Licence Pro 2009/2010
Mathématiques pour l'informatique

A.Théorie des ensembles (1)

1.Introduction

La théorie des ensembles introduite ici est souvent appelée "naïve" (par opposition à "formalisée") car

- d'une part elle s'appuie sur l'intuition et le "bon sens", dont on sait à quel point il peut être trompeur,
- d'autre part, elle mène à des paradoxes mathématiques, qui ne peuvent être levés que par une construction plus rigoureuse des notions introduites.

2.Définition (presque)

Un ensemble est une "collection" d'éléments, pas nécessairement de même nature, que l'on regroupe sous une même appellation. Le mot "collection" est à prendre ici dans le sens commun (il ne fait pas référence à une notion mathématique).

On peut comparer cette notion à celle de structure de données en algorithmique, avec toutefois les restrictions suivantes.

Un ensemble ne comporte **ni ordre**, **ni répétition**.

C'est à dire qu'on ne peut pas parler du premier ou de second, …, élément d'un ensemble, et que chaque élément de l'ensemble n'y figure qu'une seule fois.

3.Création

Il y a essentiellement deux façons de définir un ensemble : en extension et en intension (appelé aussi en compréhension).

- La définition en extension consiste à citer tous les éléments de l'ensemble, entre accolades et séparés par des virgules (ou des points-virgules pour éviter certaines ambiguïtés en français).
  Par exemple l'ensemble { 5 ; 7 ; 2 ; 0 } comporte quatre éléments.
- La définition en compréhension consiste à caractériser les éléments de l'ensemble par une propriété logique ; les éléments de l'ensemble sont les objets de l'univers qui vérifient cette propriété.
  Par exemple $\{ t, t \in \mathbb{N} \text{ et } t<9 \}$ est aussi l'ensemble $\{ 8, 5, 4, 1, 3, 7, 0, 2, 6\}$

4.Prédicats

Appartenance :

- on écrit $x \in A$ pour exprimer le fait que x est l'un des éléments de l'ensemble A. Par exemple $5 \in \mathbb{N}$, $12 \in \{6, 12, 24\}$.
- on écrit $x \notin A$ pour exprimer le fait que x n'est pas l'un des éléments de l'ensemble A. Par exemple $\sqrt{2} \notin \mathbb{N}$, $8 \notin \{6, 12, 24\}$.

Inclusion :

- on écrit $B \subseteq A$ pour exprimer le fait que tous les éléments de l'ensemble B sont aussi des éléments de l'ensemble A.
  On dit que B est un sous-ensemble (une partie) de A
- on écrit $B \not\subset A$ pour exprimer le fait que B n'est pas un sous-ensemble de A ; c'est à dire qu'il existe au moins un élément de B qui n'est pas élément de A.
- on évitera d'utiliser le symbole $\subset$ qui, selon les auteurs, désigne soit la même chose que $\subseteq$, soit l'inclusion stricte (c'est à dire $B \subset A \Leftrightarrow (B \subseteq A \text{ et } A \not\subset B)$)
- on trouvera aussi les symboles $\supset$ et $\supseteq$ pour la contenance (mais le terme "contient" est ambigu, il peut s'appliquer à une partie ou à un élément)

5.Opérateurs

Intersection :

on note $A \cap B$ l'ensemble des éléments communs à A et B : $A \cap B = \{x, x \in A \text{ et } x \in B\}$

Réunion :

on note $A \cup B$ l'ensemble des éléments appartenant à A mélangés avec ceux appartenant à B : $A \cup B = \{x, x \in A \text{ ou } x \in B\}$

Complémentaire :

on note A-B l'ensemble des éléments de A n'appartenant pas à B : $A-B = \{x, x \in A \text{ et } x \notin B\}$

on trouvera aussi les notations $A \backslash B$ ou $\complement_A B$, ou encore $\overline{B}$ lorsque l'ensemble A est sous-entendu (par exemple tout l'univers)

