51IF2IK3 – Examen Session 2

23 juin 2009 – durée 3h – barème indicatif

Soit un fil sur lequel on désire enfiler des perles de trois couleurs qu'on dénote pour simplifier avec les nombres 0, 1 et 2.

Pour constituer un collier, on pose une seule contrainte : il ne doit pas y avoir sur le fil deux séquences adjacentes identiques. Par exemple : le collier 0 est correct, mais 00 est incorrect ; 01 et 010 sont des colliers corrects, mais 0100 et 0101 sont incorrects car pour le premier la séquence 0 est répétée tandis que pour le deuxième la séquence 01 est répétée. La seule possibilité pour rallonger le collier 010 est d'ajouter la perle 2.

Question 1. (1p) Compléter le collier 0102 pour obtenir un collier correct avec 6 perles. Combien de colliers corrects de 6 perles on peut obtenir en partant de 0102 ?

Question 2. (2p) Dérouler à la main un algorithme de backtrack permettant, en rangeant les entiers des colliers dans une pile, d'obtenir **TOUS** les colliers de longueur 5. On suppose qu'il existe un test `collierCorrect` qui renvoie vrai si et seulement si le nouveau collier rangé dans la pile est correct.

Question 3. (7p) *Cette partie de l'examen doit être réalisée sous Eclipse !*

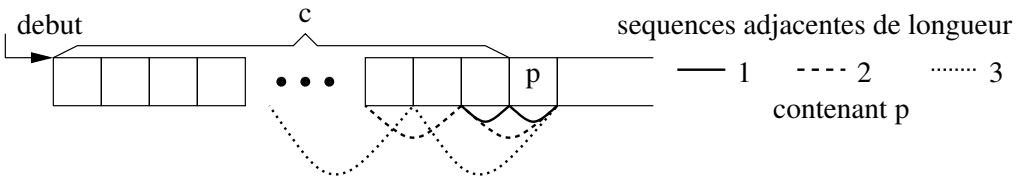
Programmer l'algorithme de backtrack du point précédent en définissant une classe `CollierBase` ayant les méthodes suivantes :

- `traiter` qui implémente le backtrack pour engendrer **TOUS** les colliers de longueur L. La pile contiendra les entiers constituant le collier en cours de construction. Pour implémenter la pile, vous utiliserez la classe `Stack`.
- `affiche` qui affiche le contenu de la pile (donc qui permet d'afficher une solution).
- `collierCorrect` qui appelle la méthode *statique* `testCollier` de la classe `TestCollier` dont le code binaire est disponible à l'adresse `/home/malben45/TestCollier.class`. La méthode `testCollier` reçoit comme argument une pile d'entiers (`Stack<Integer>`) et renvoie un booléen.
- `main` qui permet de donner la longueur L avant de lancer le backtrack.

Si besoin, vous pouvez implémenter d'autres méthodes auxiliaires.

Question 4. (6p) L'objectif de ce point est d'obtenir un algorithme pour `collierCorrect`, sans utiliser `testCollier`.

1. L'allongement du collier d'un seul côté dans l'algorithme de backtracking conserve la propriété d'absence de séquences adjacentes identiques pour le début du collier. C'est pourquoi, à l'ajout d'une perle à la fin du collier, uniquement les séquences contenant cette dernière perle doivent être considérées. Ce fait est formalisé par la propriété suivante (voir figure) : si à un collier correct c on ajoute une perle p, le collier cp est correct ssi les séquences adjacentes dont une contient p sont différentes.
   (a) (1p) Montrer que cette propriété est vraie pour l'allongement avec la perle 2 du collier 010.
   (b) (1p) Prouver formellement cette propriété.



2. Le fait ci-dessus permet de construire un algorithme pour `collierCorrect` qui considère itérativement les séquences adjacentes de longueur 1, 2, etc. telles qu'une des séquences se termine par la perle ajoutée (voir figure).
   (a) (1p) Pour un collier de longueur L, combien de séquences adjacentes dont une contient la dernière perle ajoutée doivent être considérées ?
   (b) (1p) Écrire un algorithme `identiques` qui reçoit un vecteur (tableau) d'entiers t, deux indices d_1 et d_2 (de fin de chaque sous-séquence), et une longueur de sous-séquence ℓ ; cet algorithme renvoie vrai ssi les séquences $t[d_1-\ell+1]..t[d_1]$ et $t[d_2-\ell+1]..t[d_2]$ sont identiques.
   (c) (2p) Écrire l'algorithme pour le test `collierCorrect` qui reçoit comme argument un vecteur d'entiers.

Question 5. (4p) *Cette partie de l'examen doit être réalisée sous Eclipse !*

1. (1p) Pour programmer l'algorithme de `collierCorrect`, définir une classe `Collier` qui hérite de la classe `CollierBase` et redéfinit la méthode `collierCorrect`.
2. (3p) Programmer l'algorithme de `collierCorrect` sans faire appel à `testCollier`. Si besoin, vous pouvez implémenter d'autres méthodes auxiliaires. Tester cette méthode en appellant la méthode (héritée) `traiter`.