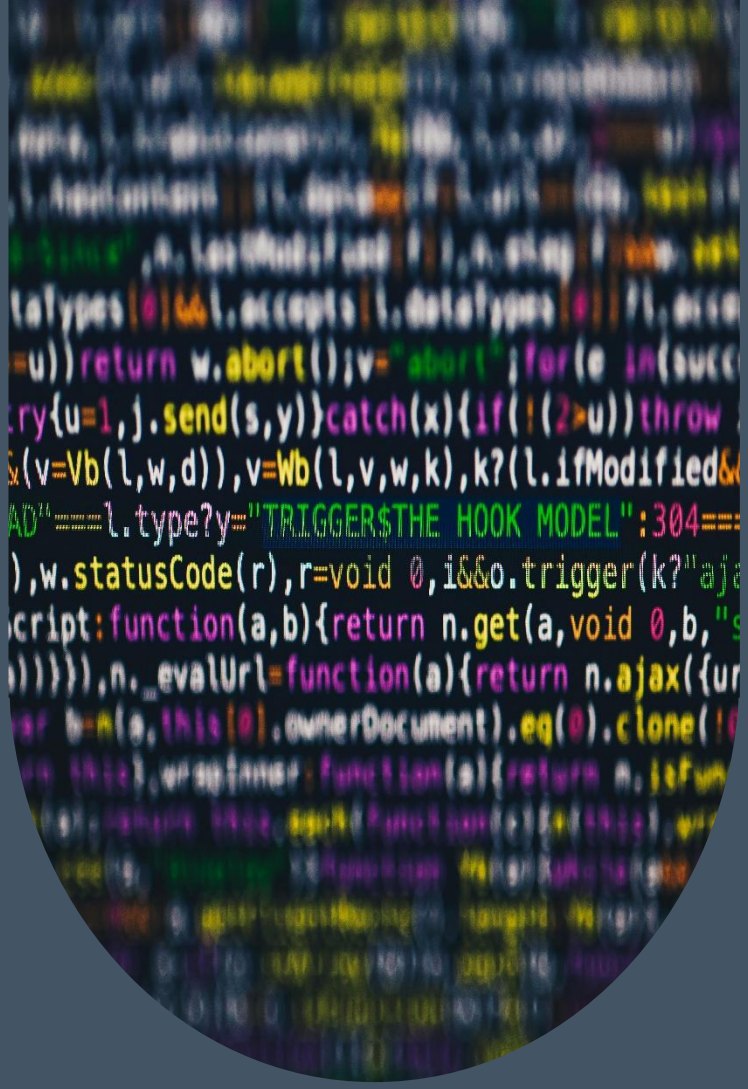
Games Engineering II

Coverage, and types of testing?