6.Quelques propriétés

Pour tous ensembles A et B,

- on note A=B le fait que les deux ensembles A et B ont les mêmes éléments, c'est à dire que $B \subseteq A$ et $A \subseteq B$
- $A \cap B \subseteq A \subseteq A \cup B$
- $A \cap B = B \cap A$
- $A \cup B = B \cup A$
- $A-B \subseteq A$

Il existe un unique ensemble vide, noté $\varnothing$ ou $\{\}$, qui ne contient aucun élément. Il vérifie les propriétés suivantes :

- $\varnothing \subseteq A$ (en effet, pour que $\varnothing \not\subset A$, il faut montrer qu'il existe un élément de $\varnothing$ qui n'est pas dans A, ce qui est impossible puisque $\varnothing$ n'a pas d'élément)
- $A-A = \varnothing$

7.Cardinal

On appelle cardinal d'un ensemble A le nombre de ses éléments. On le note Card(A) ou #A ou |A|.

Relation fondamentale des dénombrements : pour tous ensembles A et B,

$$Card(A)+Card(B)=Card(A \cup B)+Card(A \cap B)$$

B. Exercices

1.Définir en extension les ensembles suivants :

$\left\{n, n \in N / n < 20 \ \ et \ \ n \ \ pair \ \ et \ \ \frac{n}{2} \ \ impair\right\}$

$\left\{x, x \in N / x < 20 \ \ et \ \ \sin(x * pi / 3) > 0\right\}$

$\left\{lettre, lettre \in \{A..Z\} / "T" + lettre + "IS" \ \ est \ \ un \ \ mot \ \ français\right\}$

$\left\{x \in N, x < 20, x^2 - x - 2 \ \ divisible \ \ par \ \ 5\right\} \ \left\{2 * x, x \in N, x < 20, x^2 - x - 2 \ \ divisible \ \ par \ \ 5\right\}$

$\left\{y \in N, \exists x \in N, x < 20, x^2 - x \equiv 2[5], y = 2 * x\right\}$

2.Définir en intension les ensembles suivants :

$\{1,3,5,7,9,11,13,15\} \quad \{1,4,9,16,25,36,49\}$

$\{a,c,d,l,m,n,o,s,t,y\} \ \{stop, post, pots, tops, spot, opts\}$

$\{1,3,5,7,8,10,12\} \quad \{(1,2),(2,3),(3,4),(4,5)\}$

$\{2,6,12,20\} \quad \{123,234,345,456,567,678,789\}$

3.Montrer que $\overline{A \cap B} = \overline{A} \cup \overline{B}$ *et* $\overline{A \cup B} = \overline{A} \cap \overline{B}$ (lois de De Morgan).

4.Montrer que $A \cap \overline{B} = A \cap \overline{C} \ \Leftrightarrow \ A \cap B = A \cap C$

5.Montrer que $A \subset B \Leftrightarrow A \cap B^C = \varnothing \Leftrightarrow A^c \cap B^c = B^c \Leftrightarrow A^c \cup B = \Omega$

6.Simplifier l'expression

$$\left(\overline{A} \cap \overline{B} \cap \overline{C} \cap \overline{D}\right) \cup \left(\overline{A} \cap \overline{B} \cap \overline{C} \cap D\right) \cup \left(\overline{A} \cap \overline{B} \cap C \cap D\right) \cup \left(\overline{A} \cap B \cap \overline{C} \cap D\right) \cup$$
$$\left(A \cap B \cap \overline{C} \cap D\right) \cup \left(A \cap \overline{B} \cap \overline{C} \cap D\right) \cup \left(A \cap \overline{B} \cap \overline{C} \cap \overline{D}\right) \cup \left(\overline{A} \cap B \cap C \cap D\right) \cup$$
$$\left(A \cap B \cap C \cap D\right) \cup \left(A \cap \overline{B} \cap C \cap D\right) \cup \left(\overline{A} \cap B | C \cap \overline{D}\right)$$

7.Un journal organise un sondage parmi ses abonnés et obtient 1000 réponses parmi lesquelles il y a :
312 hommes, 470 personnes mariées, 525 étudiant(e)s, 42 étudiants (de sexe masculin), 147 étudiant(e)s marié(e)s, 86 hommes mariés et 25 étudiants mariés (de sexe masculin).
Montrer que le dépouillement a été mal fait.

