Les tests en boite blanches

Yassamine Seladji

yassamine.seladji@gmail.com

4 novembre 2017

Introduction

Qu'est ce qu'un logiciel ? : c'est du code

Introduction

Qu'est ce qu'un logiciel ? : c'est du code

- Des besoins.
- Une gestion de projet.

Introduction

Qu'est ce qu'un logiciel ? : c'est du code

- Des besoins.
- Une gestion de projet.
- Une spécification.

Introduction

Qu'est ce qu'un logiciel ? : c'est du code

- Des besoins.
- Une gestion de projet.
- Une spécification.
- Une conception.

Introduction

Qu'est ce qu'un logiciel ? : c'est du code

- Des besoins.
- Une gestion de projet.
- Une spécification.
- Une conception.
- Un code source.

Introduction

Qu'est ce qu'un logiciel ? : c'est du code

- Des besoins.
- Une gestion de projet.
- Une spécification.
- Une conception.
- Un code source.
- Un exécutable.

Introduction

Qu'est ce qu'un logiciel ? : c'est du code

- Des besoins.
- Une gestion de projet.
- Une spécification.
- Une conception.
- Un code source.
- Un exécutable.
- ... et la validation et la vérification

Introduction

Les testes en boite blanche :

- Des testes complémentaires aux testes en boite noire.

Introduction

Les testes en boite blanche :

- Des testes complémentaires aux testes en boite noire.
- Des testes qui utilisent les détails de la réalisation d'un logiciel, sous forme :

Introduction

Les testes en boite blanche :

- Des testes complémentaires aux testes en boite noire.
- Des testes qui utilisent les détails de la réalisation d'un logiciel, sous forme :
  - de code.

Introduction

Les testes en boite blanche :

- Des testes complémentaires aux testes en boite noire.
- Des testes qui utilisent les détails de la réalisation d'un logiciel, sous forme :
  - de code.
  - de graphe de contrôle.

Introduction

Les testes en boite blanche :

- Des testes complémentaires aux testes en boite noire.
- Des testes qui utilisent les détails de la réalisation d'un logiciel, sous forme :
  - de code.
  - de graphe de contrôle.
- Chaque jeux de tests fait fonctionner une certaine partie du logiciel : des objectifs de couvertures.

Le graphe de flot de contrôle

Le graphe de flot de contrôle est représenté par :

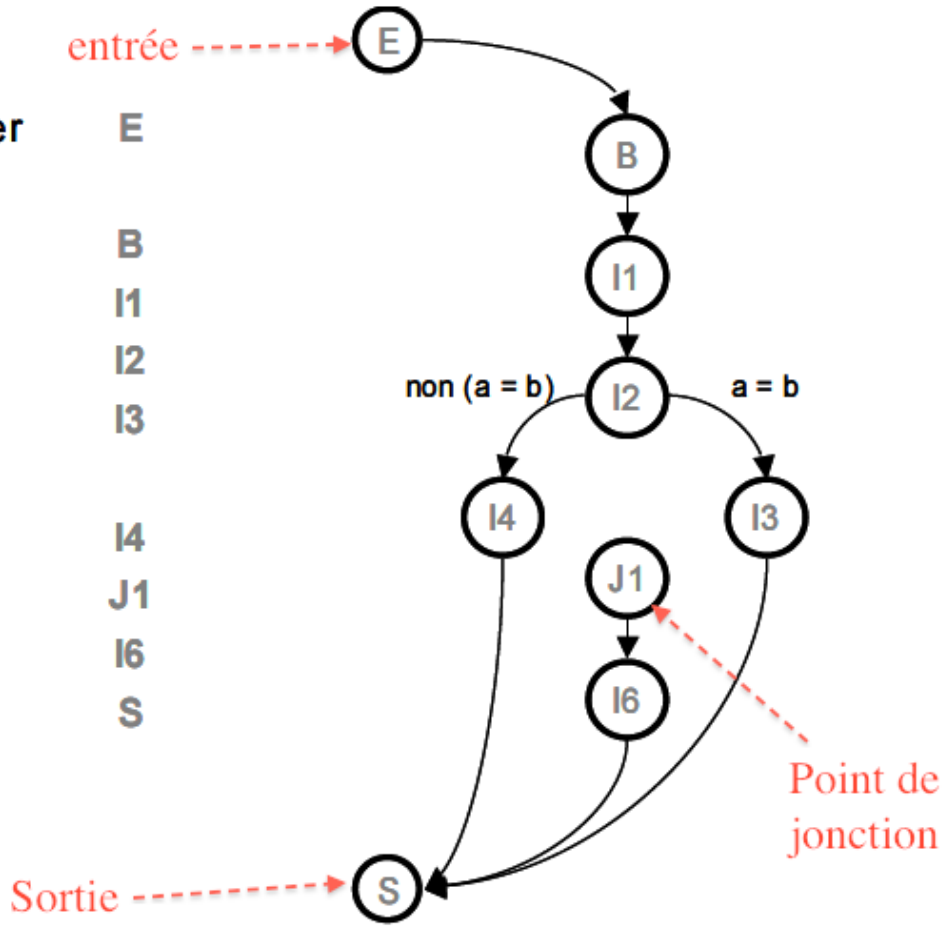
- des sommets : les instructions.
- des arcs orientés : les branchements

Le graphe de flot de contrôle

Le graphe de flot de contrôle est représenté par :