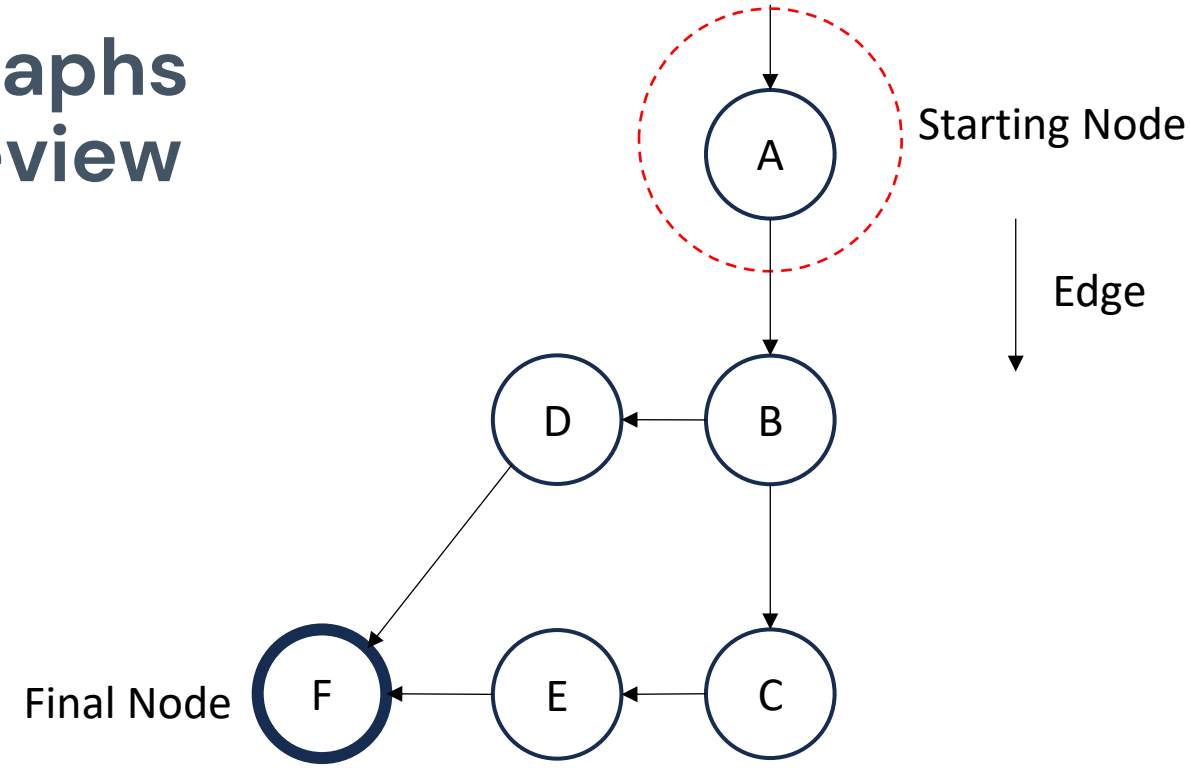
Dr. Omer Ali
omer.ali@setu.ie



SETU
Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

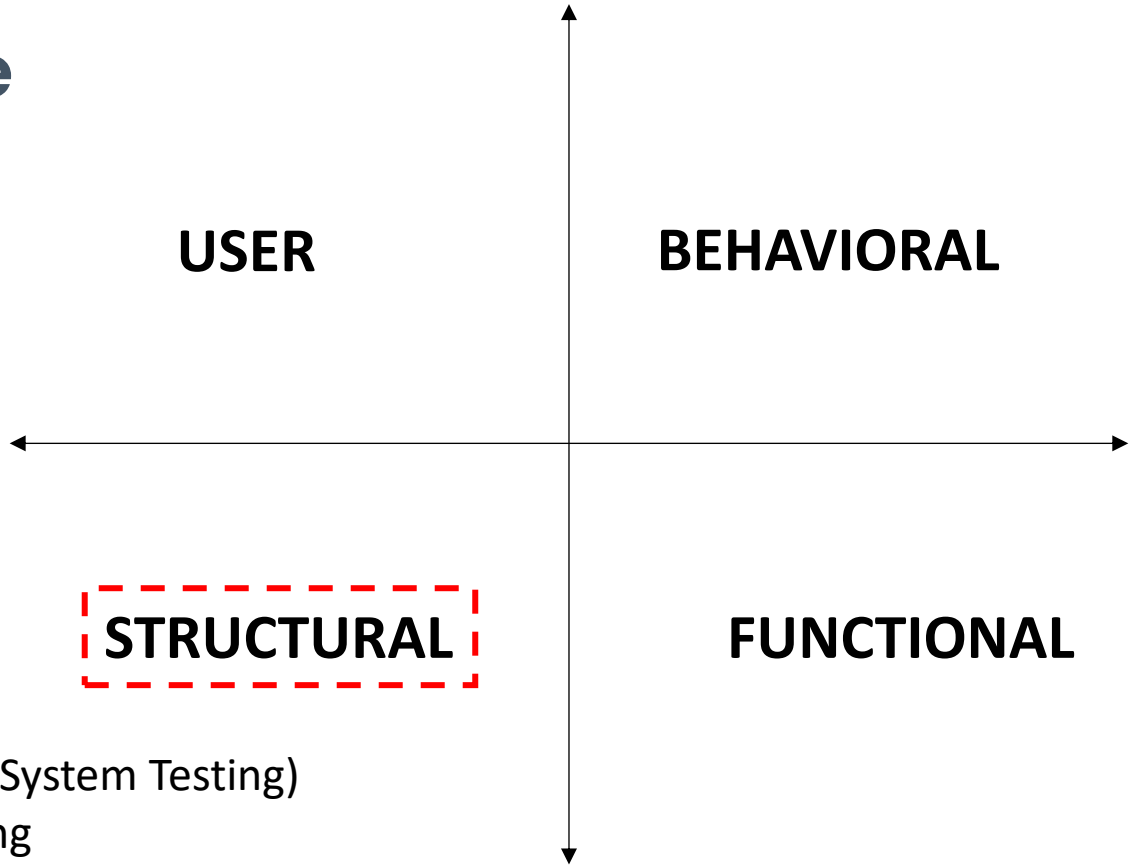
setu.ie
INSPIRING FUTURES

Graphs Review



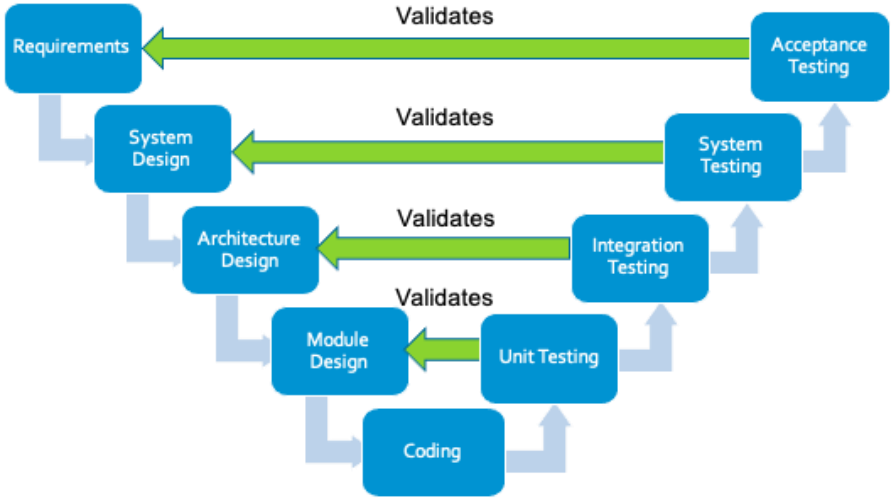
- A, B, C, D, E, F are all nodes
- Paths must connect Start and Final node (A->B->C->E->F , A->B->D->F
- A route can have one or more paths, where each path $P \in G$

Software Testing Perspective



USER → Blackbox Testing (System Testing)
BEHAVIORAL → Unit Testing
STRUCTURAL → Coding
FUNCTIONAL → Package/Integration

Software Testing Perspective Continued…



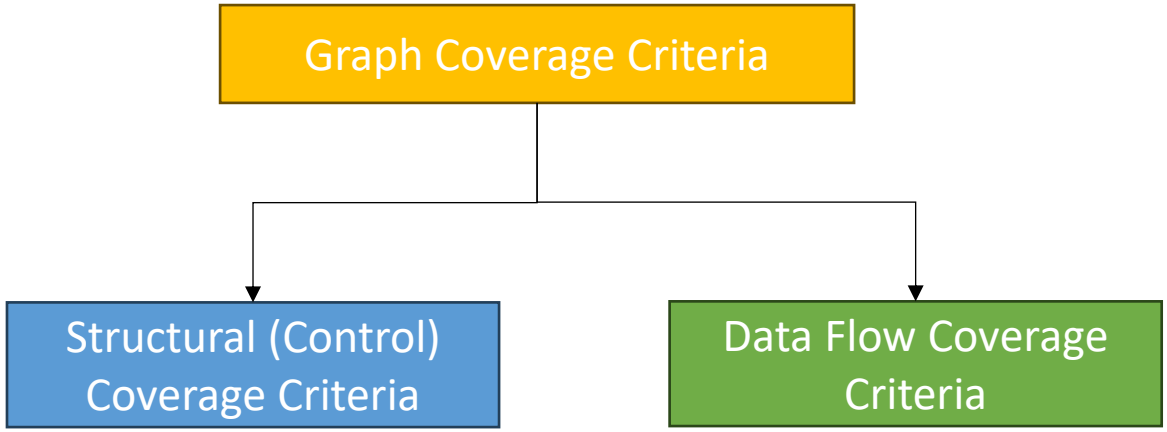
Features	Manual Testing	Automated Testing
Test Coverage	Limited test coverage due to human constraints	Potential for high test coverage by executing numerous test cases simultaneously
Consistency	Prone to human errors and inconsistencies in test execution	Consistent test execution, ensuring repeatable results
Maintenance	Test cases and documentation need to be manually updated	Test scripts need to be updated, but can be automated to some extent
Suitability for Regression Testing	Inefficient for extensive regression testing	Ideal for regression testing, allowing efficient re-execution of tests
Audit reporting	Manual logging and reporting can be time-consuming	Automated reporting ensures better traceability

How to define Test Coverage?

- Two broader categories to design test scripts.
- Graph based
- Pattern based

How to define Test Coverage?