8.Préciser si chacune des affirmations ci-dessous est vraie ou fausse en en donnant une preuve ou un contre-exemple :

$$A \cap B = A \Leftrightarrow A \subset B$$
$$A \cup B = A \Leftrightarrow A \supset B$$
$$A \cap B = A \cup B \Leftrightarrow A = B$$

9.En utilisant les fonctions et procédures ci-dessous, ainsi que des structures conditionnelles ou répétitives, écrire une fonction calculant l'intersection de ses arguments.

```
fonction estvide(E)     % VRAI si et seulement si E est l'ensemble vide %
procédure vider(E)              % E prend pour valeur l'ensemble vide %
procédure mettre(x,E)                    % ajoute x à l'ensemble E %
procédure prendre(x,E)        % met dans x un élément de E et l'en ôte %
```

10.Formaliser dans la théorie des ensembles le problème :
En lançant k dés à six faces (comportant les nombres 1, 5, 9, 18, 35, 41) quelles sommes peut-on obtenir ?

C. Théorie des ensembles (2)

11.ensemble des parties

Si E est un ensemble, on appelle partie (ou sous-ensemble) de E tout ensemble A vérifiant $A \subseteq E$.
On appelle ensemble des parties d'un ensemble E l'ensemble, noté $\wp(E)$, défini comme $\{A, A \subseteq E\}$.
Par suite, on a l'équivalence :

$$A \subseteq E \Leftrightarrow A \in \wp(E)$$

De plus, pour constituer une partie A de E, il faut choisir pour chaque élément de E si on décide de le prendre ou pas dans A. On peut donc trouver toutes les parties de E en construisant un arbre binaire de profondeur #E. On en déduit que :

- d'une part $\#\wp(E)=2^{\#E}$,
- d'autre part, on peut ainsi associer à chaque partie A de E un entier en base 2 sur #E bits.

12.fonction caractéristique

Si E est un ensemble, on appelle fonction caractéristique de E la fonction définie sur l'univers, généralement notée χ_E et qui à un élément x associe 1 si $x \in E$, et 0 sinon.

D. Exercices

13.On définit ainsi la différence symétrique : $A\Delta B =_{déf} (A \cup B) - (A \cap B)$.
Montrer que $A\Delta B = (A - B) \cup (B - A)$et simplifer $(A\Delta B)\Delta(A \cap B)$

14.Soit A l'ensemble {2,3,5,7} et B l'ensemble {1,3,7}. Construire $\mathcal{P}$(A), $\mathcal{P}$(B), $\mathcal{P}$(A∩B), $\mathcal{P}$(A)∩$\mathcal{P}$(B), $\mathcal{P}$(A∪B), $\mathcal{P}$(A)∪$\mathcal{P}$(B). Généraliser les égalités trouvées à des ensembles quelconques.

15.On définit la suite A_i par $A_0 = \varnothing$ et $A_{i+1} = \mathcal{P}(A_i)$.

- donner en extension les cinq premiers A_i.
- pour des ensembles E quelconques, peut-on écrire et a-t-on $E \in E, E \subset E, E \in \mathcal{P}(E), E \subset \mathcal{P}(E)$?
- dans le cas particulier des A_i, rediscuter de ces formules.
- calculer $A_i \cap A_j, A_i \cup A_j$ pour $i, j \in N$
- en déduire une structure de données et des algorithmes simples pour calculer union et intersection des A_i.