- des sommets : les instructions.
- des arcs orientés : les branchements

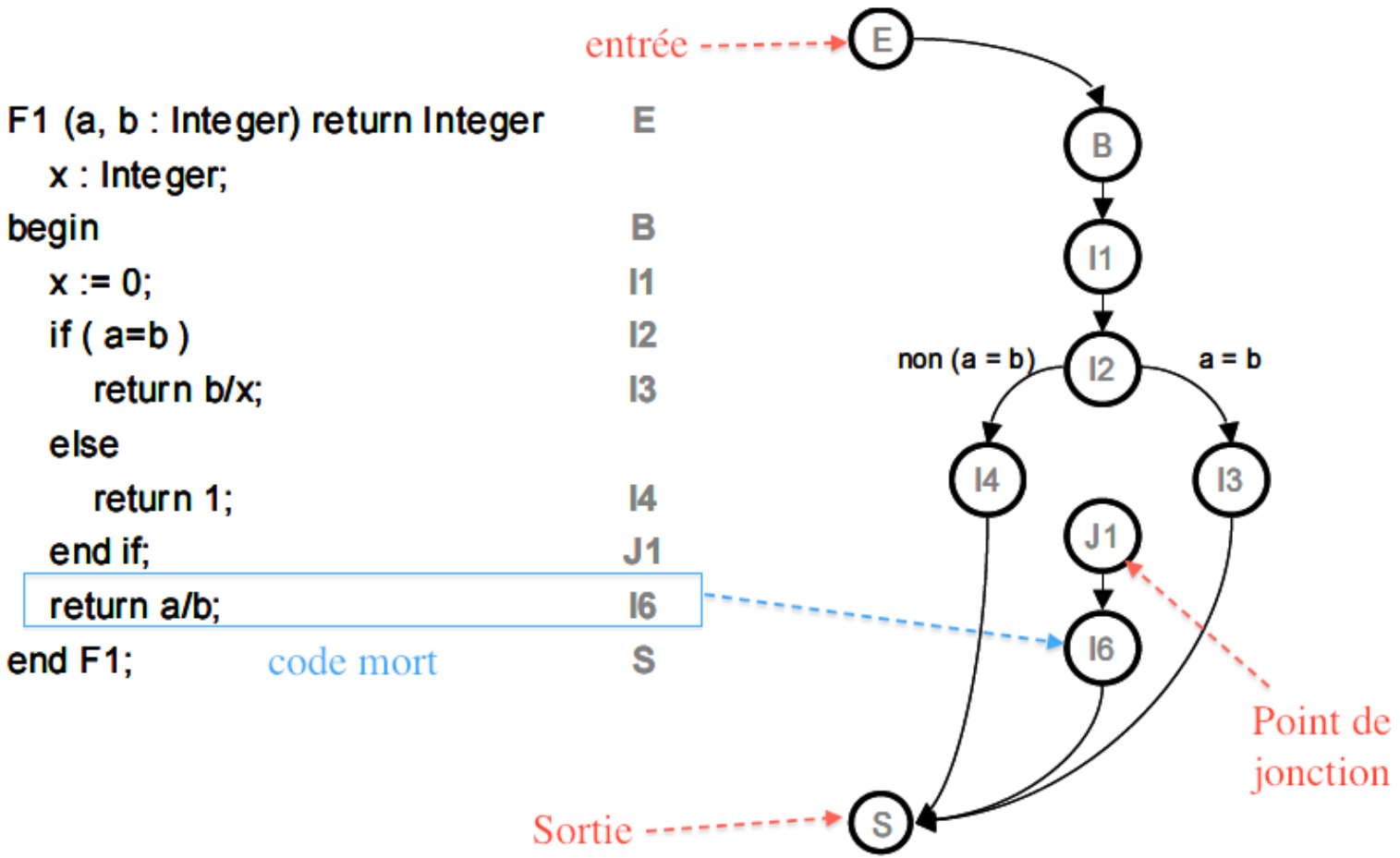
```
F1 (a, b : Integer) return Integer      E
   x : Integer;
begin                                   B
   x := 0;                              I1
   if ( a=b )                           I2
      return b/x;                       I3
   else
      return 1;                         I4
   end if;                              J1
   return a/b;                          I6
end F1;                                 S
```



Le graphe de flot de contrôle

Le graphe de flot de contrôle est représenté par :

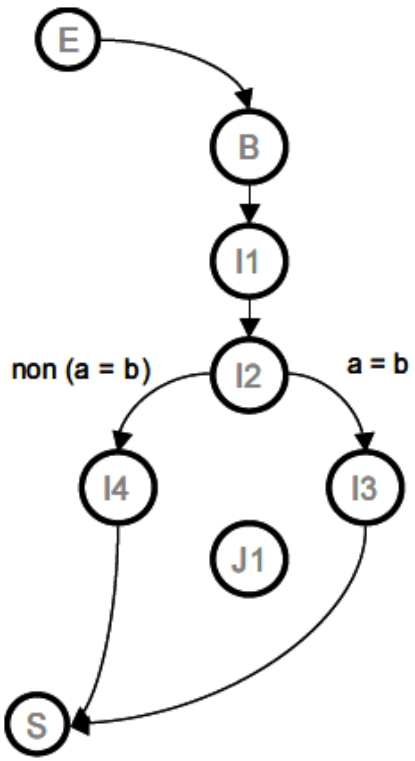
- des sommets : les instructions.
- des arcs orientés : les branchements



Des critères de couvertures : toutes les instructions

Les jeux de tests doivent parcourir la totalité des sommets correspondant à une instruction du GFC.

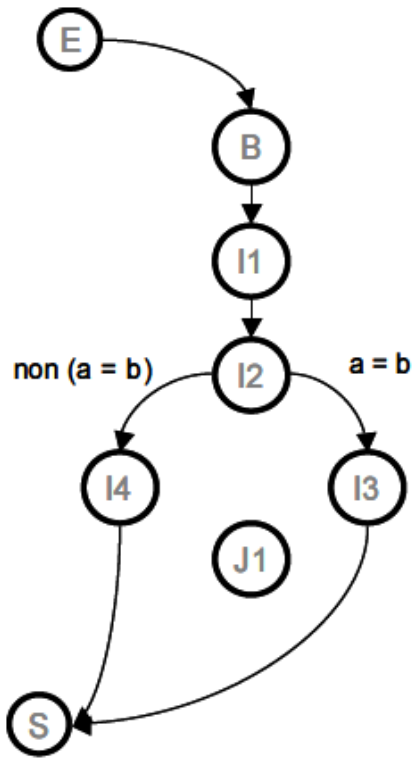
```
F1 (a, b : Integer) return Integer      E
   x : Integer;
begin                                   B
   x := 0;                              I1
   if ( a=b )                           I2
      return b/x;                       I3
   else
      return 1;                         I4
   end if;                              J1
end F1;                                 S
```