- Two broader categories to design test scripts.
- **Graph based**
- Pattern based



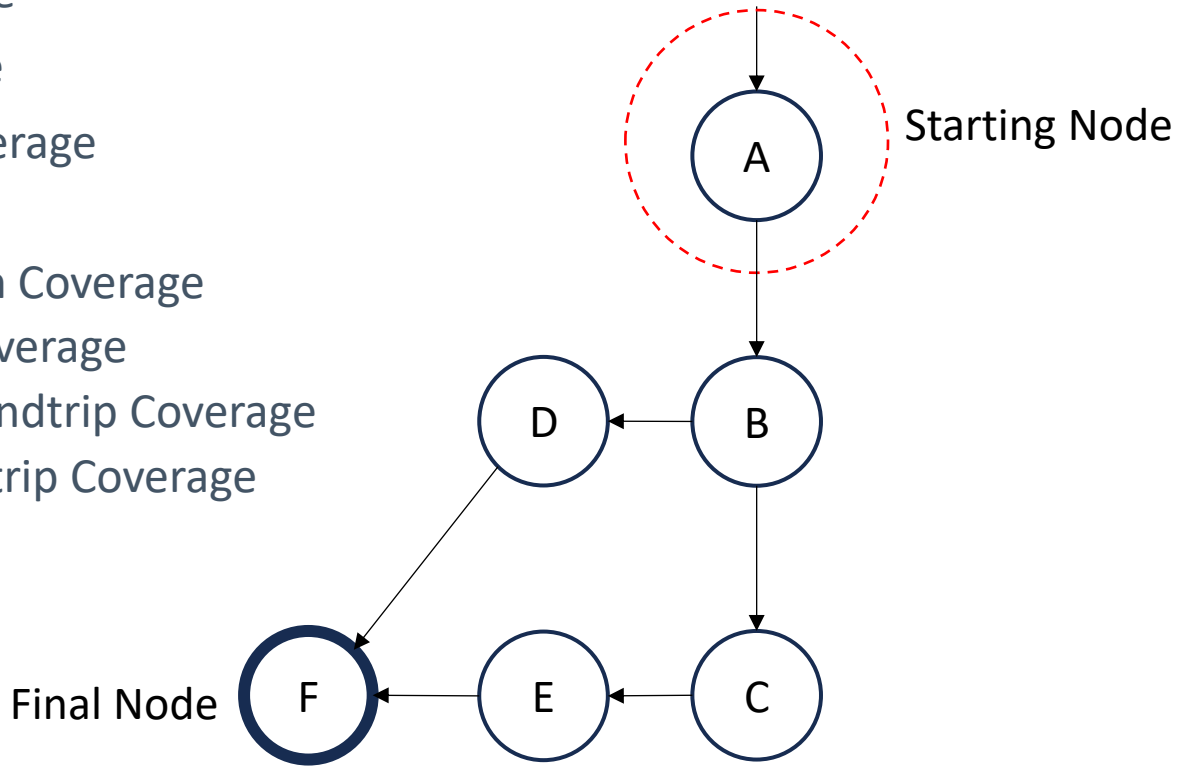
Ollscoil Teicneolaíochta an Oirdheiscirt

South East Technological University

Structural Coverage Criteria

Structural Coverage Criteria?

- Node Coverage
- Edge Coverage
- Edge-Pair Coverage
- Path Coverage
- Complete Path Coverage
- Prime Path Coverage
- Complete Roundtrip Coverage
- Simple Roundtrip Coverage

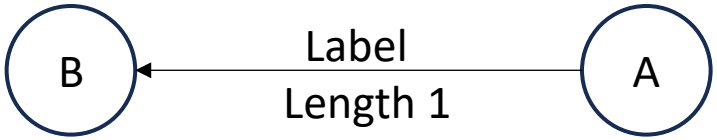


Node Coverage

- Node coverage requires that the test cases ***visit each node*** in the graph at least **once**
- Test set ***T*** satisfies node coverage on graph ***G*** *iff* for every syntactically reachable node ***n*** $\in$ ***G***, where there is a path ***P*** in (***T***) such that ***P*** visits ***n***.
- Test Requirement (TR) contains each reachable node ***n*** in graph ***G***
- ***(A path can satisfy node coverage test requirements, if it traverses through all nodes at least once)***

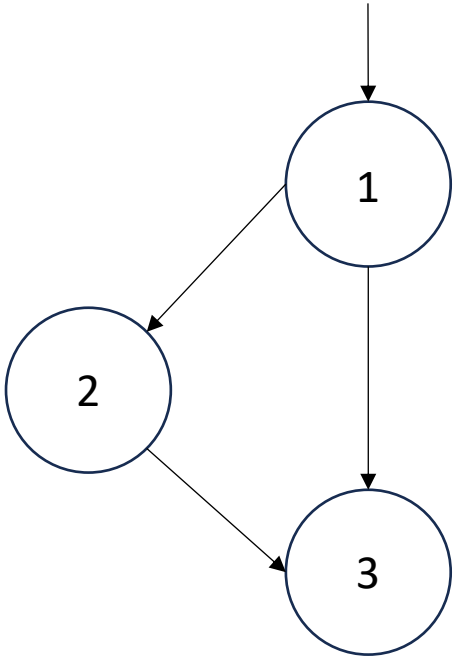
Edge Coverage

- ***TR*** contains each reachable path of length ***1***, inclusive in graph ***G***
- Edge Coverage is more robust in covering the entire graph.
- With length up to 1, it allows coverage for graphs with ***one node*** and ***no edges***
- ***(Allowing length up to 1 allows coverage to subsume node coverage)***