16.Une famille d'ensembles A est dit stable par intersection si $\forall E, F \in \mathsf{A}, E \cap F \in \mathsf{A}$.
On notera $I(\mathsf{A}) = \{C \cap D, C \in \mathsf{A}, D \in \mathsf{A}\}$.

a)Pour $E = \{a,b,c,d\}$ et $X = \{\{a,b,d\},\{a,b,c,d\},\{a,c,d\},\{b,c,d\}\}$ et $Y = \{\varnothing,\{a,b,c\},\{b\},\{c\}\}$, donner l'ensemble F des parties de E, une partition G non triviale de E, un recouvrement H de E qui ne soit pas une partition et calculer $I(\mathsf{F}), I(\mathsf{G}), I(\mathsf{H})$.

b)Déterminer si F, G, $I(\mathsf{G})$, H, X ou Y est stable par intersection.

c)Pour une famille quelconque A de parties d'un ensemble quelconque E, déterminer si A, $I(\mathsf{A})$ ou $\mathsf{P}(\mathsf{A})$est stable par intersection.

d)Pour une famille quelconque A de parties d'un ensemble quelconque E, déterminer si $\mathsf{A} \subset I(\mathsf{A})$, si $I(\mathsf{A}) \subset \mathsf{A}$, si $\mathsf{A} = I(\mathsf{A})$, si $I(\mathsf{A}) = I(I(\mathsf{A}))$. Montrer que pour n assez grand, $I^{n+1}(\mathsf{A}) = I^n(\mathsf{A})$.

e)Si une famille A de parties de E vérifie $\mathsf{A} = I(\mathsf{A})$, montrer qu'elle est stable par intersection.

Complexité

A. Comptage manuel

On considère les quatre fonctions suivantes :

```
fonction tq(nb entier) renvoie un entier
entiers toto, cpt
début
 toto ← 0 ; cpt ← 0
 tant que cpt ≤ nb faire
  toto ← toto+cpt; cpt ← cpt+1
 fin tant que
 renvoyer(toto)
fin
```

```
fonction pr(nb entier) renvoie un entier
entiers toto, cpt
début
 toto ← 0
 pour cpt de 1 à nb faire
  toto ← toto+cpt
 fin pour
 renvoyer(toto)
fin
```

```
fonction prpr(nb entier) renvoie un entier
entiers toto, cpt, ind
début
 toto ← 0
 pour cpt de 1 à nb faire
  pour ind de 1 à cpt faire
   toto ← toto+1
  fin pour
 fin pour
 renvoyer(toto)
fin
```

```
fonction rpt(nb entier) renvoie un entier
entiers toto, cpt
début
 toto ← 0
 cpt ← 0
 répéter
  cpt ← cpt+1 ; toto ← toto+cpt
 jusqu'à toto > nb
 renvoyer(toto)
fin
```

Donner le résultat de chacun des appels : `tq(0), pr(-1), prpr(0), rpt(-2), tq(4), pr(5), prpr(3), rpt(2).`

Traduire ces fonctions en Java en s'assurant que les appels précédents donnent bien le même résultat.

Effectuer à la main l'exécution de chacun des appels ci-dessous en comptant au fur et à mesure le nombre de temps élémentaires *t*.

`tq(1), tq(2), tq(3), pr(1), pr(2), pr(3), prpr(1), prpr(2), prpr(3), rpt(1), rpt(2), rpt(3)`

Tenter de généraliser ensuite pour des appels appliqués à *n* quelconque.

B. Suites

Écrire une fonction prenant en entrée un entier n et retournant la valeur du terme d'indice n de la suite définie par $u_0=1$ et $u_n=3u_{n-1}+2$.

Calculer son coût.

Écrire une fonction calculant pour n passé en paramètre la somme des n premiers termes de la suite précédente. On envisagera plusieurs solutions, que l'on comparera après calcul de leurs complexités.

C. Mini et Maxi

A) Écrire une fonction prenant en entrée un tableau d'entiers (d'indices 1 à N) et retournant le plus petit écart que l'on puisse former en soustrayant une valeur d'une case du tableau à une valeur d'une autre case. Calculer un minorant et un majorant de son coût.