Des critères de couvertures : toutes les instructions

Les jeux de tests doivent parcourir la totalité des sommets correspondant à une instruction du GFC.

```
F1 (a, b : Integer) return Integer      E
   x : Integer;
begin                                   B
   x := 0;                              I1
   if ( a=b )                           I2
      return b/x;                       I3
   else
      return 1;                         I4
   end if;                              J1
end F1;                                 S
```



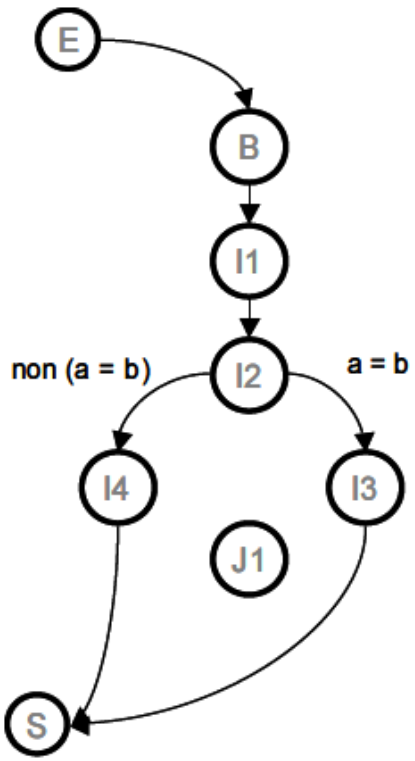
(a=1,b=1)⇒ (E,B,I1,I3,S).

(a=0,b=1)⇒ (E,B,I4,S).

Des critères de couvertures : toutes les instructions

(a=1,b=1)⇒ (E,B,I1,I3,S).

```
F1 (a, b : Integer) return Integer      E
   x : Integer;
begin                                   B
   x := 0;                              I1
   if ( a=b )                           I2
      return b/x;                       I3
   else                  ---> division
      return 1;                         I4
   end if;                  par zero     J1
end F1;                                 S
```



Des critères de couvertures : toutes les instructions

D'autres propositions de jeux de tests : (1,0), (MaxInt,0),(0,MaxInt).

- Satisfait le critère "toutes les instructions".
- Ne permet pas de trouver l'erreur de la division par zéro $\Rightarrow$ (E,B,I1,I2,J1,I4) n'est pas exécutée.

Des critères de couvertures : toutes les instructions

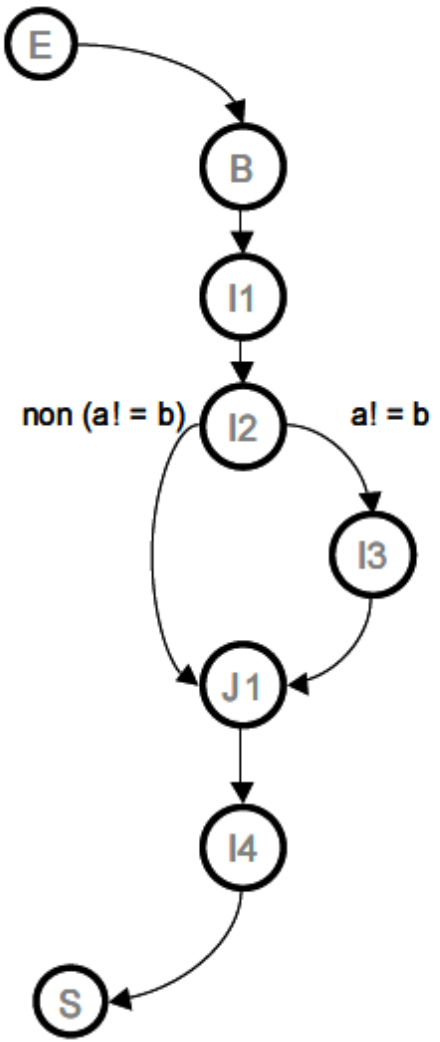
D'autres propositions de jeux de tests : (1,0), (MaxInt,0),(0,MaxInt).

- Satisfait le critère "toutes les instructions".
- Ne permet pas de trouver l'erreur de la division par zéro $\Rightarrow$ (E,B,I1,I2,J1,I4) n'est pas exécutée.

Le critère "toutes les instructions" ne prend pas en compte la notion de causalité entre les instructions.

Des critères de couvertures : toutes les instructions

```
F2 (a, b : Integer) return Integer      E
   x : Integer;
begin                                   B
   x := 0;                              I1
   if ( a!=b )                          I2
      return x := 2;                    I3
   end if;                              J1
   return (a+b) / x;                    I4
end F1;                                 S
```



Des critères de couvertures : tous les chemins

- Les séquences de test doivent couvrir l'ensemble des chemins possible du GFC.

Des critères de couvertures : tous les chemins

- Les séquences de test doivent couvrir l'ensemble des chemins possible du GFC.
- Le nombre de jeux de test peut être infini (combinaison d'instructions conditionnelles et de boucles).

Des critères de couvertures : tous les chemins

- Les séquences de test doivent couvrir l'ensemble des chemins possible du GFC.
- Le nombre de jeux de test peut être infini (combinaison d'instructions conditionnelles et de boucles).
- Prévoir d'autres critères de couvertures :

Des critères de couvertures : tous les chemins

- Les séquences de test doivent couvrir l'ensemble des chemins possible du GFC.
- Le nombre de jeux de test peut être infini (combinaison d'instructions conditionnelles et de boucles).
- Prévoir d'autres critères de couvertures :
  - Toutes les branches ou toutes les décisions.

Des critères de couvertures : tous les chemins

- Les séquences de test doivent couvrir l'ensemble des chemins possible du GFC.
- Le nombre de jeux de test peut être infini (combinaison d'instructions conditionnelles et de boucles).
- Prévoir d'autres critères de couvertures :
  - Toutes les branches ou toutes les décisions.
  - Toutes les conditions.

Des critères de couvertures : tous les chemins

- Les séquences de test doivent couvrir l'ensemble des chemins possible du GFC.
- Le nombre de jeux de test peut être infini (combinaison d'instructions conditionnelles et de boucles).
- Prévoir d'autres critères de couvertures :
  - Toutes les branches ou toutes les décisions.
  - Toutes les conditions.
  - Toutes les conditions/décisions.

