white box testing

عملي مشترك

White Box Testing

كما ذكرنا سابقًا أن ال white box testing هو اختبار البيئة الهيكلية أي تقنية لاختبار الكود البرمجي سطر سطر ويقوم بهذا الاختبار مبرمج قادر على فهم الكود عن طريق وضع قيم واختبار النتائج على أساس تلك القيم test cases

- ملاحظة :

إذا تم العمل black box testing أو white box testing أو كلاهما معا نظرا لأن طريقتي الاختبار يكملان بعضهما ستكون النتائج عبارة عن عن مجموعة من test cases

الهدف من هذا الاختبار هو :

- **تحديد مجموعة جيدة من ال test cases**
- **القضاء على ال test cases الزائدة عن الحاجة**
- **توفير حالات اختبار (تغطية كافية)**
- **أن تكون الاختبارات أكثر فعالية**
- **تحقيق أقصى استفادة من الاختبارات محدودة المصادر**

Basis paths testing steps

هو أسلوب اختبار يساعد في تحديد المسارات المستقلة من خلال التعليمات البرمجية المصدرية أي (source code) ,الهدف هو التأكد أن كل مسار مستقل يتم تنفيذه مرة واحدة على الأقل أثناء الاختبار

الخطوات المتضمنة في اختبار المسار الأساسي :

1. فهم ال code : يجب أن يكون هناك فهم واضح للكود وتحليله
2. تحديد المسارات المستقلة
3. تعيين رقم لكل عبارة المساعدة في تحديد المسار
4. إنشاء رسم بياني (graph) وهو تمثيل رسومي يوضح محدد المسارات المستقلة عبر الكود
5. إنشاء حالات الاختبار : ويتم ذلك بمجرد تحديد المسارات المستقلة وإنشاء الرسم البياني
6. تنفيذ حالات الاختبار : التأكد من تنفيذ كل مسار مستقل مرة واحدة على الأقل.

باتباع هذه الخطوات يمكنك التأكد أن الكود قد تم اختباره جيدا وأن جميع المسارات المستقلة مغطاة أثناء الاختبار

لاحظنا أن من أهم خطوات الاختبار السابق هو الرسم البياني لذلك سنتعرف على أساسيات وطريقة هذا الرسم

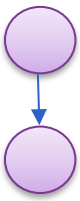
Control flow graph

وله محددات :

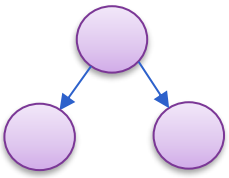
1. Lines : وهو الخطوط التي تصل بين العقد
2. Nodes : وهي الدوائر خلال الرسم
3. Region : المنطقة التي تحدها الحواف والعقد
4. Predicate node : كل عقدة يخرج منها سهم أو أكثر

