White Box Tests: Path Coverage

- When the code of a program unit is available, we can choose test cases based on the code’s control structure.

- METHOD:
  1. Determine the control structure graph of your code.
  2. Choose desired test coverage (statement, branch, path).
  3. Identify test cases as paths through the graph.
  4. Choose data to drive the code through each desired path.

Representing Program Control

Decision Points

Statements

Flow of Control

if (year<1) { throw new YearOutOfBounds(year); }
if (month==1 || month==3 || month==5 || month==7 || month==10 || month==12) {
  numDays = 32;
}
else if (month==4 || month==6 || month==9 || month==11) {
  numDays = 30;
}
else if (month==2) {
  if (isLeapYear(year)) {
    numDays = 29;
  }
  else {
    numDays = 28;
  }
}
else { throw new MonthOutOfBounds(month); }
return numDays;

Now identify all possible paths through this graph.
Example 1 (cont)

- Having identified the graph of `getNumDaysInMonth`,
- list every possible path through the graph:
  - 1, throw1
  - 1, 2, n=32, return
  - 1, 2, 3, 4, 5, n=28, return
  - 1, 2, 3, 4, 5, n=29, return
  - 1, 2, 3, n=30, return
  - 1, 2, 3, 4, throw2
- Then, choose data to drive the program through each of these paths …

Test Cases for Path Coverage

- case {year=0, month=1} tests {throw1}
- case {year=1901, month=1} tests {n=32 return}
- case {year=1901, month=2} tests {n=28 return}
- case {year=2004, month=2} tests {n=29 return}
- case {year=1901, month=4} tests {n=30 return}
- case {year=1901, month=0} tests {throw 2}

Test Coverage Measures

- **Statement Testing**
  - every statement in the unit is executed at least once in some test
- **Branch Testing**
  - for every decision point in the code, each branch is chosen at least once in some test
- **Path Testing**
  - every distinct path through the code is executed at least once in some test

Comparison of Code Coverage

- **Statement test**
  - paths: 1-2-3-4-5-6-7
  - data: X>K, RESULT=0
- **Branch test**
  - paths: 1-2-3-4-5-6-7 & 1-2-4-5-6-1
  - data: X>K, RESULT >0 & X<=K, RESULT <=0
- **Path test**
  - 1-2-3-4-5-6-7 & 1-2-3-4-5-6-1 & 1-2-4-5-6-7 & 1-2-4-5-6-1
  - X>K, R>0 & X<=K, R<=0 & X<=K, R>0 & X<=K, R<=0

A bigger picture (you don’t have to know all these methods!)

All paths  ↓  All definition-use paths

All computational/some predicate uses

All computational uses  ↓  All definition uses

All predicate/some computational uses

Branch

Statement

All predicate uses
A Program Complexity Measure

- **DEF.** The *Cyclomatic Complexity* of a graph with *V* vertices and *E* edges is \( E - V + 2 \) (or in Sommerville: *the number of conditions +1*).

- Cyclomatic complexity gives the minimum number of tests necessary to cover all edges, which is the number of independent paths through the graph.

- **E.g.:** Calculate the CC for the previous two example programs. What does a high CC tell you?

Unit Testing by Path Coverage

- **Test Purpose:**
  - to detect logical faults in implemented units by executing as many paths through the program as is practical

- **Evidence:**
  - graph of program control structure & list of paths tested
  - this process can be automated

- **Assumptions:**
  - significant faults occur in reachable paths

- **Deduction:**
  - oracle determines if output correct for a given input

- **Verdict:**
  - Outputs correct for all tested paths or incorrect paths identified

Comparison: unit test methods

- For *getNumDaysInMonth* there were similar numbers of test cases for the different methods: path needed 6 cases, equivalence class 6 and boundary another 4 cases.
- Both path and equivalence test February cases.
- Path tests can only find errors in program paths and so e.g.
  - does not identify y/100 case
  - does not catch omissions eg. 1900 not a leap year
- Path tests cover control not data structures so will miss e.g. array index out of bounds.
- Only code inspection found that August was missing.
- *…are there any other limitations of path testing?*