Des critères de couvertures : tous les chemins

- Les séquences de test doivent couvrir l'ensemble des chemins possible du GFC.
- Le nombre de jeux de test peut être infini (combinaison d'instructions conditionnelles et de boucles).
- Prévoir d'autres critères de couvertures :
  - Toutes les branches ou toutes les décisions.
  - Toutes les conditions.
  - Toutes les conditions/décisions.
  - Toutes les conditions multiples.

Des critères de couvertures : tous les chemins

- Les séquences de test doivent couvrir l'ensemble des chemins possible du GFC.
- Le nombre de jeux de test peut être infini (combinaison d'instructions conditionnelles et de boucles).
- Prévoir d'autres critères de couvertures :
  - Toutes les branches ou toutes les décisions.
  - Toutes les conditions.
  - Toutes les conditions/décisions.
  - Toutes les conditions multiples.
  - Toutes les conditions-décisions modifiées (MC/DC)

Des critères de couvertures : tous les chemins

- Les séquences de test doivent couvrir l'ensemble des chemins possible du GFC.
- Le nombre de jeux de test peut être infini (combinaison d'instructions conditionnelles et de boucles).
- Prévoir d'autres critères de couvertures :
  - Toutes les branches ou toutes les décisions.
  - Toutes les conditions.
  - Toutes les conditions/décisions.
  - Toutes les conditions multiples.
  - Toutes les conditions-décisions modifiées (MC/DC)
  - Tous les i-chemins.

Des critères de couvertures : Toutes les branches ou toutes les décisions

- Définir des jeux de test qui passent par toutes les branches du GFC.

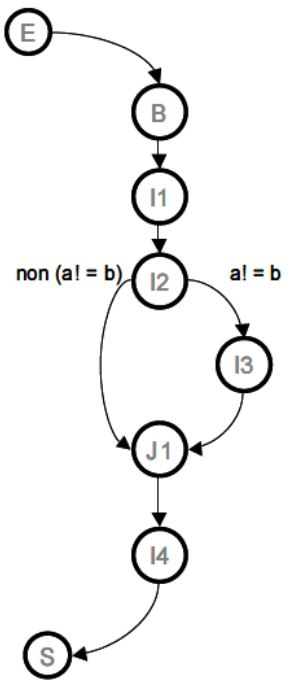
Des critères de couvertures : Toutes les branches ou toutes les décisions

- Définir des jeux de test qui passent par toutes les branches du GFC.
- Pour des instructions conditionnelles : définir un jeux de test pour la décision "Vrai" et un pour la décision "Faux".

Des critères de couvertures : Toutes les branches ou toutes les décisions

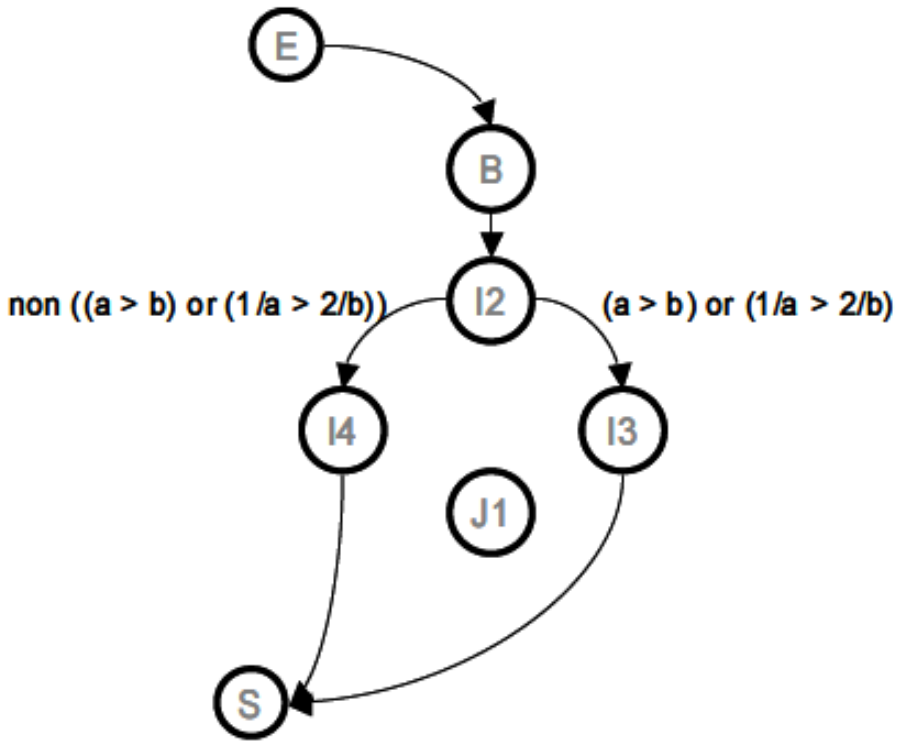
- Définir des jeux de test qui passent par toutes les branches du GFC.
- Pour des instructions conditionnelles : définir un jeux de test pour la décision "Vrai" et un pour la décision "Faux".

```
F2 (a, b : Integer) return Integer      E
   x : Integer;
begin                                   B
   x := 0;                              I1
   if ( a!=b )                          I2
      return x := 2;                    I3
   end if;                              J1
   return (a+b) / x;                    I4
end F1;                                 S
```