Basic control flow graph structure

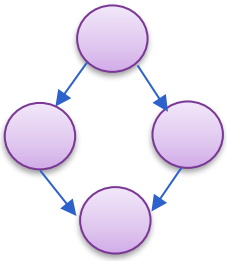
- Straight line code



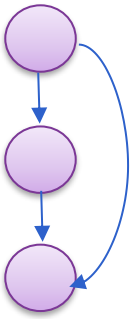
- If-then



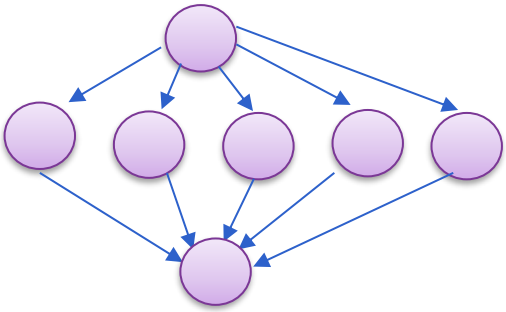
- If-then-Else

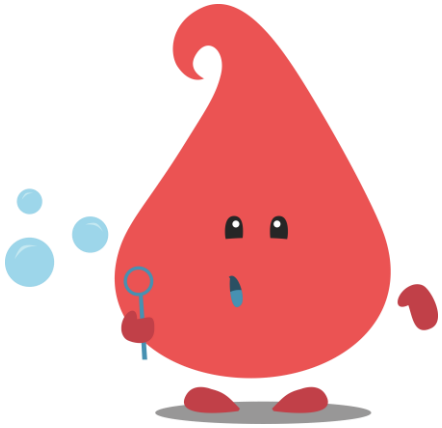


- Loop

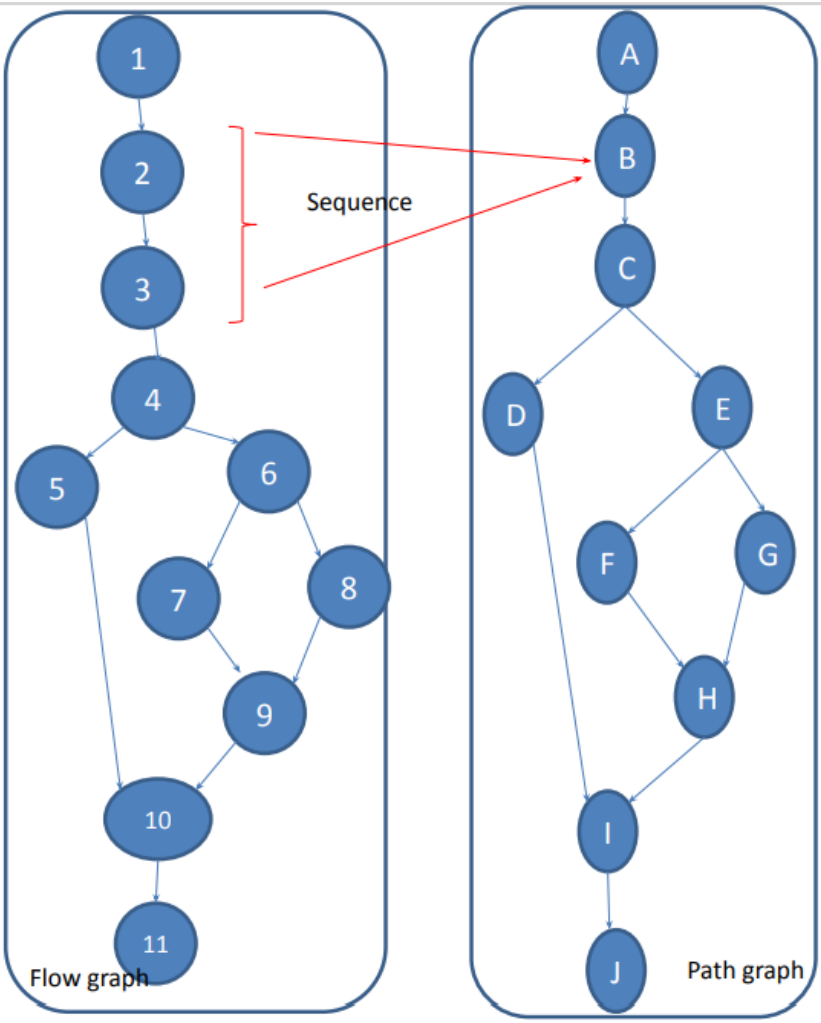


- Case statement

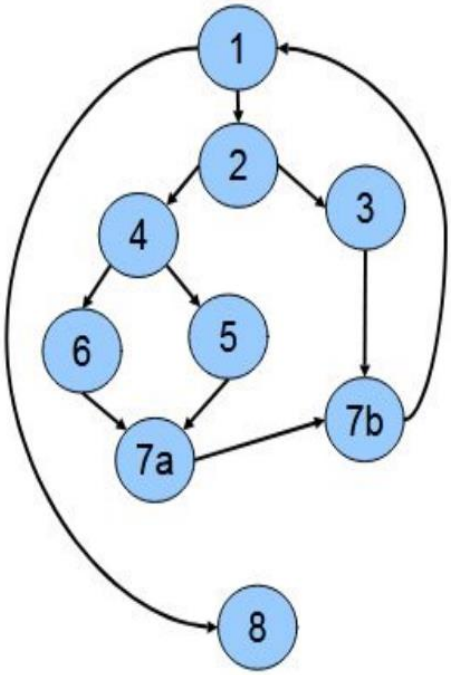




1-Program greatest
2- input A
3- input B
4- if(A>B)
5- then print(A)
6- else If(B>A)
7- then print(B)
8- else print both(=)
9- endif
10- endif
11- exit



