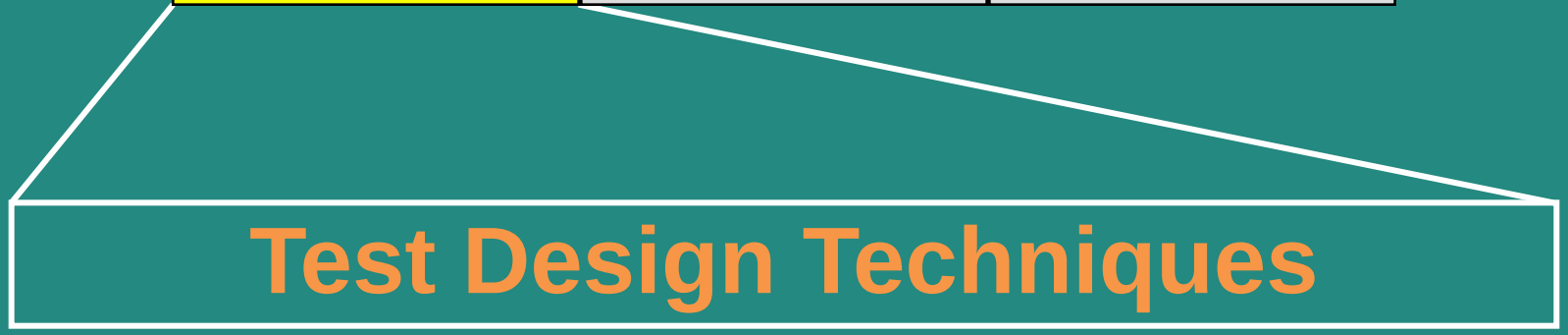
Chapter 4

Software Testing

Foundation Of Software Testing

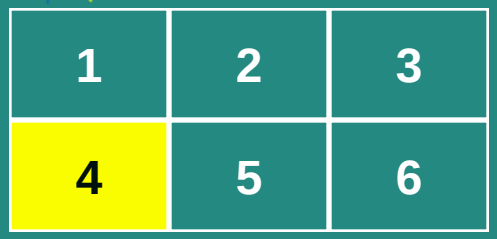
1 Principles	2 Lifecycle	3 Static testing
4 Test design techniques	5 Management	6 Tools



Contents

The test development process

- Formality of test documentation
- Test analysis: identifying test conditions
- Test design: specifying test cases
- Test implementation: specifying test procedures or scripts



Test Design Techniques

Contents

Three types of systematic technique

Static (non-execution)

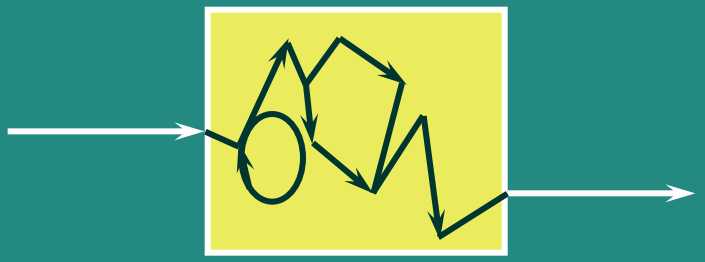
- **examination of documentation, source code listings, etc.**

Functional (Black Box)

