante du jeu Candy Crush)

Dans cette version simplifiée du célèbre jeu de *Tile-Matching*, nous considérons un plateau de jeu de 5 lignes et 8 colonnes dont chaque case comporte un bonbon. Il existe 5 types de bonbons qui sont représentés à l'écran par les lettres `B, O, J, V, R` (pour bleu, orange, etc.).

Le but du jeu est de dépasser un certain score (1000) en un certain nombre de coups (20). Un coup consiste à échanger les bonbons de deux cases adjacentes, de manière à aligner, si possible, au moins 3 bonbons du même type. L'échange ne peut se faire que suivant deux cases adjacentes, juxtaposées horizontalement ou verticalement.

Si trois bonbons ou plus sont alignés, ceux-ci disparaissent, ce qui permet au joueur de gagner 10 points par bonbon. Les bonbons dans les colonnes supérieures descendent alors et de nouveaux bonbons, tirés de manière aléatoire, apparaissent dans la ligne des bonbons déplacés jusqu'à compléter toutes les lignes. Suite à ces déplacements de bonbons, de nouveaux alignements peuvent apparaître, ré-itérant le processus jusqu'à ce qu'il n'y ait plus d'alignement possible. Lors d'un échange de bonbons, des alignements peuvent apparaitre à la fois suivant une ligne et une colonne. Dans ce cas, tous les bonbons correspondant à ces alignements disparaissent.

À la fin de ce sujet vous trouverez un exemple d'un début de partie de cette variante de Candy Crush.

Dans ce sujet on vous demande d'écrire plusieurs fonctions utiles pour implémenter le jeu. La plus grande partie des fonctions demandées peuvent être écrites indépendamment des autres.

Fonctions utilitaires (9 points) [1]

Le plateau de jeu sera représenté par un tableau à deux dimensions de bonbons. Votre programme doit pouvoir marcher pour n'importe quelle taille de tableau de moins de 9 colonnes et moins de 26 lignes (ces restrictions sont faites uniquement pour faciliter l'affichage).

Q1. Écrivez l'énumération `Bonbon` modélisant les différentes couleurs de bonbons.

Q2. (1,5 pt) Écrire une procédure `void initialiser(Bonbon[][] plateau)` qui initialise le plateau de jeu avec des bonbons tirés de manière aléatoire. Pour cette question, on ne se préoccupe pas de la possible présence d'alignements. Le plateau donné en paramètre est supposé être alloué préalablement.

Q3. (1,5 pt) Écrire une procédure `afficher` qui affiche le plateau de jeu pour obtenir la même présentation que proposée dans l'exemple qui suit, à savoir que

- chaque type de bonbon est représenté par sa lettre correspondante (à faire dans une autre fonction à part)
- et que les lignes sont désignées par une lettre et les colonnes par un chiffre.

```
  12345678
A|VROOOVBR
B|VBBRVRBJ
C|BBOORRJO
D|OVBRJJRJ
E|VRBOJJRB
```

On s'intéresse maintenant à l'échange de bonbons de cases adjacentes. Les cases à échanger vont être désignées par l'utilisateur par une séquence de quatre chiffres et lettres. Chaque case est désignée d'abord par la lettre correspondant à sa ligne puis le chiffre correspondant à la colonne. Par exemple `C7D7` correspond à l'échange du contenu des cases de coordonnées `C7` et `D7`. On suppose ici que les cases passées en paramètre à la fonction `boolean casesAdjacentes(String cases)` appartiennent bien au plateau de jeu.

1. Le barème de chaque exercice est donné à titre indicatif.

Q4. Écrire une fonction de test pour la fonction `casesAdjacentes`.

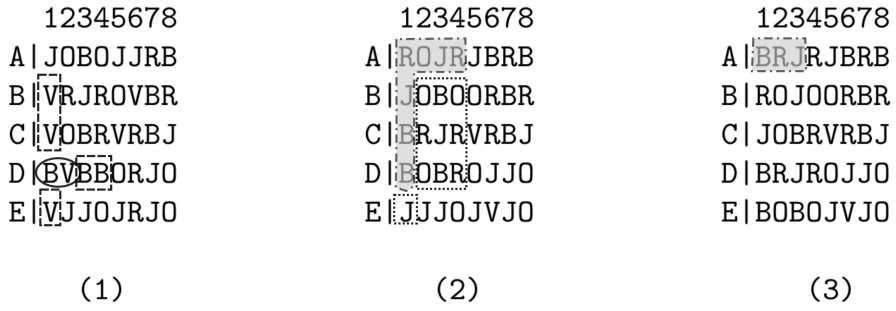
Vous veillerez à avoir les étapes suivantes lors de chaque tour de jeu:

— demander à l'utilisateur de rentrer des coordonnées de deux cases à échanger;
— échanger les bonbons de ces cases;
— éliminer les bonbons qui participent à des alignements horizontaux ou verticaux;
— faire tomber les bonbons dans les places vides, puis ajouter de nouveaux bonbons aléatoires dans les cases qui restent vides, et éliminer les éventuels nouveaux alignements qui apparaissent. Répéter cette étape en cascade jusque ce que aucun nouvel alignement n'apparaisse.

Le jeu s'arrête quand on a atteint le score demandé (1000) ou si on a utilisé tous les coups (20). Ne pas oublier les affichages nécessaires pour guider l'utilisateur.

Q5. Écrire l'algorithme principal `void algorithm()` permettant de jouer à Candy Crush. Vous devez réutiliser les fonctions demandées dans les exercices précédents. Vous pouvez également introduire d'autres fonctions, pour lesquelles dans un premier temps vous donnerez uniquement les signatures. Si vous en avez le temps, vous écrirez ensuite les tests permettant de valider ces fonctions avant finalement d'écrire les corps de ces fonctions.

Exemple de début de partie du Candy Crush.



`(1)` est la configuration initiale. Après échange de `D1` et `D2` on obtient un alignement horizontal de 3 `B` sur la ligne `D`, et un alignement vertical de 4 `V` sur la colonne `1`.

`(2)` est le résultat de la suppression des alignements. Le `J` qui se trouvait dans `A1` en `(1)` est maintenant tombé en `E1` suite à la disparition des `V`. Le bloc de 3×3 bonbons `OBORJROBR` est tombé une ligne plus bas suite à la disparition des `B`. Les nouveaux bonbons tirés aléatoirement sont ceux entouré de ·-·-·. Un nouvel alignement de `J` s'est formé sur la ligne `E`.

`(3)` les `J` de l'alignement ont disparu. Les bonbons des colonnes `1`, `2` et `3` sont tombés une ligne plus bas et trois nouveaux bonbons `BRJ` sont apparus dans les cases `A1`, `A2` et `A3`. Il n'y a plus d'alignement, il faudra demander un nouvel échange au joueur.