Validation d'algorithmes
L3 MIAGE, UGA
2020/2021

Raphaël Charrondière
raphael.charrondiere@inria.fr
Semaine 5 - échéance S6

Correction TD 3: Graphes de flot de contrôle

Où l'on commence à manipuler les GFC.

```
read(i);
j ← 0;
a ← 0;
b ← 1;
tant que j < i faire
    b ← a + b;
    a ← b − a;
    i ← i + 1;
fin
retourner a;
```

Algorithme 1 : Fibonacci

```
productID ← readInput();
dureeVie ← getExpiration(productID)-clock();
dureeConservation ← getDureeConservation(productID);
stock ← getStock(productID);
reduction← 0;
si dureeVie<3 jours alors reduction ← 10;
si dureeVie<2 jours alors reduction ← 40;
si dureeVie<10 jours ET dureeConservation>30 jours alors
  reduction ← 50;
si stock<10 alors reduction ← 0;
si dureeVie<1 jour alors reduction ← 70;
retourner reduction;
```

Algorithme 2 : Réduction

Exercice 1 Fibonacci

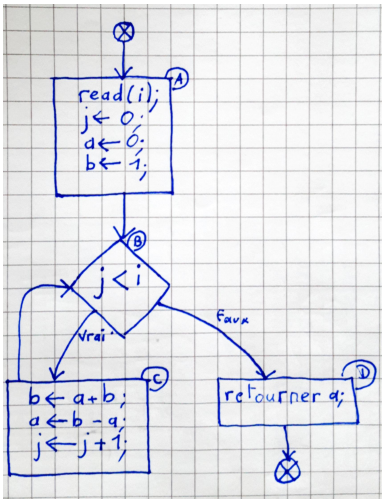
La suite de Fibonacci se définit de la manière suivante : $\mathcal{F}_0 = 0$, $\mathcal{F}_1 = 1$ et $\forall i > 1,\ \mathcal{F}_i = \mathcal{F}_{i-1} + \mathcal{F}_{i-2}$. Les premiers termes de la suite sont $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89$. On propose le programme suivant pour calculer l'i-ème terme :

(1) Vérifier que l'algorithme 1 renvoie bien ce qu'il doit renvoyer. *Vous avez vu comment faire dans le cours et les TD précédents, non ?*

`Correction :` On se rend compte qu'il y a une erreur dans l'énoncé. Il faut remplacer $i \leftarrow i + 1$ par $j \leftarrow j + 1$. On nous demande un invariant. On propose $I(m_0, m) \equiv a = \mathcal{F}_i \wedge b = \mathcal{F}_{i+1}$ (rappel on note $m(a)$ par a et $m_0(a)$ par a_0), par initialisation $I(m_0, m_0)$ est vrai. Soit m_1 l'état mémoire en début d'itération de boucle et m_2 l'état mémoire à la fin de cette itération. On suppose $I(m_0, m_1)$. On a $b_2 = a_1 + b_1 = \mathcal{F}_{i_1} + \mathcal{F}_{i_1+1} = \mathcal{F}_{i_1+2} = \mathcal{F}_{i_2+1}$ et $a_2 = b_2 - a_1 = b_1 = \mathcal{F}_{i_1+1} = \mathcal{F}_{i_2}$. Quand la boucle s'arrête, on a $j = i$ d'où d'après l'invariant $a = \mathcal{F}_j$.

(2) Donner le graphe de flot de contrôle de l'algorithme.

`Correction :`



(3) Donner les chemins (sous forme algébrique) de ce graphe.

`Correction :` A(BC)*BD

(4) Proposer des données de test pour couvrir le critère « tous les nœuds ».

`Correction :` On propose la DT i=1, cela donnera le chemin ABCBD qui passe par tout les nœuds

(5) Proposer des données de test pour couvrir le critère « tous les arcs ».

Raphaël Charrondière
raphael.charrondiere@inria.fr

`Correction :` La même DT peut être utilisée, le chemin ABCBD passe par tout les arcs

---

Exercice 2 Les arcs sont des nœuds

Cet exercice a été fait en cours, mais il est bon de le refaire pour valider ses acquis.

(1) Montrer qu'une couverture par arcs implique une couverture par nœud.

`Correction :` Dans un GFC, aucun nœud n'est isolé : en effet, le nœud d'entrée est relié à la première instruction et chaque commande est suivie d'un autre nœud chaque condition a deux arcs sortant, donc chaque nœud a un arc sortant, sauf le nœud de sortie qui est forcément relié a un autre nœud.