Another Example




```
1.    do while records remain
          read record;
2.        if record field 1 = 0
3.            then process record;
                  store in buffer;
                  increment counter;
4.        elsif record field 2 = 0
5.            then reset record;
6.            else process record;
                  store in file;
7a.       endif;
      endif;
7b.   enddo;
8.  end;
```

من أهم الخطوات في اختبار مسار الأساس (Basic Path Testing) هي حساب التعقيد من خلال مخطط control flow graph (CFG) .

مخطط ال CFG هو تمثيل رسومي لتدفق التحكم أو مسار التنفيذ من خلال برنامج وهو يصور تسلسل تنفيذ العبارات في البرنامج, وهو مقياس لعدد المسارات المستقلة خطيا الموجودة في البرنامج, لذلك يمكن استخدامه لتحديد الحد الأدنى لعدد الاختبارات التي يجب تنفيذها لتغطية المسار الأساسي الكامل.

حساب مخطط التدفق (CFG) لبرنامج معين :

هناك عدة طرق :

- **بعد رسم** control flow graph

$$V(G) = edges - nodes + 2p$$

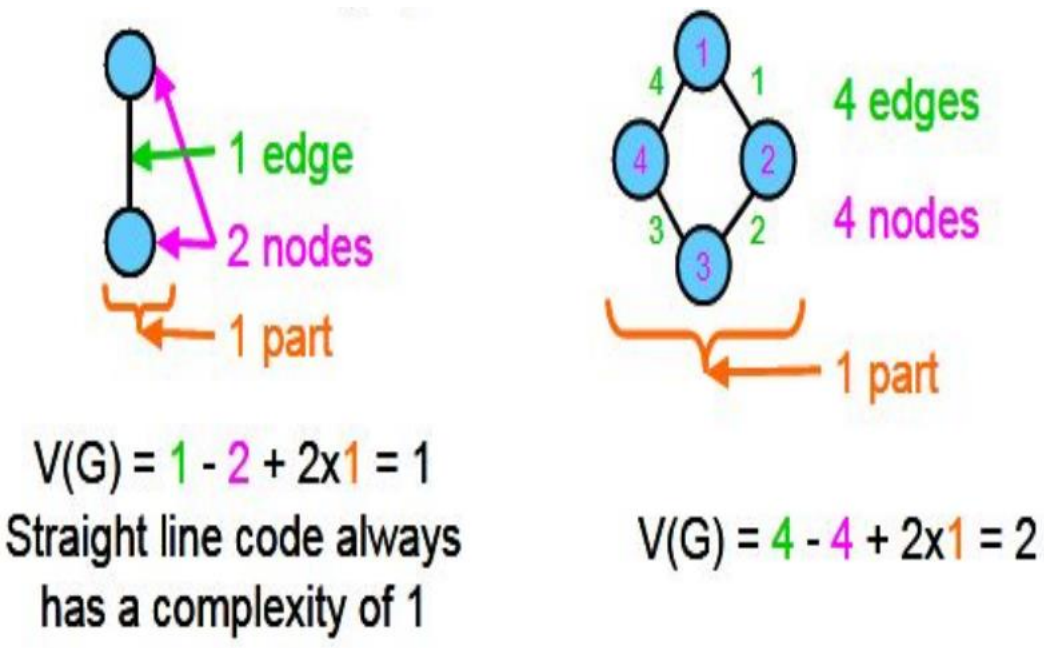
حيث :

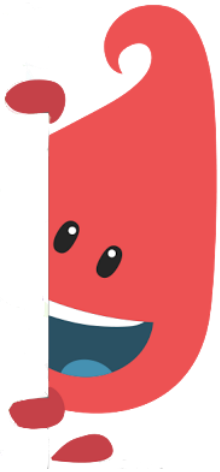
p : عدد الأجزاء غير المتصلة من الرسم البياني.