Node & Edge Coverage Example

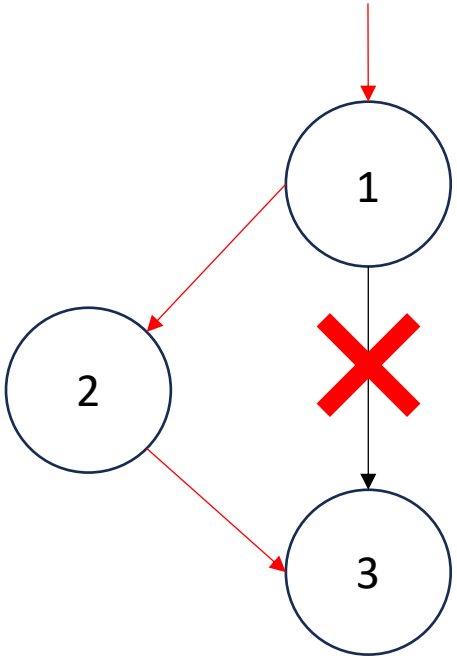
Node Coverage
TR = {1,2,3}
Test Path =?



Node & Edge Coverage Example

Node Coverage
TR = {1,2,3}
Test Path = [1,2,3]

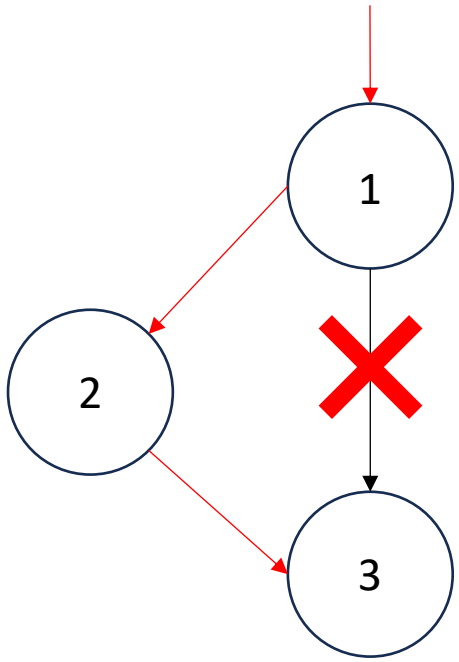
Any shortcomings?



Node & Edge Coverage Example

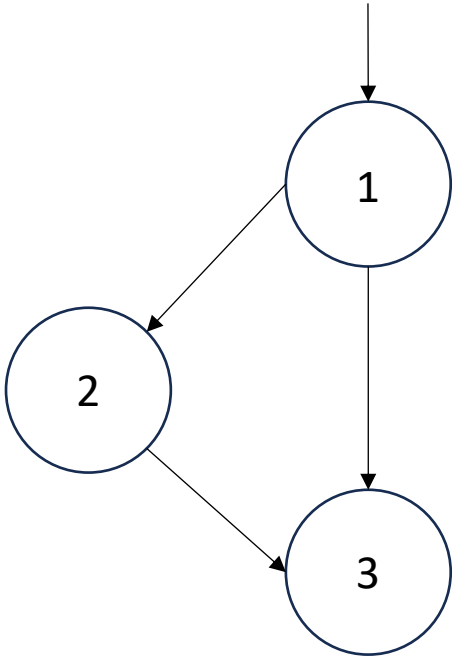
Node Coverage
TR = {1,2,3}
Test Path = [1,2,3]

Any shortcomings?
Although TR is completed, but not all paths were covered.



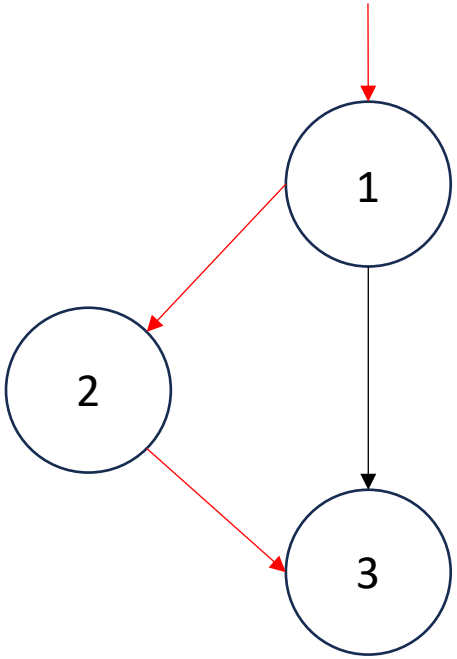
Node & Edge Coverage Example

Edge Coverage
TR = {(1,2),(2,3),(1,3)}
Test Path =?