A bis) Réécrire cette fonction en supposant le tableau trié par ordre croissant. Estimer de la même façon son coût.

B) Écrire une fonction prenant en entrée un tableau d'entiers (d'indices 1 à N) et retournant cette fois le plus grand écart que l'on puisse former en soustrayant deux valeurs contenues dans le tableau. Calculer un minorant et un majorant de son coût.

B bis)Réécrire cette fonction en supposant le tableau trié par ordre croissant. Estimer de la même façon son coût.

B ter) En déduire une fonction rapide même sur un tableau non trié.

D. Coût moyen

On considère la fonction suivante donnant le «et» d'un tableau de booléens.

```
// T est un tableau (indices 1 à n) de booléens
static boolean et(boolean[] T, int n){
            int i=1;
            while (i<n && T[i] ) {i++;}
            return T[i];
}  // le résultat est Faux si 1 élément au moins de T est Faux
```

a) Calculer le coût de cette fonction pour chaque tableau de 2 éléments. En déduire le coût au mieux, au pire et en moyenne pour la taille 2. Donner les formes des données correspondantes

b) Calculer de même pour chaque tableau de taille 3.

c) généraliser pour un tableau de taille n.

E. À demi !

On considère un tableau de réels (de type NOTES) dont on utilise les cases 1 à N. On veut compter le nombre de valeurs supérieures ou égales à 10 dans ce tableau.

Écrire une fonction prenant en paramètre le tableau et N et donnant ce nombre.
Calculer le coût de l'appel de cette fonction sur le tableau ci-dessous.

11	8	15	9	7	3	12	9	17	5

a) Pour quel(s) tableau(x) à 10 éléments le coût de la procédure est-il le plus grand ? Et que vaut-il ?
b) Pour quel(s) tableau(x) à 10 éléments le coût de la procédure est-il le plus petit ? Et que vaut-il ?
c) Pour N quelconque, pour quelle forme des données aura-t-on le coût au pire ? Que vaut ce coût en fonction de N ?
d) Pour N quelconque, pour quelle forme des données aura-t-on le coût au mieux ? Que vaut ce coût en fonction de N ?

F. Habit de recherche

On considère un tableau (de type TAB, le type des éléments n'est pas précisé — n'est pas important) dont on utilise les cases 1 à N. On désire écrire une procédure de recherche d'un élément dans ce tableau qui renvoie un booléen (valant VRAI si l'élément figure dans le tableau, FAUX sinon) et un entier : la position de l'élément dans le tableau.

a) Écrire une telle procédure en utilisant une boucle pour. Étudier son coût. Discuter du cas des occurences multiples.
b) Améliorer la procédure précédente en utilisant une autre boucle. Étudier son coût. Discuter des occurences multiples.
c) Améliorer la procédure précédente en supposant le tableau trié. Étudier son coût. Discuter des occurences multiples.
d) Toujours dans le cas d'un tableau trié, on peut utiliser la recherche dichotomique : on compare l'élément cherché avec l'élément au centre du tableau, ce qui permet de restreindre la recherche à une moitié du tableau, puis on recommence avec cette moitié de tableau : on compare l'élément cherché avec celui au centre de cette moitié, ce qui permet de se restreindre à un quart du tableau, et ainsi de suite. Étudier le coût d'une telle procédure.