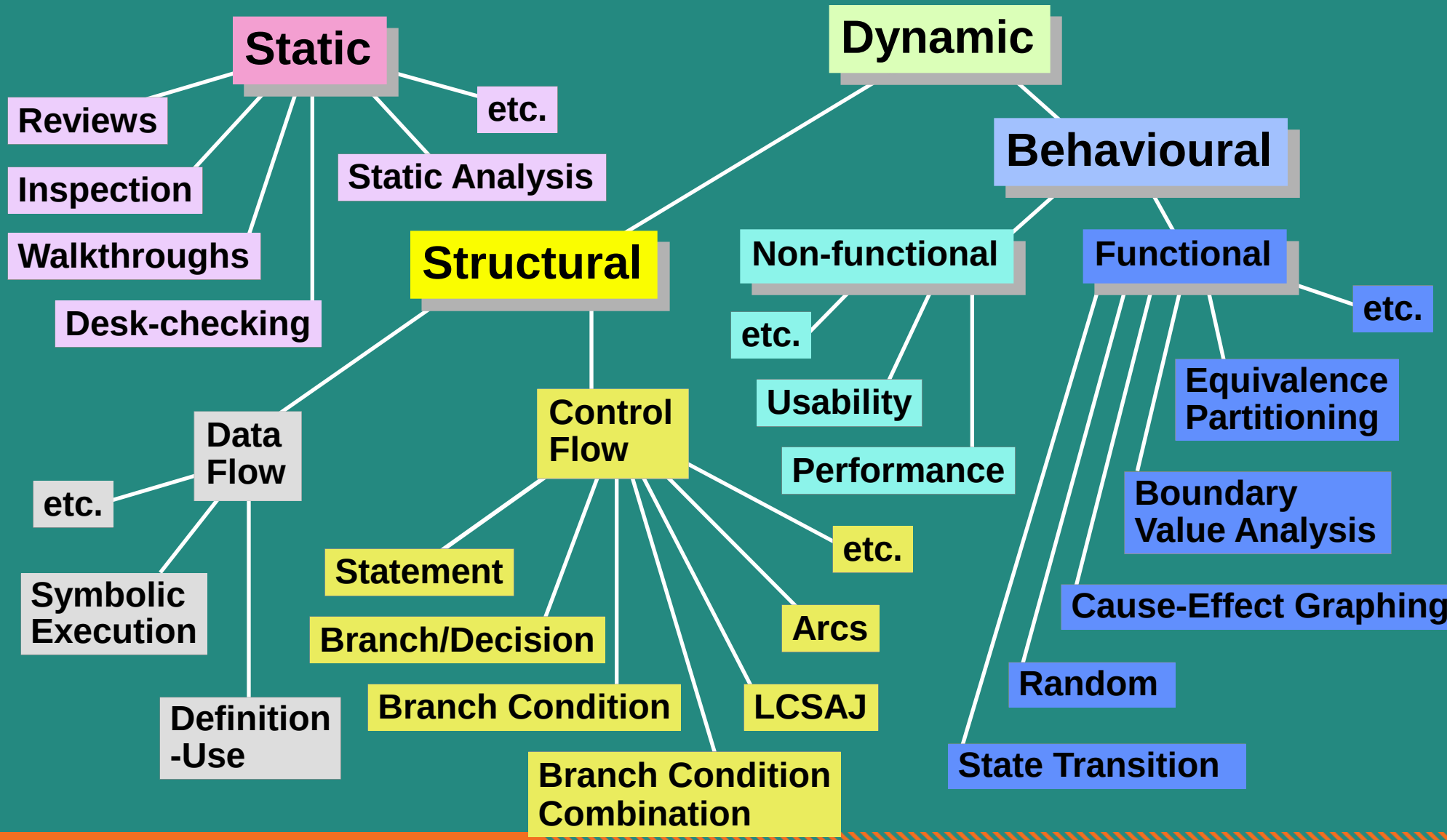
- **based on behaviour / functionality of software**

Structural (White Box)