S'il y a une couverture par arc pour un ensemble de tests :

Pour tout nœud n : Ce nœud est relié un arc a. Or la couverture par arc implique qu'il existe une DT dont le flot passera par a, le flot passera par n, puisque c'est une des extrémités de a.

Alors tout nœud est couvert, il y a couverture par nœud

(2) Montrer que l'inverse n'est pas vrai, mais donner une technique pour le rendre vrai.

`Correction :` Il y a un contre exemple en cours (slide 23)

Pour rendre l'implication inverse vrai il faut une propriété supplémentaire sur le GFC : chaque nœud suivant un nœud de décision n'a qu'un seul arc entrant. On note qu'on peut ajouter des nœuds commandes vides pour cela.

Avec cette propriété supplémentaire, si les tests couvrent tout les nœuds : pour tout arc a, si a sort du nœud d'entrée il sera couvert, sinon si a sort d'un nœud commande, le nœud étant couvert il y a une DT pour lequel le flot passe par ce nœud, il va forcément passer a, vu qu'un nœud de commande n'a qu'un seul arc sortant, sinon a sort d'une condition a rentre dans un nœud n, or la propriété indique que n n'a qu'un arc entrant, n est couvert donc il y a une DT dont le flot passe par n, le flot ne peut que passer par a pour rentrer dans n, donc a est couvert.

Exercice 3 Enchaîner les ifs

Un magasin veut inciter les clients à acheter les produits alimentaire en fin de vie en premier pour éviter les pertes. Il baisse alors les prix suivants plusieurs critères, en considérant la date de péremption et le stock disponible. Si plusieurs critères sont applicables, le dernier critère de la liste est appliqué

C1 : Si la date de péremption est dans les trois jours, le produit reçoit une réduction de 10%

C2 : Si la date de péremption est dans les deux jours, le produit reçoit une réduction de 40%

C3 : Si la date de péremption est dans les dix jours et qu'il s'agisse d'un article à longue conservation, le produit reçoit une réduction de 50%

C4 : Si le stock est faible le produit n'est pas diminué

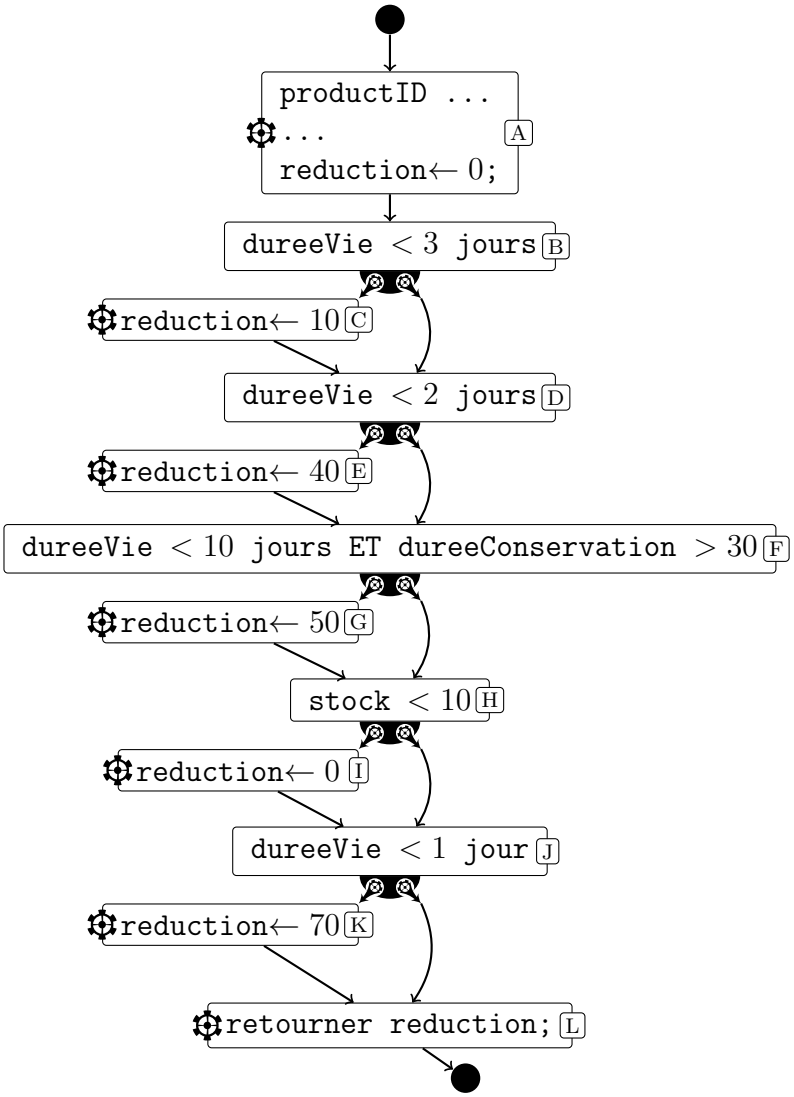
C5 : Si la date de péremption est le jour même, le produit reçoit une réduction de 70%