$edges$: عدد الحواف

$nodes$:عدد العقد

مثال:

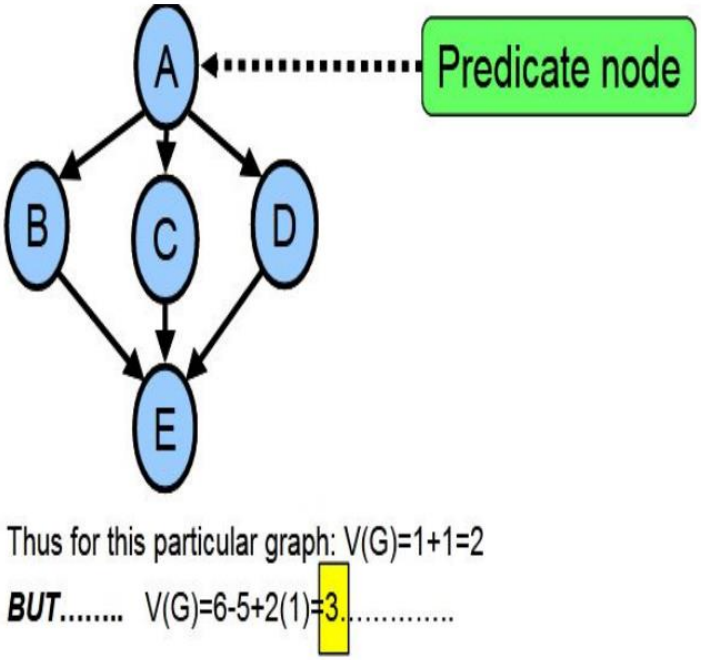


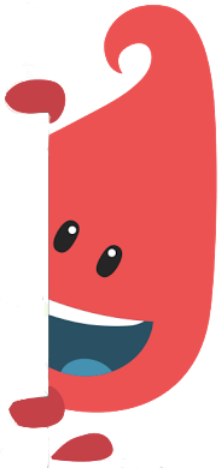


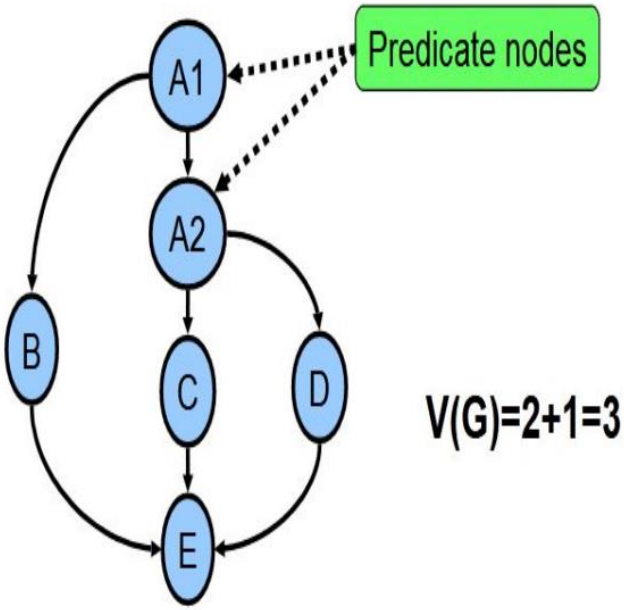
- استخدام صيغة العقد الأصلية :

$$V(G) = num\ of\ predicate\ Nodes + 1$$

مثال :