Node & Edge Coverage Example

Edge Coverage
TR = {(1,2),(2,3),(1,3)}
Test Path = [1,2,3]

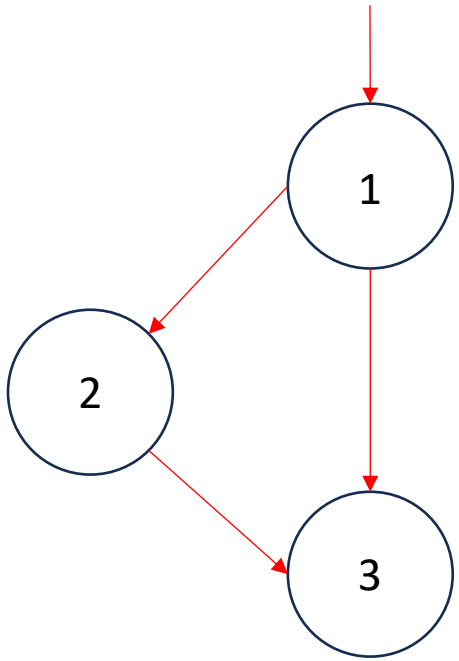


Node & Edge Coverage Example

Edge Coverage
TR = {(1,2),(2,3),(1,3)}
Test Path = [1,2,3] , [1,3]

Node and Edge coverage are different only when there is an edge in another sub-path between a pair of nodes

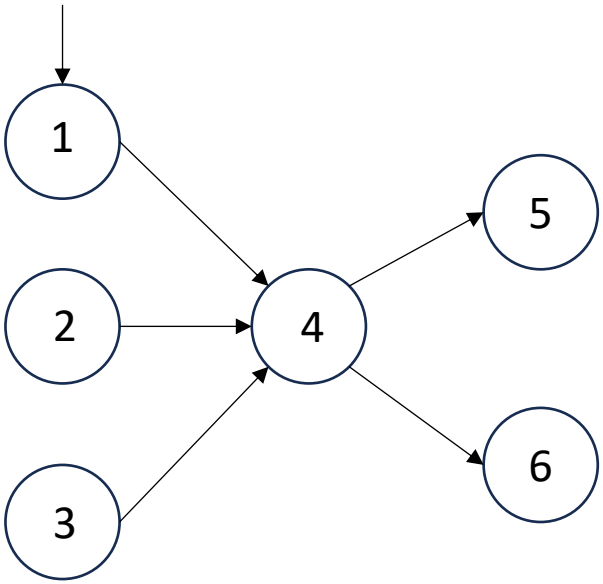
If edged are covered, that means nodes are automatically covered. In other words, it could be said that ***Edges*** subsumed ***nodes***.



Edge-Pair Coverage Example

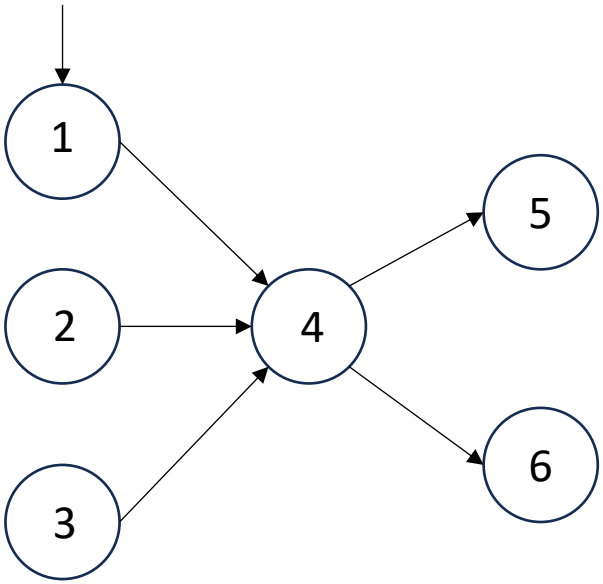
- Edge Pair Coverage: ***TR*** contains each reachable path of length up to ***2***, inclusive in ***G***
- Paths of lengths up to 2 corresponds to pairs of edges
- Again, the phrase “length up to 2” ensures edge-pair coverage holds for graphs with less than 2 edges

Edge-Pair Coverage Example



- TR = { [1,4,5], [2,4,5],[3,4,5],[1,4,6],[2,4,6],[3,4,6] }
- Test Path=?