G. Notations asymptotiques

On classifie les fonctions dont la limite en $+\infty$ est $+\infty$ en fonction de leur comportement à l'infini :
on définit

$$O(g) = \left\{ fonctions \quad f / \exists c \in R^{+*}, \exists n_0 \in N, \forall n \geq n_0, 0 \leq f(n) \leq c.g(n) \right\}$$
$$o(g) = \left\{ fonctions \quad f / \forall c \in R^{+*}, \exists n_0 \in N, \forall n \geq n_0, 0 \leq f(n) \leq c.g(n) \right\}$$
$$\theta(g) = \left\{ fonctions \quad f / \exists c_1, c_2 \in R^{+*}, \exists n_0 \in N, \forall n \geq n_0, c_1.g(n) \leq f(n) \leq c_2.g(n) \right\}$$
$$\Omega(g) = \left\{ fonctions \quad f / \exists c \in R^{+*}, \exists n_0 \in N, \forall n \geq n_0, 0 \leq c.g(n) \leq f(n) \right\}$$

On écrit traditionnellement f=O(g) au lieu de $f \in O(g)$, et on lit "f est un grand O de g". Intuitivement, f=θ(g) signifie que f et g varient de la même façon à l'infini, à une constante multiplicative près. Par exemple un polynôme anxn+an-1xn-1+…+a0 est un θ(xn). On peut, sous certaines conditions de régularité sur f et g, écrire que :

$$f = \theta(g) \Leftrightarrow \lim_{n \to +\infty} \left(\frac{f(n)}{g(n)}\right) = c > 0 \quad \text{càd } f \text{ et } g \text{ varient de la même façon}$$
$$f = O(g) \Leftrightarrow \lim_{n \to +\infty} \left(\frac{f(n)}{g(n)}\right) = c \geq 0 \quad \text{càd } f \text{ ne varie pas plus vite que } g$$
$$f = o(g) \Leftrightarrow \lim_{n \to +\infty} \left(\frac{f(n)}{g(n)}\right) = 0 \quad \text{càd } f \text{ varie moins vite que } g$$
$$f = \Omega(g) \Leftrightarrow \lim_{n \to +\infty} \left(\frac{f(n)}{g(n)}\right) > 0 \ (\textit{éventuellement} \ +\infty) \quad \text{càd } f \text{ varie plus vite que } g$$

a) Comparer entre elles (avec les notations ci-dessus) les fonctions suivantes de la variable n :

$2n+1$, $7n-333$, $\ln(n)$, $\log(n)$, $\log(n+1)$, $\log(n^2)$, $\sqrt{n}$, e^n,n^2,n^3,n^4, $n!$, $\exp(n^2)$,e^{2*n},e^{n+1}

b) trouver des fonctions dans les classes suivantes :

$o(\log(n))$, $\Omega(e^n)$, $o(1/n)$

Étude des algorithmes : structures de données

A. Piles/Files

On dispose du type Pile spécifié ci-dessous.

Signatures
pileVide : → Pile
empiler : Elem × Pile → Pile
sommet : Pile → Elem
depiler : Pile → Pile
estPileVide : Pile → Booléen

Axiomes
depiler(pileVide) = ERREUR.
depiler(empiler(e, p)) = p
sommet(pileVide) = ERREUR.
sommet(empiler(e, p)) = e
estPileVide(pileVide) = VRAI
estPileVide(empiler(e, p)) = faux

Prouver que :
`sommet(depiler(empiler(e3,depiler(empiler(e2,empiler(e1,p))))))  = e1`

Puis simplifier :
`depiler(empiler(7,depiler(empiler(sommet(empiler(5,empiler(3,p))),empiler(9,q)))))`

Proposez l'implémentation de deux piles sur un même vecteur de taille [1..n] de telle manière qu'il n'y ait aucun débordement de pile sauf si la somme des nombres d'éléments des deux piles dépasse n.

On dispose du type File spécifié ainsi :

Signatures
fileVide : → File
enfiler : Elem × File → File
bout : File → Elem
defiler : File → File
estFileVide : File → Booléen

Axiomes
defiler(fileVide) = ERREUR.
defiler(enfiler(e, fileVide)) = fileVide
defiler(enfiler(a, enfiler(b, p))) = defiler(enfiler(b, p))
bout(fileVide) = ERREUR.
bout(enfiler(e, fileVide)) = e
bout(enfiler(a, enfiler(b, p))) = bout(enfiler(b, p))
estFileVide(fileVide) = VRAI
estFileVide(enfiler(e, p)) = FAUX

Trouver des conditions pour que :
`enfiler(e4,defiler(enfiler(e3,defiler(defiler(enfiler(e2,enfiler(e1,f)))))))=enfiler(e4,f)`

a) Montrez comment implémenter une pile à l'aide de deux files.

b) Montrez comment implémenter une file à l'aide de deux piles.