Le magasin veut coder un programme pour afficher la réduction à appliquer. Prenons la solution la plus simple, l'algorithme 2.

(1) Donner le graphe de flot de contrôle.

`Correction :` Note : en suivant les conventions du cours, le branche vrai est toujours à droite, mais rien ne vous empêchait de ne pas suivre cette convention si vous indiquez le rôle des branches.

Raphaël Charrondière
raphael.charrondiere@inria.fr



2 Donner des données de test pour couvrir le critère "tous les arcs"

`Correction :` En utilisant part exemple les données de test

— [dureeVie=0, dureeConsevation =31, stock =3]

— [dureeVie=5, dureeConsevation = 10, stock = 30]

on couvre tout les arc (ici, toutes les conditions sont vraies pour la première DT, et fausses pour la seconde). Cette réponse n'est qu'un exemple, et il n'y a pas de réponse unique. On pourrait aussi proposer

— [dureeVie=5, dureeConsevation =31, stock =3]

— [dureeVie=0, dureeConsevation =5, stock =7]

— [dureeVie=2, dureeConsevation = 10, stock = 24]

3 Écrire l'expression algébrique des chemins.

`Correction :` En réutilisant la notation du cours A ? qui signifie zéro ou une fois A, on obtient ABC ?DE ?FG ?HI ?JK ?L (Note : plusieurs syntaxes sont possibles, par exemple A(B+BC)(D+DE)...)

4 Pour chaque chemin donner une donnée de test (on supposera que `clock()` renvoie le 1/1/2021). *Indice* : Regardez la question suivante.

`Correction :` Note : cette correction a été faite l'année dernière, d'où la remarque suivante, "Personne n'a voulu utiliser la date d'expiration, pourtant c'était ce qu'il fallait utiliser et non dureeConservation. Je vais faire comme tout le monde du coup". Certains chemins n'ont pas de données de tests, (par exemple si dureeVie< 2 alors dureeVie< 3)

— `ABCDEFGHIJKL` [dureeVie=0, dureeConsevation =31, stock =3]

— `AB DEFGHIJKL` Pas de DT

Raphaël Charrondière
raphael.charrondiere@inria.fr

— `ABCD FGHIJKL` Pas de DT
— `ABCDEF HIJKL` [dureeVie=0, dureeConsevation = 10, stock = 3]
— `ABCDEFGH JKL` [dureeVie=0, dureeConsevation = 100, stock = 30]
— `ABCDEFGHIJ L` [dureeVie=1, dureeConsevation = 100, stock = 3]
— `AB D FGHIJKL` Pas de DT
— `AB DEF HIJKL` Pas de DT
— `AB DEFGH JKL` Pas de DT
— `AB DEFGHIJ L` Pas de DT
— `ABCD F HIJKL` Pas de DT
— `ABCD FGH JKL` Pas de DT
— `ABCD FGHIJ L` [dureeVie=2, dureeConsevation = 100, stock = 3]
— `ABCDEF H JKL` [dureeVie=0, dureeConsevation = 10, stock = 30]
— `ABCDEF HIJ L` [dureeVie=1, dureeConsevation = 10, stock = 3]
— `ABCDEFGH J L` [dureeVie=1, dureeConsevation = 100, stock = 30]
— `AB D F HIJKL` Pas de DT
— `AB D FGH JKL` Pas de DT
— `AB D FGHIJ L` [dureeVie=5, dureeConsevation = 100, stock = 3]
— `AB DEF H JKL` Pas de DT
— `AB DEF HI JL` Pas de DT
— `AB DEFGH J L` Pas de DT
— `ABCD F H JKL` Pas de DT
— `ABCD F HI JL` [dureeVie=2, dureeConsevation = 10, stock = 3]
— `ABCD FGH J L` [dureeVie=2, dureeConsevation = 100, stock = 30]
— `ABCDEF H J L` [dureeVie=1, dureeConsevation = 10, stock = 30]
— `AB D F H JKL` Pas de DT
— `AB D F HIJ L` [dureeVie=20, dureeConsevation = 100, stock = 5]
— `AB D FGH J L` Pas de DT
— `AB DEF H J L` Pas de DT
— `ABCD F H J L` [dureeVie=2, dureeConsevation = 10, stock = 30]
— `AB D F H J L` [dureeVie=5, dureeConsevation = 10, stock = 30]

5 ➤ Quel problème y a-t-il à décorréler la structure du code ?