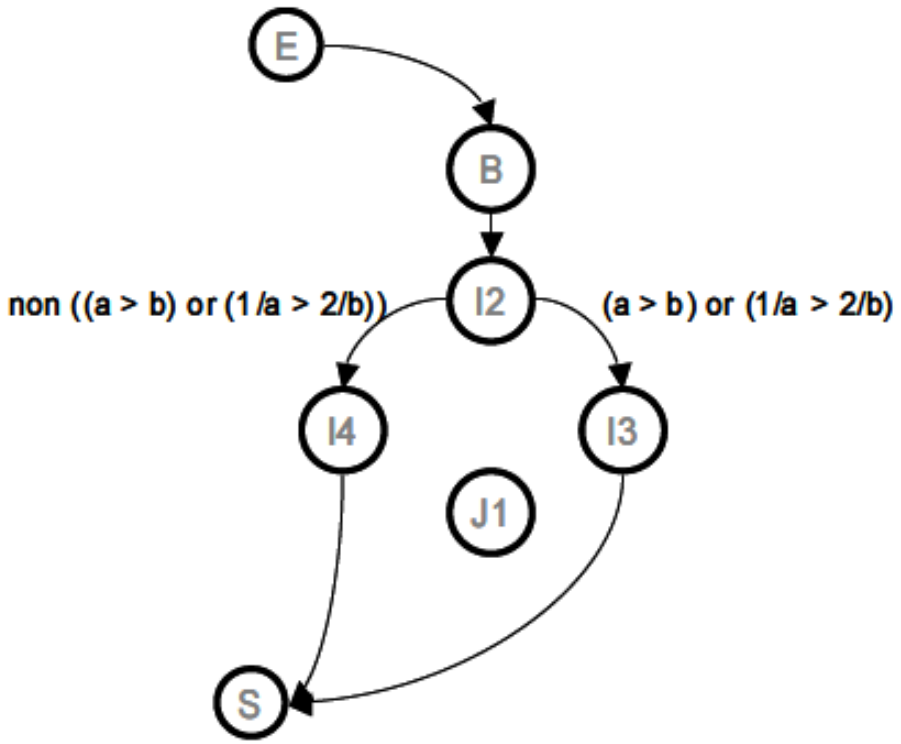
Des critères de couvertures : Toutes les branches ou toutes les décisions

```
F3 (a, b : Float) return Float          E
begin                                   B
   if ( a>b ) or (1/a > 2/b) then       I1
      return a;                         I2
   else
      return b;                         I4
   end if;                              J1
end F1;                                 S
```



Des critères de couvertures : Toutes les branches ou toutes les décisions

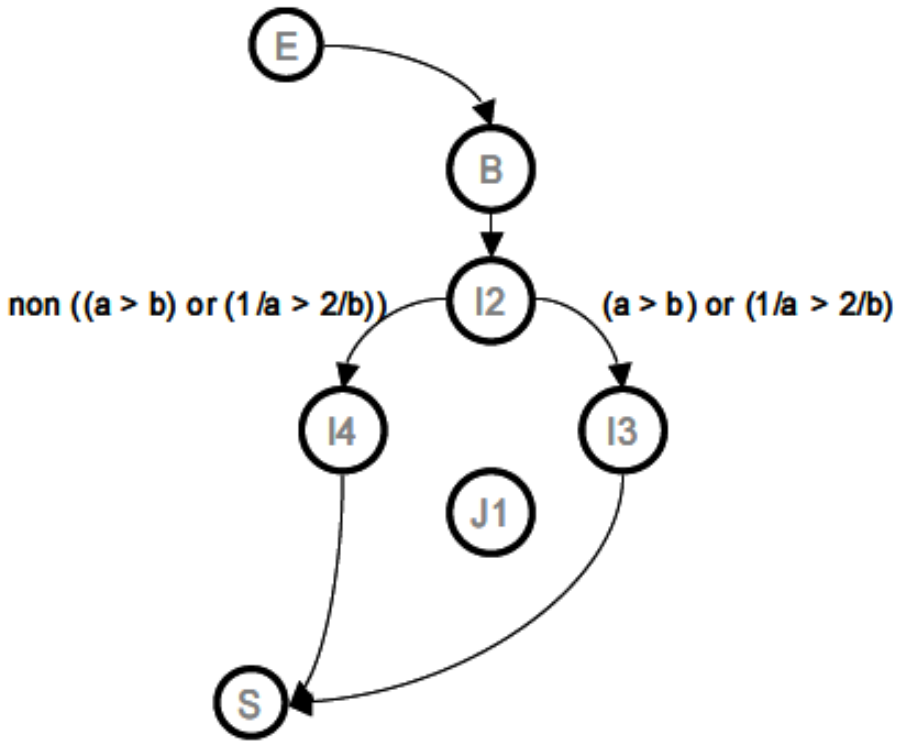
```
F3 (a, b : Float) return Float        E
begin                                 B
   if ( a>b ) or (1/a > 2/b) then     I1
      return a;                       I2
   else
      return b;                       I4
   end if;                            J1
end F1;                               S
```



(a=1,b=1) et (a=1,b=2) satisfait le critère toutes les branches.

Des critères de couvertures : Toutes les branches ou toutes les décisions

```
F3 (a, b : Float) return Float          E
begin                                   B
   if ( a>b ) or (1/a > 2/b) then       I1
      return a;                         I2
   else
      return b;                         I4
   end if;                              J1
end F1;                                 S
```



(a=1,b=1) et (a=1,b=2) satisfait le critère toutes les branches.
Pour a=0 ou b=0 l'erreur de division par zéro n'est pas détectée.

Des critères de couvertures : Toutes les branches ou toutes les décisions

Limitations :

- Ne permet pas de découvrir une erreur dans une expression conditionnelle composée.
- Un jeux de valeurs vérifiant (A=Vrai, B=Vrai) et (A=Faux, B=Vrai) :
  - Permet de couvrir toutes les branches.
  - Ne permet pas de découvrir une erreur sur les opérateurs utilisés.

Des critères de couvertures : Toutes les conditions-décisions

- Ce critère permet de couvrir toutes les branches et toutes les valeurs des conditions intervenant dans les expressions conditionnelle.
- Pour une expression contenant N conditions, il faut N+1 jeux de tests.
- L'expression (A or B), il faut des jeux de tests pour :
  - A = Vrai.
  - A = Faux.
  - B = Vrai.
  - B = Faux.
  - (A or B) = Faux.
  - (A or B) = Vrai.

Des critères de couvertures : Toutes les conditions-décisions modifiées (MC/DC)

- Améliore les différents critères de couverture basés sur les conditions.
- Utiliser pour la certification des logiciels embarqués pour l'avionique au niveau A.
- Pour une décision contenant N conditions, il faut au plus N+1 jeux de tests.