B. Listes

On dispose du type Liste spécifié ci-dessous.

Signatures
listeVide : → Liste
cons : Elem × Liste → Liste
prem : Liste → Elem
suite : Liste → Liste
estListeVide : Liste → Bool

Axiomes
suite(listeVide) = ERREUR.
suite(cons(e, L)) = L
prem(listeVide) = ERREUR.
prem(cons(e, L)) = e
estListeVide(listeVide) = VRAI
estListeVide(cons(e, L)) = FAUX

Spécifier algébriquement puis proposer des implémentations (en en calculant le cout) de fonctions donnant :
le nombre d'éléments d'une liste
le nombre d'occurences d'un élément dans une liste
l'indice de la première occurence d'un élément dans une liste
la concaténation de deux listes
le retournement d'une liste
la concaténation de deux listes

C. Ensembles

Donner une spécification algébrique des ensembles d'entiers puis spécifier et implémenter des fonctions donnant :

le nombre d'éléments d'un ensemble
l'union de deux ensembles
la différence symétrique de deux ensembles
VRAI si et seulement si les deux ensembles sont égaux

Des A-List au H-code

1. Association-List

Une A-List est une liste de couples (clé,valeur) vérifiant :

• chaque clé n'apparaît que dans un seul couple, par contre une même valeur peut être dans plusieurs couple ;

• l'ajout d'un couple avec une clé existante a pour effet de remplacer le couple ayant déjà cette clé.

Étudier des implémentations (sous forme de liste chaînée) de A-List dont les clés sont des entiers et les valeurs des chaînes de caractères et disposant des méthodes suivantes :

```
boolean contientClef(int k)       // vrai ssi la clef k figure dans un couple

boolean contientValeur(String v)  // vrai ssi la valeur v figure dans au moins un couple

String valeurAssociee(int k)      // l'unique valeur associée à la clef k (supposée exister)

void mettreValeur(int k, String v)      // ajoute le couple (k,v), ou remplace un éventuel (k,.)

void supprimerClef(int k)         // supprime l'éventuel couple (k,.)
```

Envisager deux versions selon que les clefs sont ordonnées ou pas et discuter dans chaque cas du coût de chaque méthode.

2. Tableaux associatifs

Etudier des implémentations de ces mêmes A-List sous forme de deux tableaux statiques :

• un tableau de clefs permettant d'associer à chaque clef un indice

• un tableau de valeur donnant la valeur associée à un indice

Par exemple, une A-List comportant les trois couples (7,"Alice") , (15,"Chantal") , (23,"Benoit") pourra être stockée dans les deux tableaux :

1	7		1	Alice
2	15		2	Chantal
3	23		3	Benoit
4			4	
5			5	

Et l'accès à la valeur associée à la clef 15 se fait en deux temps : d'abord trouver l'indice de la clef 15 dans le tableau de clefs (ici 2) puis retourner la valeur de même indice dans le tableau des valeurs.

Discuter de l'encombrement mémoire de la structure et des coûts des méthodes. Comment gérer le cas où l'on veut mettre plus de clefs que de places prévues dans le tableau ?

3. Table de Hashage

Plutôt que de prévoir un grand tableau pour les clefs, on définit une fonction (appelée fonction de hashage) qui calcule l'indice à partir de la clef. Dans l'exemple précédent, on pourrait utiliser par exemple la fonction qui à n associe (1+ n mod 7).