`Correction :` Le problème est que certains chemins sont impraticables, c'est à dire qu'il n'y a pas forcément de données de test pour un chemin donné comme le montre la question précédente.

6 ➤ Pour pouvoir exploiter un critère « tous les chemins », il faut réécrire le code. Proposez une nouvelle version du code. On demande à ce que l'instruction `retourner` soit utilisée une seule fois.

`Correction :`

productID ← readInput();
dureeVie ← getExpiration(productID)-clock();
dureeConservation ← getDureeConservation(productID);
stock ← getStock(productID);
reduction← 0;
si *dureeVie<1 jour* **alors** reduction ← 70;
sinon si *stock<10* **alors** ;
sinon si *dureeVie<10 jours ET dureeConservation>30 jours* **alors** reduction ← 50;
sinon si *dureeVie<2 jours* **alors** reduction ← 40;
sinon si *dureeVie<3 jours* **alors** reduction ← 10;
retourner *reduction*;

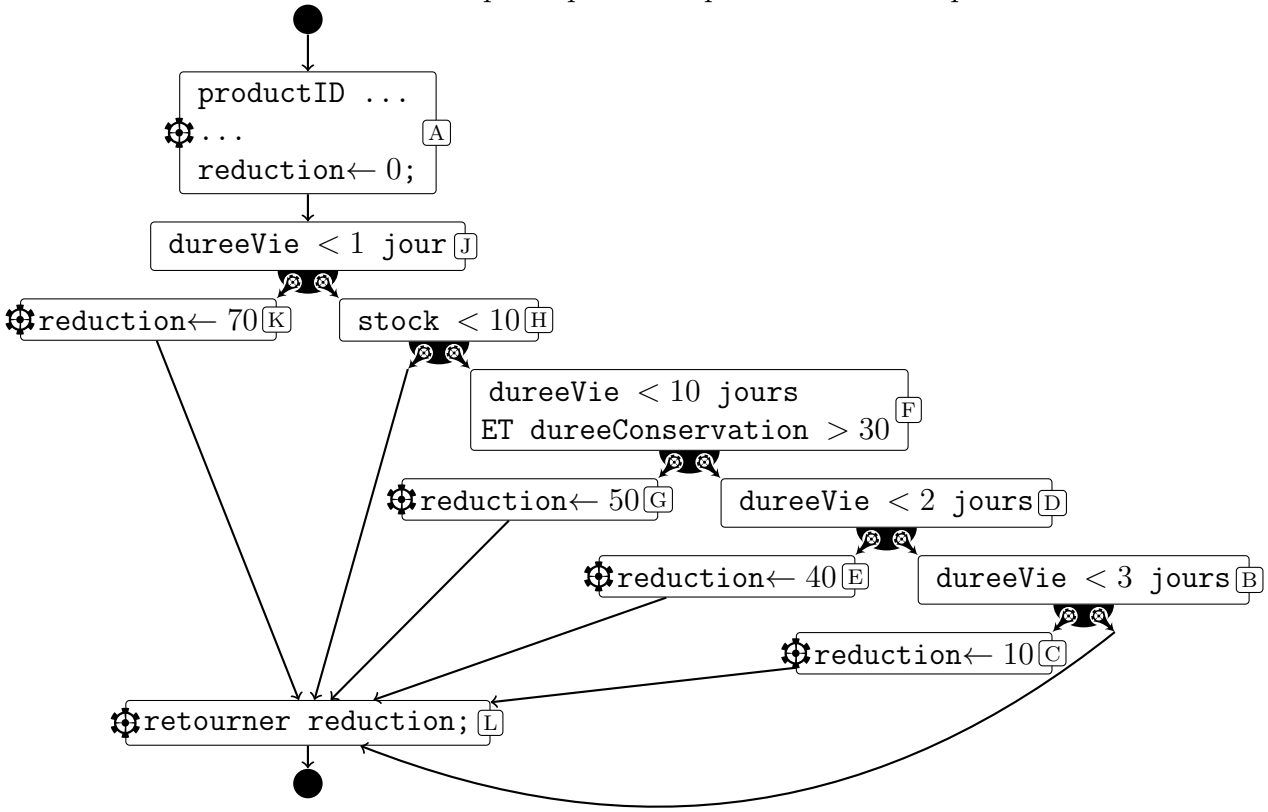
Algorithme 3 : Réduction

Raphaël Charrondière
raphael.charrondiere@inria.fr

(Note, la réponse n'est pas unique, on peut trouver d'autres variantes de cet algorithme qui répondent à l'énoncé.)