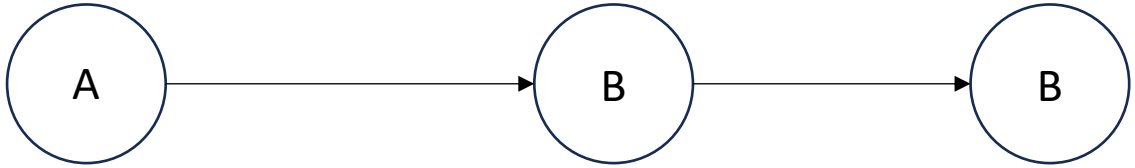
Edge-Pair Coverage Example



- TR = { [1,4,5], [2,4,5],[3,4,5],[1,4,6],[2,4,6],[3,4,6] }
- Test Path=Same as above
- If Edge-Pair coverage is achieved, then Edge coverage and Node coverage is automatically achieved. In other words, ***Edge-Pair Coverage*** subsumes ***Edge Coverage*** subsumes ***Node Coverage***.

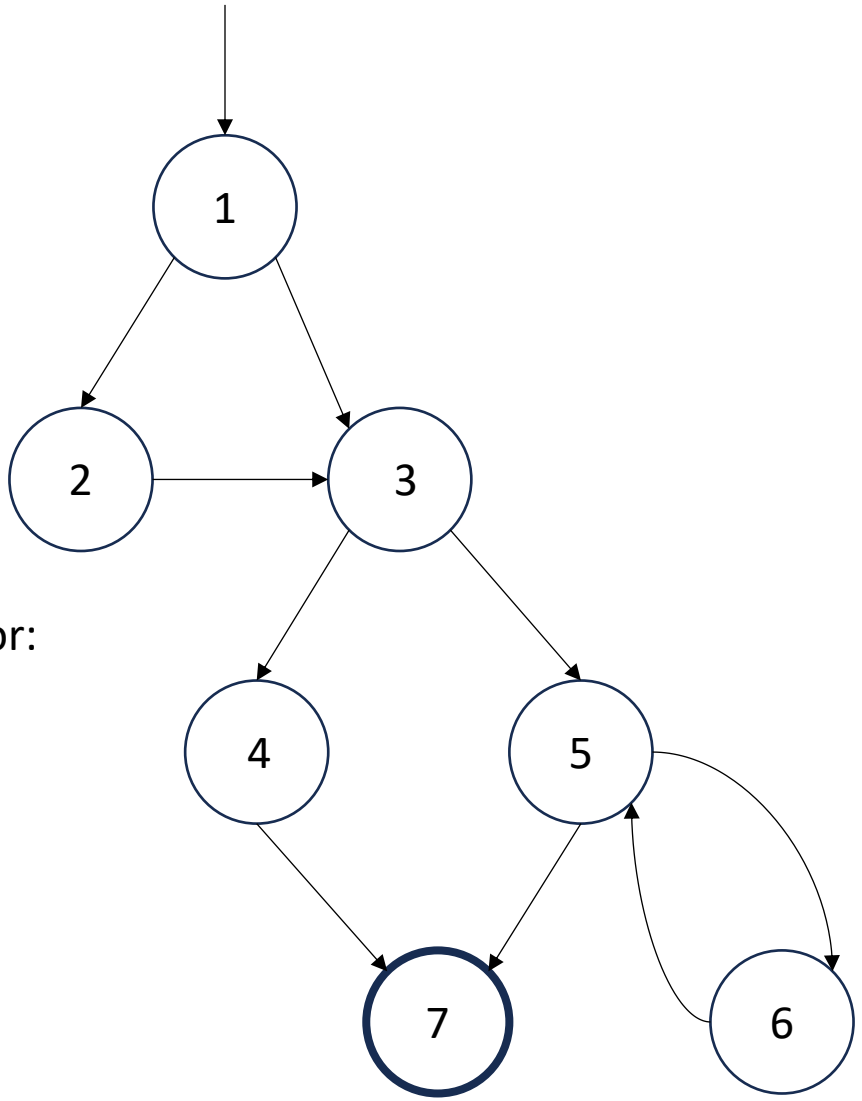
Complete Path Coverage

- An extension of edge-pair coverage where all paths are considered
- For complete path coverage, ***TR*** must contain all paths in graph ***G***.
- But what if we encounter loops?



We will end up covering and additional path (A → B → B), but essentially, we will still be at Node B

Overall Coverage Criteria



Write Test Requirements and Test Path for:

- Node Coverage
- Edge Coverage
- Edge-Pair Coverage
- Complete Path Coverage

Structural testing in practice?

When to Test?

- At Unit, integration, (sub)-system, acceptance stages

How and How much to Test?

