Grands Moulins

INITIATION À LA PROGRAMMATION (IP2)

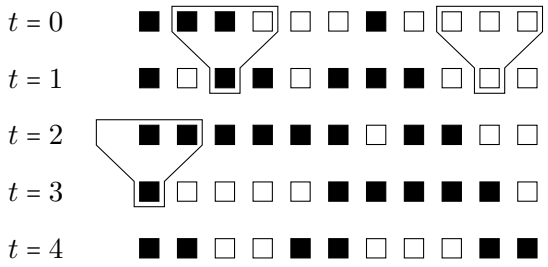
TP 6 - 7 : Liste doublement chaînée et automate cellulaire

Le but de ce TP est d'implémenter une version simplifiée d'automate cellulaire, vu alors comme une liste doublement *chaînée de booléens, qui indique si la cellule correspondante est vivante ou morte.*

Exercice 1 [Liste doublement chaînée]

1. Créez une classe `Cellule` contenant :
   (a) trois attributs privés : `precedente` et `suivante` de type `Cellule`, `noire` de type `boolean` ;
   (b) les accesseurs pour ces attributs ;
   (c) un constructeur `Cellule(boolean noire)` qui initialise l'attribut `noire` avec l'argument (et les deux autres attributs à `null`) ;
   (d) une méthode `void afficher()` qui imprime (sans retour à la ligne) un dièse `#` ou un tiret `-` selon que `noire` est vraie ou fausse.
2. Définissez une classe de liste pour ces cellules, avec références vers début et fin, et écrivez les méthodes qui prennent en argument un booléen et permettent d'ajouter en début ou en fin une cellule correspondante.
3. Ecrivez des méthodes d'affichage, et testez votre code en définissant une séquence correspondant à la première ligne de la figure ci dessous.

Exercice 2 [Automate] La suite décrit le fonctionnement d'un automate cellulaire.
À chaque instant t, chaque cellule est soit noire (`noire==true`) soit blanche (`noire==false`). L'état à l'instant $(t+1)$ d'une cellule donnée dépend de l'état à l'instant t de ses voisines, ainsi que de son propre état à l'instant t.
Considérons par exemple la règle de l'*unanimité*, c'est-à-dire que la cellule d'indice i à l'étape $(t+1)$ est blanche si les cellules $(i-1)$, i et $(i+1)$ portent le même état à l'étape t, voir figure ci-dessous. (Par convention, la cellule virtuellement à gauche de la première (*resp.* virtuellement à droite de la dernière) est toujours considérée comme blanche).



Sont encadrés des exemples d'application de la règle de l'unanimité. À $t = 1$, la cellule 3 est noire car, à $t = 0$, les états des cellules 2, 3 et 4 ne sont pas identiques. De façon analogue, l'avant-dernière cellule devient blanche à $t = 1$ car ses deux voisines ont le même état que le sien à $t = 0$. Le troisième cadre souligne le fait que la première cellule considère que sa voisine de gauche est blanche.

1. Renommez votre classe de liste en `Automate`, et ajoutez une methode `void initialisation()` qui fixe l'état de l'automate comme représenté à la figure précédente à $t = 0$

   On veut créer une fonction qui change le statut `noire` des cellules en fonction du temps. Il n'est pas possible de faire ceci avec un seul parcours de la liste car la mise à jour prématurée d'une cellule peut changer le résultat de la mise à jour de sa voisine. La solution retenue consiste à enrichir le modèle en ajoutant un attribut, et faire plusieurs passages.

2. Ajoutez à la classe `Cellule` un attribut `prochainEtat` de type `boolean`. Modifier les constructeurs pour que ce nouvel attribut soit toujours initialisé à `false`.
3. Ajoutez à l'automate une méthode `prochaineEtape()` qui met `prochainEtat` à `true` si la cellule sera noire à l'instant suivant (et `prochainEtat` à `false` si elle sera blanche) en suivant la règle de l'unanimité.
4. Ajoutez ensuite une méthode `miseAJour()` qui met à jour la valeur de `noire` en fonction de celle stockée dans `prochainEtat`.
5. Créez dans la classe `Automate` la méthode `uneEtape()` qui parcourt la liste *deux fois*, la première fois en calculant le prochain état, puis en faisant la mise à jour.
6. Ajoutez une méthode `nEtapes(int n)` qui affiche d'abord l'état courant, puis effectue n étapes successives, en affichant les étapes intermédiaires.
7. Testez avec `n` valant `4` et comparez avec la figure.
8. Ajoutez un constructeur `Automate(String str)` qui prend une chaîne de caractères constituée de `'#'` et de `'-'`, et crée l'automate correspondant.
9. Testez avec diverses chaînes de caractères, en particulier celle avec un seul `#` au milieu.
10. Organisez votre code pour qu'on puisse construire des automates qui utilisent soit la règle de l'unanimité, soit celle de la majorité que vous écrirez également.