Des critères de couvertures : Toutes les conditions-décisions modifiées (MC/DC)

- Le choix du jeux de test se fait comme suite :

Des critères de couvertures : Toutes les conditions-décisions modifiées (MC/DC)

- Le choix du jeux de test se fait comme suite :
  - Pour chaque condition, choisir sa valeur que si la variation de cette condition influe sur le résultat de la décision indépendamment des autres conditions.

Des critères de couvertures : Toutes les conditions-décisions modifiées (MC/DC)

- Le choix du jeux de test se fait comme suite :
  - Pour chaque condition, choisir sa valeur que si la variation de cette condition influe sur le résultat de la décision indépendamment des autres conditions.
- Exemple : (A or B)

Des critères de couvertures : Toutes les conditions-décisions modifiées (MC/DC)

- Le choix du jeux de test se fait comme suite :
  - Pour chaque condition, choisir sa valeur que si la variation de cette condition influe sur le résultat de la décision indépendamment des autres conditions.
- Exemple : (A or B)
  - La condition A influe sur la décision : (A=Vrai,B=Faux) = Vrai , (A=Faux,B=Faux) = Faux.

Des critères de couvertures : Toutes les conditions-décisions modifiées (MC/DC)

- Le choix du jeux de test se fait comme suite :
  - Pour chaque condition, choisir sa valeur que si la variation de cette condition influe sur le résultat de la décision indépendamment des autres conditions.
- Exemple : (A or B)
  - La condition A influe sur la décision : (A=Vrai,B=Faux) = Vrai , (A=Faux,B=Faux) = Faux.
  - La condition B influe sur la décision : (A=Faux,B=Vrai) = Vrai.

Des critères de couvertures : Toutes les conditions-décisions modifiées (MC/DC)

- Le choix du jeux de test se fait comme suite :
  - Pour chaque condition, choisir sa valeur que si la variation de cette condition influe sur le résultat de la décision indépendamment des autres conditions.
- Exemple : (A or B)
  - La condition A influe sur la décision : (A=Vrai,B=Faux) = Vrai , (A=Faux,B=Faux) = Faux.
  - La condition B influe sur la décision : (A=Faux,B=Vrai) = Vrai.
  - La condition (A=Vrai,B=Vrai) = Vrai n'est pas prise en compte, car les variations de A et B sont considérées dans les autres tests.

Toutes les conditions-décisions modifiées (MC/DC)

Freiner=((vitesse≥100)or(pente≥30))and((mode=1)or(mode=2))

1. Construire une table avec en colonne les conditions et en ligne les différentes valeurs des conditions.

Toutes les conditions-décisions modifiées (MC/DC)

Freiner=((vitesse≥100)or(pente≥30))and((mode=1)or(mode=2))

1 Construire une table avec en colonne les conditions et en ligne les différentes valeurs des conditions.

	Vitesse>100	Pente>30	Mode=1	Mode=2	Freiner
1	*V*	*V*	*V*	*V*	
2	V	V	**V**	F	**V**
3	V	V	**F**	V	**V**
4	*V*	*V*	***F***	*F*	***F***
5	V	F	V	V	
6	**V**	**F**	**V**	**F**	**V**
7	**V**	**F**	**F**	**V**	**V**
8	**V**	**F**	**F**	**F**	**F**
9	*F*	*V*	*V*	*V*	
10	**F**	**V**	**V**	**F**	**V**
11	*F*	*V*	*F*	*V*	***V***
12	F	V	F	F	**F**
13	*F*	*F*	*V*	*V*	
14	**F**	**F**	**V**	**F**	**F**
15	F	F	F	V	**F**
16	F	F	F	F	**F**

Toutes les conditions-décisions modifiées (MC/DC)

Freiner=((vitesse≥100)or(pente≥30))and((mode=1)or(mode=2))

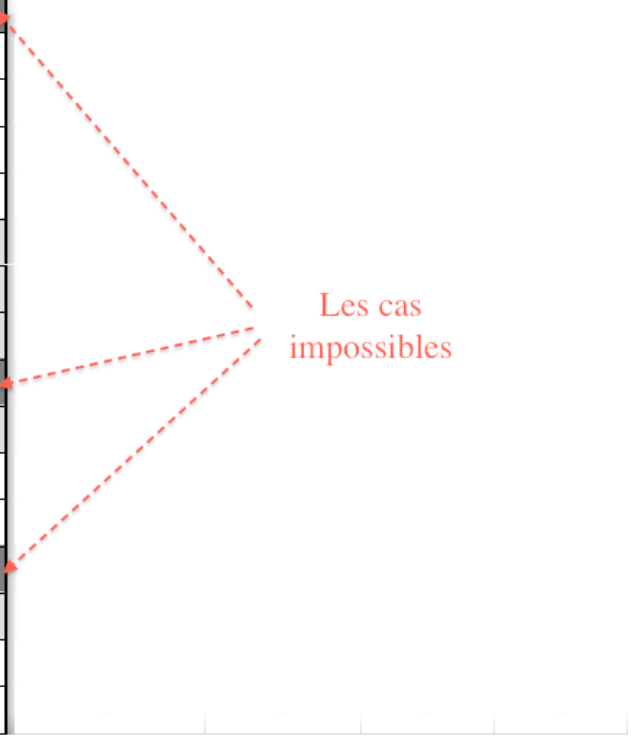
2 Supprimer les lignes qui représentent les cas impossibles.

Toutes les conditions-décisions modifiées (MC/DC)

Freiner=((vitesse≥100)or(pente≥30))and((mode=1)or(mode=2))

2 Supprimer les lignes qui représentent les cas impossibles.

	Vitesse>100	Pente>30	Mode=1	Mode=2	Freiner
1	*V*	*V*	*V*	*V*	
2	V	V	**V**	F	**V**
3	V	V	**F**	V	**V**
4	*V*	*V*	***F***	*F*	***F***
5	V	F	V	V	
6	**V**	**F**	**V**	**F**	**V**
7	**V**	**F**	**F**	**V**	**V**
8	**V**	**F**	**F**	**F**	**F**
9	*F*	*V*	*V*	*V*	
10	**F**	**V**	**V**	**F**	**V**
11	*F*	*V*	*F*	*V*	***V***
12	F	V	F	F	**F**
13	*F*	*F*	*V*	*V*	
14	**F**	**F**	**V**	**F**	**F**
15	F	F	F	V	**F**
16	F	F	F	F	**F**



Toutes les conditions-décisions modifiées (MC/DC)

Freiner=((vitesse≥100)or(pente≥30))and((mode=1)or(mode=2))

3 Déterminer pour chaque condition les lignes qui montrent l'influence de la condition sur la décision indépendamment des autres conditions.