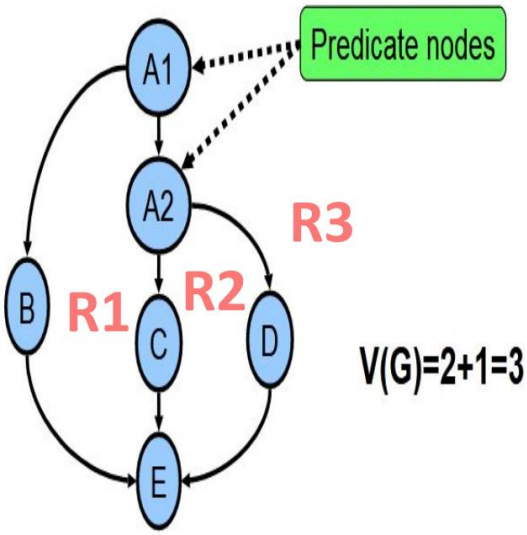




$$V(G) = num\ of\ predicate\ Nodes + 1$$

- $V(G) = num\ of\ regions\ in\ the\ control\ graph$

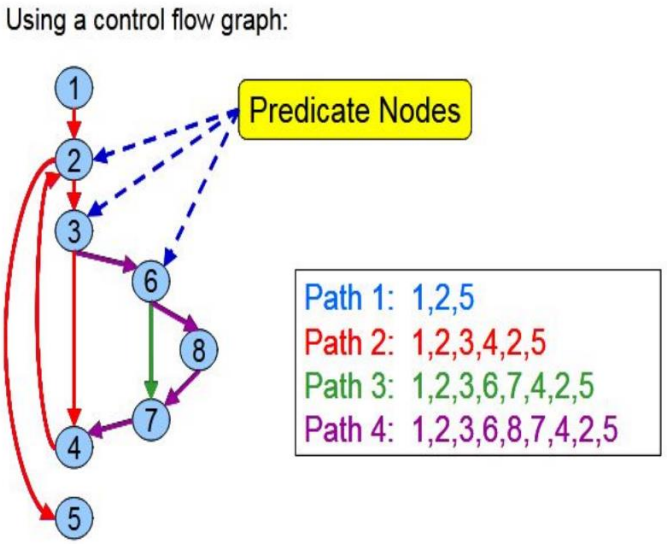
عدد المناطق المغلقة ويضاف إليها المنطقة الخارجية



أيضا نذكر أنه من الخطوات المهمة جدا في اختبار المسار الأساسي هو اختيار وتحديد المسارات الأساسية المستقلة

كل مسار أساسي يجب أن يحوي على الأقل حافة واحدة غير مختبرة وإلا يعتبر مسار زائد يجب التخلص منه

Step 3: Choose a Basis Set of Paths



كل مسار أساسي يجب ان يحوي على الأقل حافة واحدة غير مختبرة و إلا فإنه يعتبر مسار زائداً يجب التخلص منه

وأهم خطوة لدينا وهي الهدف: توليد حالات الاختبار, وتكون قيمة التعقيد الذي تم حسابه في الخطوة السابقة يعبر عن عدد المسارات المستقلة الموجودة وبالتالي تعبر عن العدد الأقل لحالات الاختبار التي سأحتاجها لتغطية المسارات .

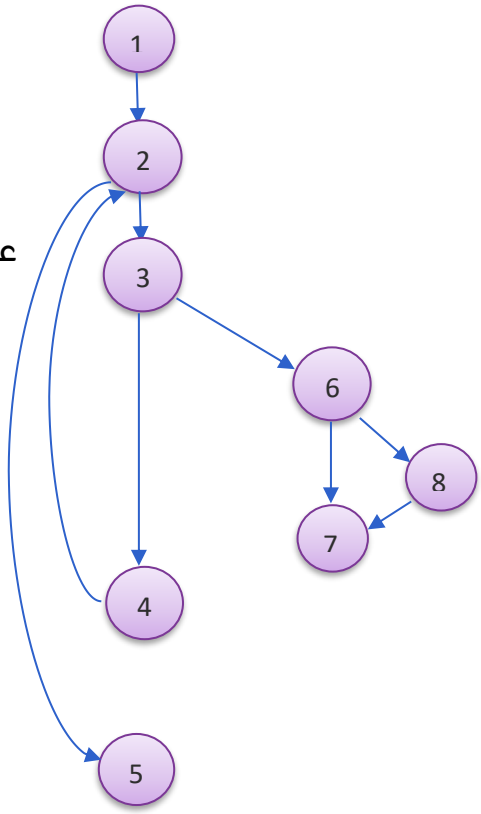
مثال كامل :

1) Draw a control flow graph
2) Calculate cyclomatic complexity

I. $V(G) = edges - nodes + 2p$
$= 10 - 8 + 2 = 4$

II. $V(G)$ = عدد العقد الأصلية + 1
$= 3 + 1 = 4$

III. $V(G)$ = عدد المناطق المغلقة ويضاف إليها المنطقة الخارجية
$V(G) = 4$



3) Choose a set of basis paths :