7 Donner le graphe de flot de contrôle du nouveau code, puis les données de test pour couvrir le critère « tous les chemins ».

`Correction :` Les même nœuds que la première question ont été repris.



Il y a maintenant 6 chemins, donnons en la liste avec une donnée de test pour chacun, l'ensemble des données de test couvrira alors le critère « tous les chemins ».

— `AJKL` [dureeVie=0, dureeConsevation = 10, stock = 30]
— `AJHL` [dureeVie=2, dureeConsevation = 10, stock = 3]
— `AJFGL` [dureeVie=2, dureeConsevation = 100, stock = 30]
— `AJFDEL` [dureeVie=1, dureeConsevation = 10, stock = 30]
— `AJFDBCL` [dureeVie=2, dureeConsevation = 10, stock = 30]
— `AJFDBL` [dureeVie=5, dureeConsevation = 10, stock = 30]

Raphaël Charrondière
raphael.charrondiere@inria.fr

Exercice 4 Extension des graphes de flot de contrôle

```
suivant T faire
    cas où T = 1 faire
        instructionA;
    fin
    cas où T = 2 faire
        instructionB;
    fin
    cas où T = 3 faire
        instructionC;
    fin
    autres cas faire
        Cas par défaut
    fin
fin
```

Algorithme 4 : Switch 1

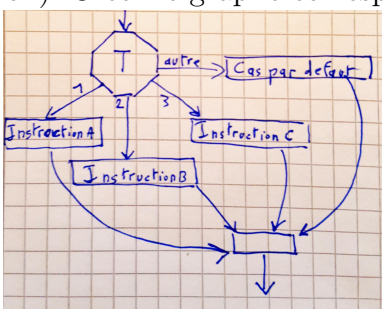
```
suivant T faire
    cas où T = 1 faire
        instructionA;
    cas où T = 2 faire
        instructionB;
    cas où T = 3 faire
        instructionC;
    fin
    cas où T = 4 faire
        instructionD;
    autres cas faire
        Cas par défaut
    fin
fin
```

Algorithme 5 : Switch 2

Vous avez certainement déjà vu l'instruction `switch` (par exemple en java) : suivant la valeur d'une variable T l'exécution du programme passe à une étiquette, ensuite l'exécution se poursuit jusqu'à la fin du `switch` sauf si l'instruction `break` est rencontrée. Dans cet exercice, nous allons ajouter un nouveau type de nœud au graphe de flot de contrôle pour gérer cette instruction.

1 ➤ Dans l'algorithme `Switch 1`, selon la valeur de T, une seule instruction est exécutée (il y a un `fin` après l'instruction). Créez le graphe correspondant, et donnez les propriétés du nouveau nœud.

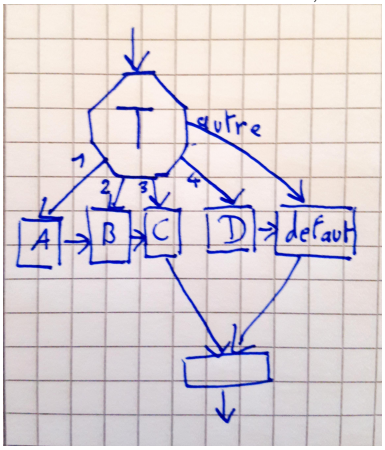
`Correction :`



Le nœud switch (ici octogonal) possède autant de sorties que de cas, vu que c'est similaire à un if else, on a une structure très similaire avec un nœud similaire au nœud de condition.

2 ➤ L'algorithme `Switch 2` est une variante où une instruction peut être atteinte par plusieurs cas. Par exemple si T vaut 1 les instructions A, B et C sont exécutées. Créez le graphe correspondant.

`Correction :`



Raphaël Charrondière
raphael.charrondiere@inria.fr

Ici c'est un peu particulier, et simplement les instructions ne sont pas redirigées vers une sortie commune mais parfois vers le prochain cas.