Toutes les conditions-décisions modifiées (MC/DC)

Freiner=((vitesse≥100)or(pente≥30))and((mode=1)or(mode=2))

3 Déterminer pour chaque condition les lignes qui montrent l'influence de la condition sur la décision indépendamment des autres conditions.

	Vitesse>100	Pente>30	Mode=1	Mode=2	Freiner	Vitesse>100	Pente>30	Mode=1	Mode=2
1	*V*	*V*	*V*	*V*					
2	V	V	**V**	F	V	-	-	4	-
3	V	V	**F**	V	V	-	-	-	4
4	*V*	*V*	***F***	*F*	*F*	-	-	*2*	*3*
5	V	F	V	V					
6	**V**	**F**	**V**	**F**	V	**14**	**-**	**8**	**-**
7	**V**	**F**	**F**	**V**	V	**15**	**-**	**-**	**8**
8	**V**	**F**	**F**	**F**	F	**-**	**-**	**6**	**7**
9	*F*	*V*	*V*	*V*					
10	**F**	**V**	**V**	**F**	V	**-**	**14**	**12**	**-**
11	*F*	*V*	*F*	*V*	*V*	-	*15*	-	*12*
12	F	V	F	F	F	-	-	10	11
13	*F*	*F*	*V*	*V*					
14	**F**	**F**	**V**	**F**	F	**6**	**10**	**-**	**-**
15	F	F	F	V	F	7	11	-	-
16	F	F	F	F	F	-	-	-	-

Toutes les conditions-décisions modifiées (MC/DC)

Freiner=((vitesse≥100)or(pente≥30))and((mode=1)or(mode=2))

3 Déterminer pour chaque condition les lignes complémentaires qui montrent l'influence de la condition sur la décision indépendamment des autres conditions.

	Vitesse>100	Pente>30	Mode=1	Mode=2	Freiner	Vitesse>100	Pente>30	Mode=1	Mode=2
1	*V*	*V*	*V*	*V*					
2	V	V	**V**	F	V			4	-
3	V	V	**F**	V	V			-	4
4	*V*	*V*	***F***	*F*	*F*			*2*	*3*
5	V	F	V	V					
6	**V**	**F**	**V**	**F**	V	**14**	**-**	**8**	**-**
7	**V**	**F**	**F**	**V**	V	**15**	**-**	**-**	**8**
8	**V**	**F**	**F**	**F**	F	**-**	**-**	**6**	**7**
9	*F*	*V*	*V*	*V*					
10	**F**	**V**	**V**	**F**	V	**-**	**14**	**12**	**-**
11	*F*	*V*	*F*	*V*	*V*	-	*15*	-	*12*
12	F	V	F	F	F	-	-	10	11
13	*F*	*F*	*V*	*V*					
14	**F**	**F**	**V**	**F**	F	**6**	**10**	**-**	**-**
15	F	F	F	V	F	7	11	-	-
16	F	F	F	F	F	-	-	-	-

La condition Mode=1 est couverte par les lignes 2 et 4

Toutes les conditions-décisions modifiées (MC/DC)

4 Extraire un ensemble de lignes tel que chaque condition est couverte par deux lignes complémentaires de l'ensemble. On considère les lignes (6,7,8,10,14).

	Vitesse>100	Pente>30	Mode=1	Mode=2	Freiner	Vitesse>100	Pente>30	Mode=1	Mode=2
1	*V*	*V*	*V*	*V*					
2	V	V	**V**	F	**V**	-	-	4	-
3	V	V	**F**	V	**V**	-	-	-	4
4	*V*	*V*	***F***	*F*	***F***	-	-	*2*	*3*
5	V	F	V	V					
6	**V**	**F**	**V**	**F**	**V**	**14**	**-**	**8**	**-**
7	**V**	**F**	**F**	**V**	**V**	**15**	**-**	**-**	**8**
8	**V**	**F**	**F**	**F**	**F**	**-**	**-**	**6**	**7**
9	*F*	*V*	*V*	*V*					
10	**F**	**V**	**V**	**F**	**V**	**-**	**14**	**12**	**-**
11	*F*	*V*	*F*	*V*	***V***	-	*15*	-	*12*
12	F	V	F	F	**F**	-	-	10	11
13	*F*	*F*	*V*	*V*					
14	**F**	**F**	**V**	**F**	**F**	**6**	**10**	**-**	**-**
15	F	F	F	V	**F**	7	11	-	-
16	F	F	F	F	**F**	-	-	-	-

Critères de couverture : Tous les i-chemins

- Utiliser dans le cas de GFC avec boucles.
- Difficile de définir des jeux de tests pour tous les chemins d'une boucle.
- Définir des tests qui passent de zéro à i-fois dans la boucle.