- User specified tests that justifies the costs and offers maximum coverage
- TR must be clearly devised to know when to STOP

- We consider functional aspects of our System Under Test ***SUT*** while ***unit*** and ***integration testing***
- However, we need to define
  - Test case selection
  - Test adequacy

Test Selection & Adequacy Criteria

- Test Selection Criteria
- Conditions that must be fulfilled by a test
- For example, a criterion for a ***numerical program*** whose input domain is the ***integers***, might specify that each test must contain:
  - One positive integer
  - One negative integer
  - And a Zero
- {-9,03}, {-10,0,120},{-1,0,1} are three set of tests selected by this criterion.

Test Selection & Adequacy Criteria

- Test Adequacy Criteria
- The properties of a program that must be exercised to constitute a thorough testing
- For example, we can claim that our software has been ***adequately tested*** if:
  - The test suite causes each method to be executed at least once
  - The test suite tested logical flow between each method at least once
- This certainly provides quantifiable objectives for completeness

Test Selection & Adequacy Criteria – Example

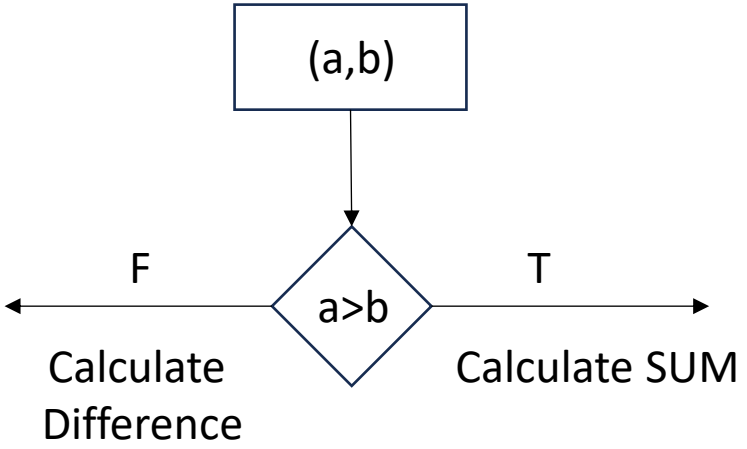
- Example 1: We need to write a program that takes two integer inputs ***a***, and ***b***, and computes :
  - Sum , if **a>b**
  - Difference, if **a<b**

- Requirement Specifications:
  - Accepts integer inputs
  - Return sum of two integers if first is greater than second
  - Return difference of two integers if first is smaller than second

- Consideration Criteria:
  - “A Test ***T*** for Program (***P***, ***R***), where ***R*** are the set of requirements, is considered adequate if each path in ***P*** is traversed at least once”

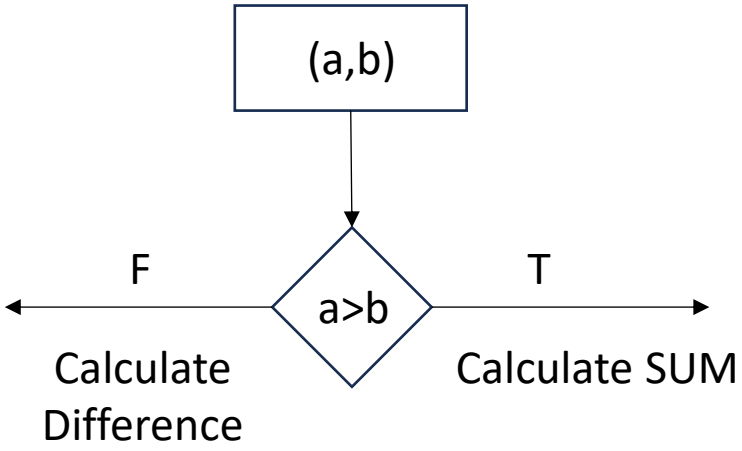
Test Selection & Adequacy Criteria – Example

- Consideration Criteria:
  - "A Test ***T*** for Program (***P***, ***R***), where ***R*** are the set of requirements, is considered adequate if each path in ***P*** is traversed at least once"



Test Selection & Adequacy Criteria – Example

- Consideration Criteria:
- "A Test ***T*** for Program (***P***, ***R***), where ***R*** are the set of requirements, is considered adequate if each path in ***P*** is traversed at least once"




```
int Test(int a, int b)
{
    if (a >b)
        return a+b;
    else
        return a-b;
}
```

Can you write a Test Case with 50% and 100% Coverage?

T1 = [ ]

Questions & Feedback?

Please feel free to email for your queries.

Course materials will be uploaded to Blackboard after every lecture.

setu.ie

INSPIRING FUTURES

Ollscoil Teicneolaíochta an Oirdheiscirt

South East Technological University