Validation d'algorithmes
L3 MIAGE, UGA
2020/2021

Raphaël Charrondière
raphael.charrondiere@inria.fr
Semaine 4 - échéance S5

Feuille de TD 2,5: Des exemples de variants et d'invariants

Où il faut maitriser le chapitre 1 et 2.

Ce TD est à rendre individuellement.

Exercice 1 Un algorithme, des variants

```
   Entrées : a, b ∈ ℕ
1  tant que a > 0 ∧ b > 0 faire
2  |  r ← BooleanAleatoire();
3  |  si r alors
4  |  |  a ← a − 1;
5  |  sinon
6  |  |  b ← b − 1;
7  |  fin
8  fin
9  retourner a + b
```

Algorithme 1 : Course de billes

1. *(Cours)* Comment montre-t-on qu'une boucle termine ?
2. Donner un variant de la forme $\varphi : \mathbb{M} \to \mathbb{N}$ pour l'algorithme 1 et montrer que c'est bien un variant.
3. On défini l'ordre $<_\times$ sur les paires d'entiers par $\forall a_1, a_2, b_1, b_2 \in \mathbb{N},\ (a_1, a_2) <_\times (b_1, b_2) \Leftrightarrow a_1 \leqslant b_1 \wedge a_2 \leqslant b_2 \wedge (a_1 \neq b_1 \vee a_2 \neq b_2)$. Montrez que $<_\times$ est un ordre strict bien fondé.
4. On se place maintenant dans l'ensemble des paires d'entiers muni de l'ordre $<_\times$, montrer que $\psi : m \mapsto (m(a), m(b))$ est bien un variant pour l'algorithme 1 .

Exercice 2 Champ de bataille

On décrit informellement l'algorithme suivant : on a une grille d'entiers naturels ($\mathbb{N}$) de taille finie (c'est-à-dire un tableau 2D). L'algorithme cherche une case où appliquer une règle R, il l'applique et recommence jusqu'à ne plus trouver de case où appliquer la règle. Les voisins d'une case, sont les 4 cases adjacentes par les arrêtes de la case.

1. On propose la règle R suivante : Si la valeur de la case n'est pas zéro et qu'une des cases voisines a une valeur strictement supérieure, on décrémente la valeur de la case. Proposer un variant pour prouver que l'algorithme fini.
2. On choisi maintenant la règle suivante : si la case A a une valeur v_A et un de ses voisins a une valeur strictement inférieur alors on décrément la valeur de A et on fixe la valeur de ses voisins a $v_A - 1$. Proposer un variant pour montrer que l'algorithme fini.
3. Peut-on généraliser ? C'est-à-dire, quelque soit la règle R, l'algorithme finira-t-il ? *Justifiez votre réponse !*

Raphaël Charrondière
raphael.charrondiere@inria.fr

Exercice 3 Graphes et cycles

Dans cet exercice, les graphe sont orientés. On rappelle qu'un graphe orienté $\mathcal{G}$ se définit par $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ un ensemble de nœuds et d'arcs. Les nœuds sont des entiers $\mathcal{N} = [\![1, k]\!]$. Les arcs sont des paires d'entiers $(u, v) \in \mathcal{N} \times \mathcal{N}$. Cherchez la définition de graphes dans n'importe quel encyclopédie/cours pour plus de détails. On propose l'algorithme suivant pour connaitre les nœuds appartenant à un cycle.

```
   Entrées : G un graphe
   Output : La liste des nœuds appartenant à un cycle
 1 N ← l'ensemble des nœuds de G;
 2 A ← l'ensemble des arcs de G;
 3 n ← nombre de nœuds de G;
 4 r ← ∅;
 5 t ← Tableau2D(n,n);
 6 Remplir t avec des 0;
 7 pour i de 1 à n faire
 8 |   pour (u, v) ∈ A faire
 9 |   |   pour n ∈ N faire
10 |   |   |   si t[n, u] == 1 OU n == u alors
11 |   |   |   |   t[n, v] ← 1;
12 |   |   |   fin
13 |   |   fin
14 |   fin
15 fin
16 pour i de 1 à n faire
17 |   si t[n, n] == 1 alors
18 |   |   r ← r ∪ {n};
19 |   fin
20 fin
21 retourner r
```

Algorithme 2 : Détection de cycles

1. À quoi correspond le tableau t ?
2. On peut découper le code en trois blocs : ligne 1 à 6 , (B1) ligne 7 à 15 une première boucle et (B2) ligne 16 à 20. On ne va pas rentrer dans les détails des boucles imbriquées ligne 8 à13. De même les variants de boucles ne sont pas très intéressants. Donner un invariant pour la boucle du bloc (B1). *Bien sûr justifiez votre réponse en expliquant clairement. Il n'est pas utile de trop rentrer dans les détails.*
3. Donner un invariant pour la boucle du bloc (B2). *Cette question doit être très facile si vous avez compris l'algorithme et en particulier le rôle de t.*

Exercice 4 Plus longue sous-suite

Soit deux mots A=$a_1a_2\dots a_k$, B=$b_1b_2\dots b_m$, une plus longue sous-suite commune (abrégée PLSSC) est un mot C=$c_1\dots c_n$ tel que C est une sous-suite de A et une sous-suite de B, et pour tout autre sous-suite commune D, C sera plus longue que D.

Une sous-suite de A est le mot A auquel on a retiré des lettres

Par exemple «ananas» et «amnésie» ont pour plus grande sous-suite commune «ans». De manière plus abstraite «abcbdab» et «bdcaba» ont pour PLSSC «bcab» ou «bdab».

On propose l'algorithme suivant :

Raphaël Charrondière
raphael.charrondiere@inria.fr


```
   Entrées : a, b des mots
   Output : Un tableau t
 1 n ←longueur(a);
 2 m ←longueur(b);
 3 t ← Tableau2D(n+1,m+1);
 4 Remplir t avec des 0;
 5 pour i de 1 à n faire
 6 |   pour j de 1 à m faire
 7 |   |   compLettre ← a[i] == b[j];
 8 |   |   t[i, j] = max(t[i − 1, j], t[i, j − 1], t[i − 1, j − 1] + compLettre);
 9 |   fin
10 fin
11 retourner t
```

Algorithme 3 : Longueur PLSSC

1 Prouvez que l'algorithme 3 termine

2 Prouver que dans le tableau renvoyé t[i,j] est la longueur du PLSSC des sous-mots $a_0 \dots a_i$ et $b_0 \dots b_j$

On utilise maintenant l'algorithme suivant :

```
   Entrées : a, b et t donnés en entrée et sortie de l'algorithme 3
   Output : Le PLSSC de a et b
 1 n ←longueur(a);
 2 m ←longueur(b);
 3 k ← t[n, m];
 4 w ← Mot de longueur(k);
 5 i ← n;
 6 j ← m;
 7 tant que k > 0 faire
 8 |   si t[i − 1, j] == t[i, j] alors
 9 |   |   i ← i − 1;
10 |   sinon si t[i, j − 1] == t[i, j] alors
11 |   |   j ← j − 1;
12 |   sinon
13 |   |   w[k] ← a[i];
14 |   |   i ← i − 1;
15 |   |   j ← j − 1;
16 |   |   k ← k − 1;
17 |   fin
18 fin
19 retourner w
```

Algorithme 4 : Longueur PLSSC

3 Montrer que l'algorithme 4 termine.

4 Montrer que l'algorithme 4 renvoie ce qu'il faut, c'est à dire le PLSSC.