- Path 1 : 1,2,3
- Path 2 :1,2,3,4,2,5
- Path 3 :1,2,3,6,7,4,2,5
- Path 4 :1,2,3,6,8,7,4,2,5

- ملاحظة : يمكن أن يساعد تحديد العقد الأصلية في تحديد مجموعة المسارات الأساسية (المستقلة)

4) Generate test cases :

- Path 1 : Testcase 1

Path 1 لايمكن اختباره بمفرده ويجب اختباره كجزء من الطريق 2,3 or 4

- Path 2: Testcase 2

اضغط على إلغاء استجابة الطلب " أدخل PIN "

- Path 3 :1,2,3,6,7,4,2,5 : Testcase 3

إدخال رقم PIN صالح من المحاولة الأولى

- Path 4 :1,2,3,6,8,7,4,2,5 : Testcase 4

إدخال رمز PIN غير صالح في المحاولة الأولى ورمز PIN صالح من المحاولة الثانية .

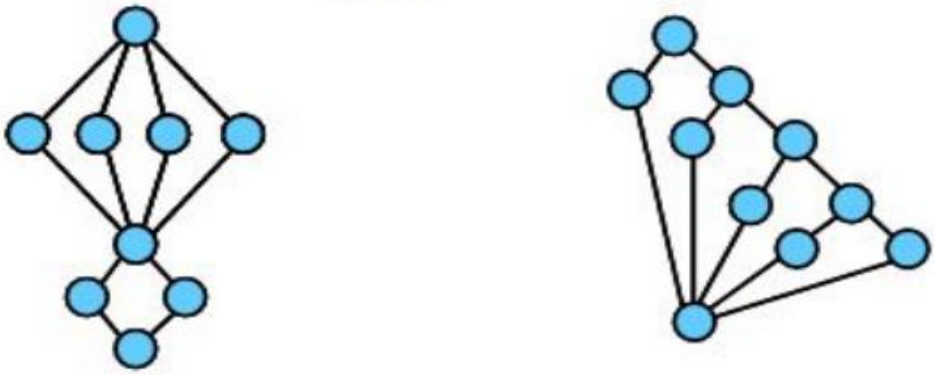
Homework

Cyclomatic Complexity - Exercise -

Control flow graph:

Model: V(G) = edges - nodes + 2p

where p = number of unconnected parts of the graph



: Calculate the Cyclomatic complexity separately for each of the two control flow graphs above.

Cyclomatic complexity of first graph = _____

Cyclomatic complexity of second graph = _____

Then assuming that these two control flow graphs are two unconnected parts of the same graph, calculate the Cyclomatic complexity of the combined graph.

Cyclomatic complexity of combined graph = _____

JUNIT Testing

اختبار شائع مفتوح المصدر للتطبيقات المستندة إلى java, يستخدم على نطاق واسع لل unit testing يركز على اختبار الوحدات الفردية أو مكونات الكود المنعزلة عن باقي النظام .

توفر ال JUnit مجموعة من التعليمات التوضيحية تسهل كتابة اختبارات الوحدة وتنفيذها.

بعض الميزات الرئيسية لها :

1. التعليقات التوضيحية لاختبار ال Junit .
2. @test : يحدد أن الطريقة هي طريقة الاختبار
3. @test(timeout =1000) : يحدد أن الطريقة ستفشل إذا استغرقت أكثر من 1000 مللي ثانية(ثانية واحدة)
4. @Before class : يحدد أن هذا ال method سيتم استدعاؤه مرة واحدة فقط قبل بدء جميع الاختبارات .
5. @Before : يحدد أنه سيتم استدعاء ال method قبل كل اختبار .
6. @After : يحدد أنه سيتم استدعاء ال method بعد كل اختبار .
7. @After class : يحدد أنه سيتم استدعاء الطريقة مرة واحدة فقط قبل بعد الانتهاء من جميع الاختبارات .

- ملاحظة : هناك أمثلة مجربة في السلايدات لمن يود الاطلاع عليها

انتهت المحاضرة