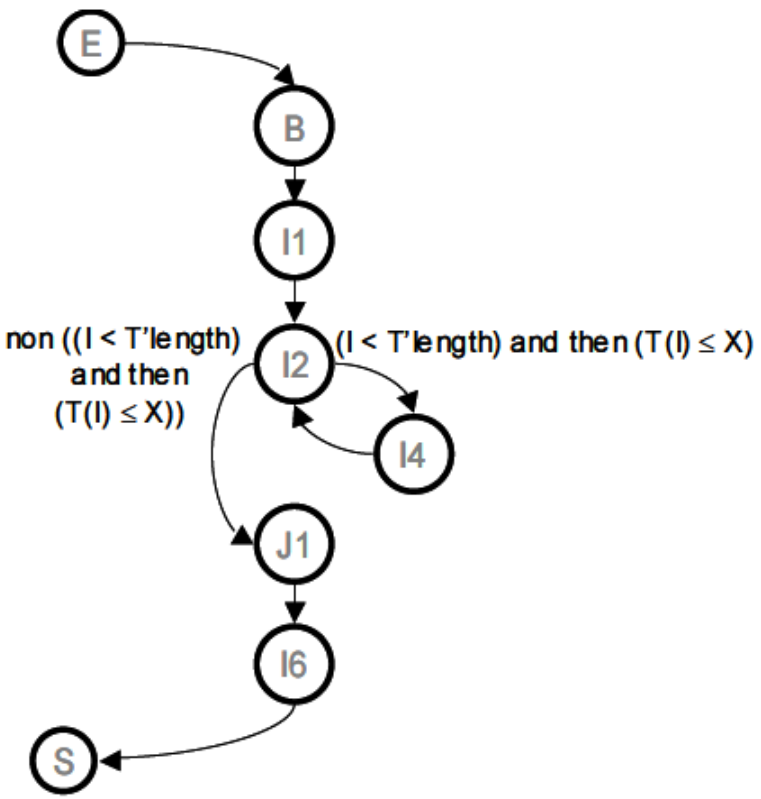
Critères de couverture : Tous les i-chemins

- Une fonction qui prend un tableau **T** d'entiers triés par ordre croissant et un entier **X**.
- Elle retourne la plus petite valeur strictement supérieure à **X**.

Critères de couverture : Tous les i-chemins

- Une fonction qui prend un tableau **T** d'entiers triés par ordre croissant et un entier **X**.
- Elle retourne la plus petite valeur strictement supérieure à **X**.

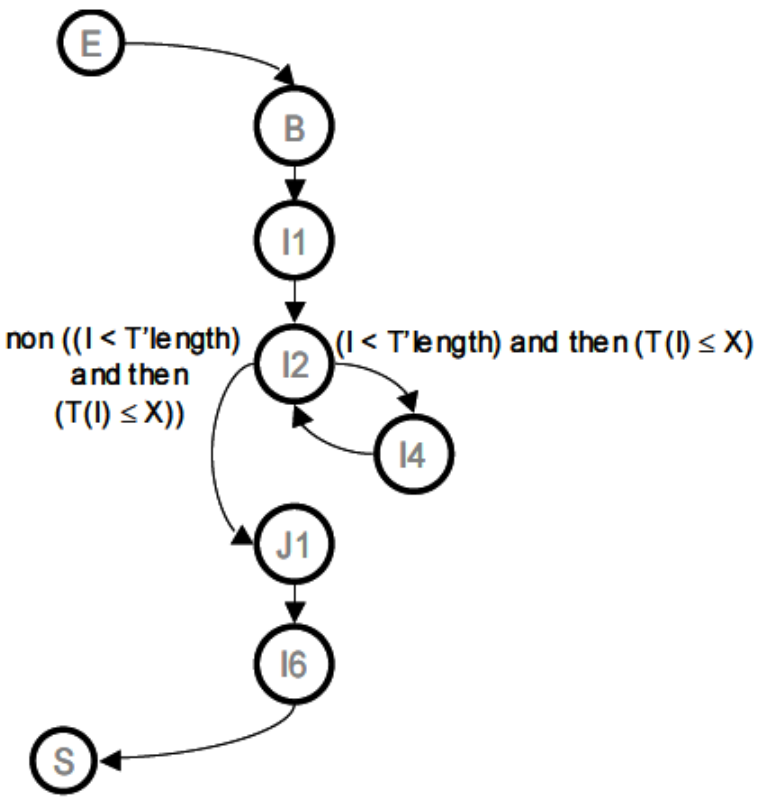
```
F2 (T: array of Integer; X : Integer) return Integer     E
   I : Integer;
begin                                                    B
   I := 1;                                               I1
   while (I<T'length) and then (T(I) ≤ X) loop           I2
      I := I+1;                                          I4
   end loop;                                             J1
   return T(I);                                          I6
end F1;                                                  S
```



Critères de couverture : Tous les i-chemins

- Le jeux de test (T=(0,2,4),X=3) permet de parcourir le chemin (B,I1,I2,I4,I2,J1,I6).
- Il satisfait le critère "toutes les branches".

```
F2 (T: array of Integer; X : Integer) return Integer     E
   I : Integer;
begin                                                    B
   I := 1;                                               I1
   while (I<T'length) and then (T(I) ≤ X) loop           I2
      I := I+1;                                          I4
   end loop;                                             J1
   return T(I);                                          I6
end F1;                                                  S
```



Critères de couverture : Tous les i-chemins

Les limitations :

- Si les valeurs de **T** sont toutes plus grandes que **X**, la fonction retourne la première valeur de **T**.
- Si les valeurs sont plus petites de **X**, la fonction retourne la dernière valeur de **T**.
- Ces erreurs ne sont pas détectées.

Critères de couverture : Tous les i-chemins

Les limitations :

- Si les valeurs de **T** sont toutes plus grandes que **X**, la fonction retourne la première valeur de **T**.
- Si les valeurs sont plus petites de **X**, la fonction retourne la dernière valeur de **T**.
- Ces erreurs ne sont pas détectées.

Solution :

- Choisir un nombre **i** pour imposer de passer par la boucle au moins **i**-fois.

Exercice

Soit la méthode Java suivante :

```
float f(float a, float b) {
        if ((a > b) || (1 / a > 2 / b)) {
            return a;
        } else {
            return b;
        }
}
```

1. Donnez un jeu de test permettant d'obtenir le critère ≪ toutes les instructions ≫.
2. Donnez un jeu de test permettant d'obtenir le critère ≪ toutes les branches ≫.
3. Dire pourquoi le critère ≪ toutes les branches≫ n'est pas ici un critère pertinent (vous pourrez montrer qu'une erreur présente n'est pas détectée alors que le critère ≪ toutes les branches ≫ est satisfait).

Conclusion

- Les tests en boite blanche utilisent le graphe de flot de contrôle afin de détecter des erreurs liées aux codes.
- Il existe plusieurs critéres de couvertures liés au graphe de flot de contrôle.
- Des critères de couvertures basés sur des flots de données peuvent aussi être utilisés.