De plus, pour ne pas occuper trop de place mémoire, on ne va réserver qu'une taille raisonnable de tableau pour les valeurs, et stocker dans chaque case une liste chaînée de couples (clef,valeur).Ainsi la fonction de hashage peut donne le même indice pour deux clefs différentes, il faudra alors passer en revue la liste pour retrouver le bon couple.

Trois valeurs permettent d'étudier le coût d'une table de hashage :

• sa capacité : le nombre de cases prévues dans le tableau ;

• sa largeur ou profondeur : la taille de la plus longue liste chaînée contenue dans le tableau ;

• son taux de remplissage : rapport du nombre de couples stockés sur la capacité.

Discuter de l'encombrement mémoire de la structure et des coûts des méthodes.

Dictionnaire

4. Introduction

On désire implémenter un dictionnaire, c'est à dire une structure de données non ordonnée avec unicité. Pour simplifier le problème, les éléments à stocker seront des chaînes de caractères, mais tout type disposant d'un ordre lexicographique conviendrait. À chaque mot est associé une valeur (par exemple le même mot dans une autre langue, ou la nature grammaticale — nom, verbe, adjectif, etc)

Les méthodes à définir sont :

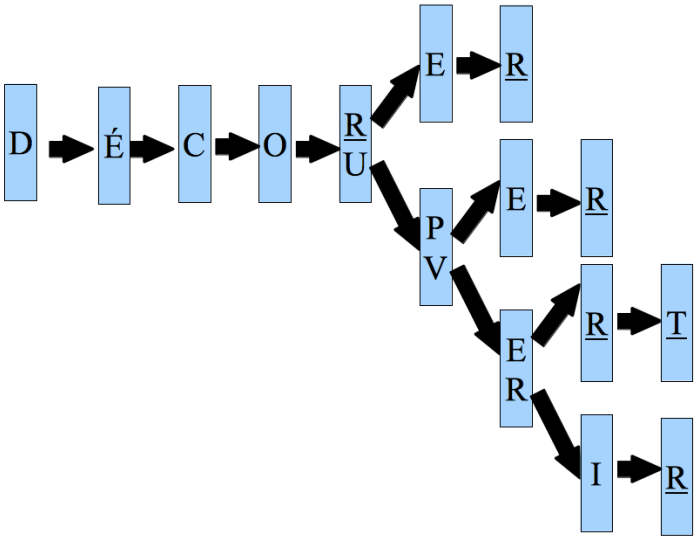
```
boolean contientMot(String mot)
// vrai ssi la chaîne mot figure dans le dictionnaire

void mettreMot(String mot, Valeur v)
// ajoute la chaîne mot au dictionnaire, avec la valeur v

void supprimerMot(String mot)
// supprime l'éventuelle chaîne mot du dictionnaire

Valeur valeurAssociée(String mot)
// donne la valeur correspondant à la chaîne mot
// (supposée figurer dans le dictionnaire)
```

Envisager l'implémentation d'une telle structure sous forme de A-list puis de table de hashage. Comparer les coûts et l'encombrement mémoire en remarquant pour les comparaisons (equals) et pour la fonction de hashage que beaucoup de mots en Français ont un début commun (par exemple découvrir, découper, décorer, découvert, découver).

5. Arbre des préfixes

Il est possible de gagner de l'encombrement mémoire en ne stockant qu'une fois les préfixes communs. On va donc construire une structure arborescente dans laquelle chaque cellule contient la « liste » des lettres possibles pour la suite du mot, et pour chaque lettre, une nouvelle cellule du même type.

Par exemple, avec les mots « décor, découvrir, découper, décorer, découvert, découver », le dictionnaire pourrait ressembler à (les lettres soulignées correspondent à une fin de mot) :



Chaque cellule comporte une suite de lettre, chacune associée à une nouvelle cellule. Proposer plusieurs implémentations de ce type selon la façon de stocker la suite de lettres (tableau statique, A-liste, table de hashage).

Discuter de l'encombrement mémoire de la structure et des coûts des méthodes en fonctions des implémentations.