IUT de Montpellier – Département informatique – 1ère année

Programmation 1 bis

TP : Classe Ensemble implantée par un tableau de booléens
Prise de conscience de la complexité en temps

1 Introduction

On se propose de créer une classe `EEBool` utilisant comme principale structure un tableau de booléens (`ensTabB`) : pour un ensemble e, la valeur de `e.ensTab[i]` vaut `true` ssi $i \in e$. Cette classe permet de représenter n'importe quel sous-ensemble d'un ensemble $\{0, ..., max - 1\}$.

(Comme on pourra le tester plus loin, cette représentation est avantageuse si l'ensemble créé n'est pas trop creux, c'est-à-dire avec un grand pourcentage d'éléments présents dans le sous-ensemble. Cette représentation contraint les éléments à être positifs ou nul, mais cette restriction pourrait être facilement levée en procédant à des décalages...)

2 Attributs de la classe `EEBool`

Tous les constructeurs de `EEBool` prennent en paramètre un entier positif ou nul n indiquant que l'ensemble créé est un sous-ensemble de l'ensemble $\{0, ..., n - 1\}$.

La classe `EEBool` contient deux attributs :

- un tableau de booléens `ensTabB` de taille n : `this.ensTabB[i]` vaut `true` ssi $i \in this$;
- un entier positif ou nul `cardinal` donnant le cardinal (nombre d'éléments) de l'ensemble.

Ainsi, il existe `this.cardinal` valeurs vraies dans le tableau `this.ensTabB`.

3 Ecriture de la classe `EEBool`

Recoder dans la classe `EEBool` les différentes méthodes de la classe `EE`, au moins quelques constructeurs et les méthodes `toString`, `contient`, `intersection`.

La méthode `deborde` n'étant plus pertinente, on pourra la remplacer par une méthode `enDehors`, qui retourne vrai ssi l'entier donné en paramètre est en dehors de l'intervalle des valeurs possibles des éléments de `this`, c'est-à-dire en dehors des valeurs représentables par `this.ensTabB`.

4 Comparaison des opérations sur les objets de EE et EEBool

Pour comparer les « vitesses » des méthodes de `EE` et `EEBool`, nous allons générer des grands ensembles d'entiers naturels aléatoirement. Un même ensemble mathématique sera à la fois encodé comme un objet de la classe `EE` et comme un objet de la classe `EEBool`. En lançant une même méthode sur ces deux objets (éventuellement plusieurs fois si l'exécution est trop rapide), nous allons pouvoir comparer les performances.

Tous ces tests s'effectuent dans une une nouvelle classe `ComparaisonEE`.

4.1 Méthode genereEE

Ecrire dans la classe `ComparaisonEE` une méthode `genereEE`, prenant comme paramètres le nombre `max` d'éléments dans l'ensemble et une `densite` qui est un nombre réel entre 0 et 1 (inclus). `genereEE` parcourt un à un tous les nombres i de 0 à $max - 1$. Elle décide, avec une probabilité `densite`, si le nombre appartient à l'ensemble. Deux autres paramètres transmis par *résultat* (référence) correspondent aux ensembles créés de type `EE` et `EEBool`.

Indication : tirer un nombre aléatoire

On rappelle les lignes utiles au tirage aléatoire de nombres entiers :

```
import java.util.Random;
Random rand = new Random();
int n = 1000;
int i = rand.nextInt(n); // met dans i un entier aléatoire de 0..n-1
```

4.2 Comparaison des performances

En appelant la méthode `genereEE` avec différentes densités (la faire varier de 0.1 en 0.1 par exemple), comparer différentes méthodes des classes `EE` et `EEBool`, par exemple `toString`, `contient`, `ajoutElt`, `intersection`.

Indication : comment chronométrer ?

Un moyen simple est d'utiliser la méthode (de classe) `currentTimeMillis` de la classe `System`. Exemple :

```
long t1 = System.currentTimeMillis();
EE e3 = e1.intersection(e2);
long t2 = System.currentTimeMillis();
```

`t2` − `t1` donne le temps CPU passé dans la méthode `intersection`.

5 Brève introduction à la complexité en temps

Ces différentes méthodes montrent différentes *complexités en temps* que l'on peut rencontrer très couramment dans les algorithmes existants : la complexité en temps constant, en temps linéaire et en temps quadratique.

La complexité en temps est une fonction qui indique (en sortie) le temps d'exécution en fonction des paramètres d'entrée de notre algorithme. Dans notre exemple, les valeurs d'entrée qui nous intéressent sont `cardinal` (le nombre d'éléments de l'ensemble) et une valeur que l'on appellera `max`, c'est-à-dire `ensTab.length` dans le cas de `EE` et `ensTabB.length` dans le cas de `EEBool`.

Le temps d'exécution ne se mesure pas en secondes (ou autre grandeur temporelle) mais en un nombre d'*opérations élémentaires*, comme des affectations ou des opérations arithmétiques.

On dira que la complexité d'un algorithme est linéaire en n si le nombre d'opérations effectuées par cet algorithme est une fonction affine de n ($a\,n + b$, avec a, b constants). Par exemple, la complexité en temps de `toString` est linéaire en `max` pour `EEBool` et elle est linéaire en `cardinal` pour `EE`.

On dira que la complexité en temps d'un algorithme est constante (on dit aussi que l'algorithme est en temps constant) si elle ne dépend pas des entrées. Expliquer pourquoi `ajoutElt` est en temps constant (dans `EE` comme dans `EEBool`).

La complexité en temps est quadratique si c'est une fonction polynomiale de degré 2 (par exemple, $a\,n^2 + b\,n + c$, avec a, b et c constants).

Question

Donner la complexité en temps de chaque méthode des classes `EE` et `EEBool`.

6 BONUS : Nouvelle classe `EETrous`

Nous allons maintenant implanter une troisième classe d'ensembles d'entiers *naturels*, `EETrous`, qui contient trois attributs :

- `ensTab` qui, comme pour `EE`, est un tableau contenant l'ensemble des éléments "tassés" dans les premiers indices du tableau, dans n'importe quel ordre.
- `cardinal` qui, comme pour `EE`, donne le cardinal (taille) de l'ensemble.
- un tableau `tabIndices` de même taille que `ensTab`). Ce tableau dual de `ensTab` est indicé sur la valeur des éléments de l'ensemble et contient comme élément l'indice où l'élément est rangé dans `ensTab`.

Exemple

Considérons l'ensemble d'entiers {2,4,5,7,8,9}. Une instance de la classe `EETrous` représentant cet ensemble pourrait contenir les attributs suivants (supposons `ensTab.length` valant 10) :

- `ensTab`

Valeur	4	7	9	8	2	5	0	0	0	0
Indice	0	1	2	3	4	5	6	7	8	9

- `cardinal` = 6

- `tabIndices`

Valeur	0	0	4	0	0	5	0	1	3	2
Indice	0	1	2	3	4	5	6	7	8	9

Ecriture de la classe `EETrous`

Recoder dans la classe `EEtrous` les différentes méthodes de la classe `EE`.

Remarque : plutôt que de copier-coller ≪ salement ≫ `EE` pour créer la nouvelle classe, nous aurions préféré spécialiser la classe existante en lui ajoutant un nouvel attribut. C'est ce que permettra de faire la notion centrale de programmation à objets appelée *héritage*... qui sera étudiée dans le module de programmation objet au deuxième semestre.

Comparaison entre `EE`, `EEBool`, `EETrous`

Etendre la comparaison menée à la section 4 à la classe `EETrous`.