- **based on structure of software**



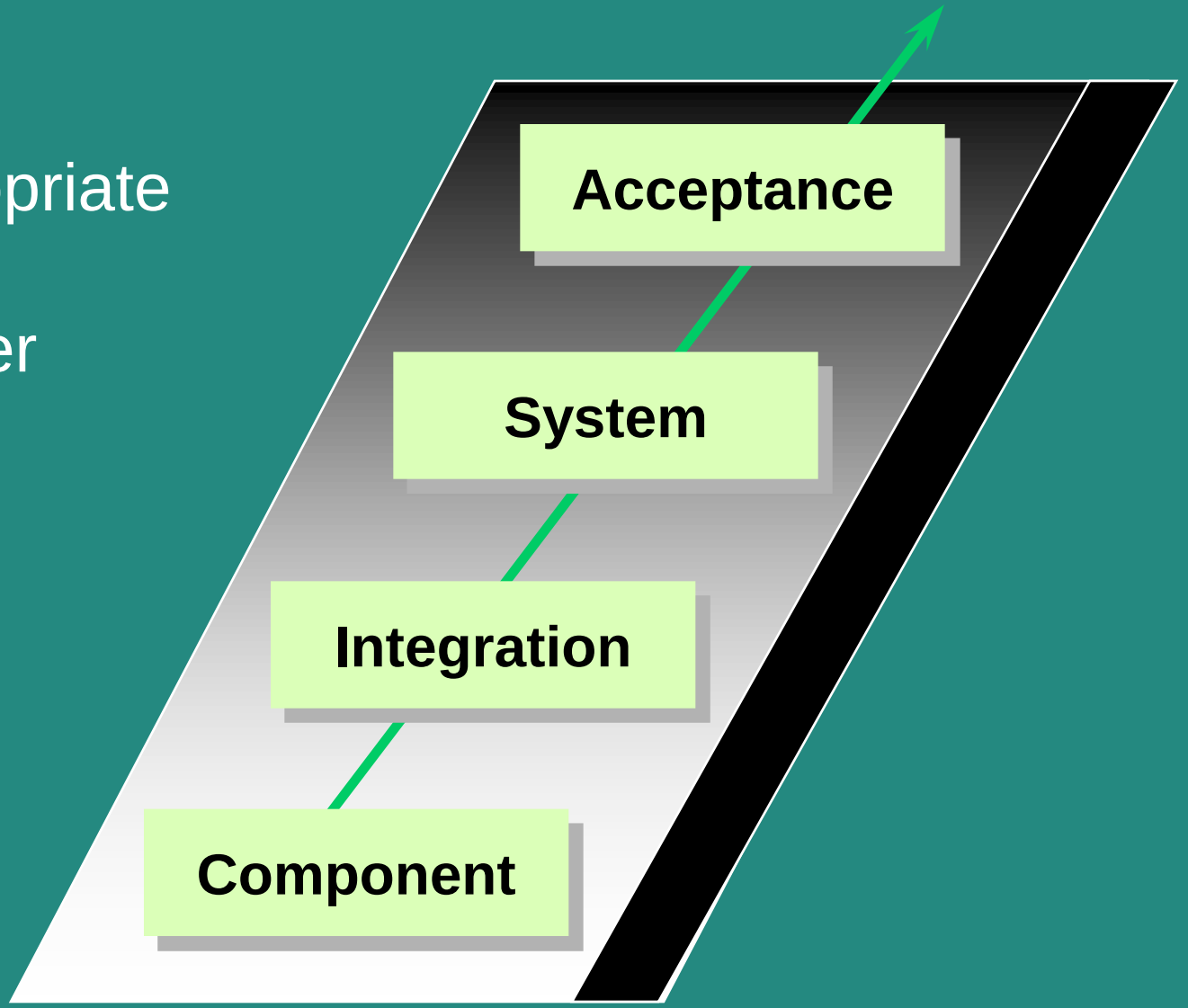
Some test techniques



Black box versus white box?

Black box appropriate at all levels but dominates higher levels of testing

White box used predominately at lower levels to compliment black box



Contents

Black Box test design and measurement techniques

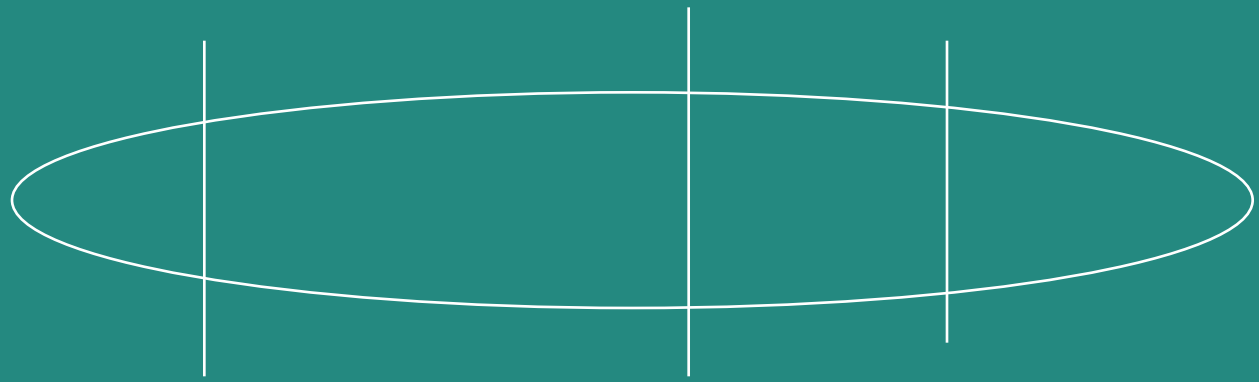
- Techniques defined in BS 7925-2
  - Equivalence partitioning ✓
  - Boundary value analysis ✓
  - State transition testing ✓
  - Cause-effect graphing ✓
  - Syntax testing ✗
  - Random testing ✗
- Also defines how to specify other techniques

Also a measurement technique?

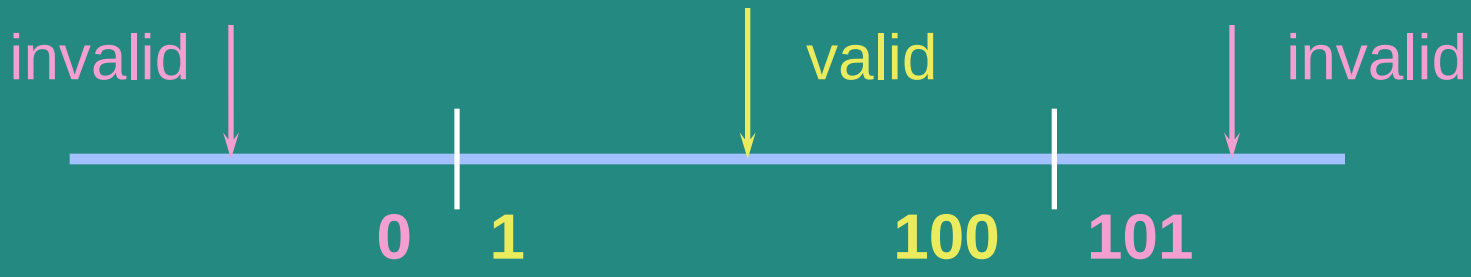
✓ = Yes

✗ = No

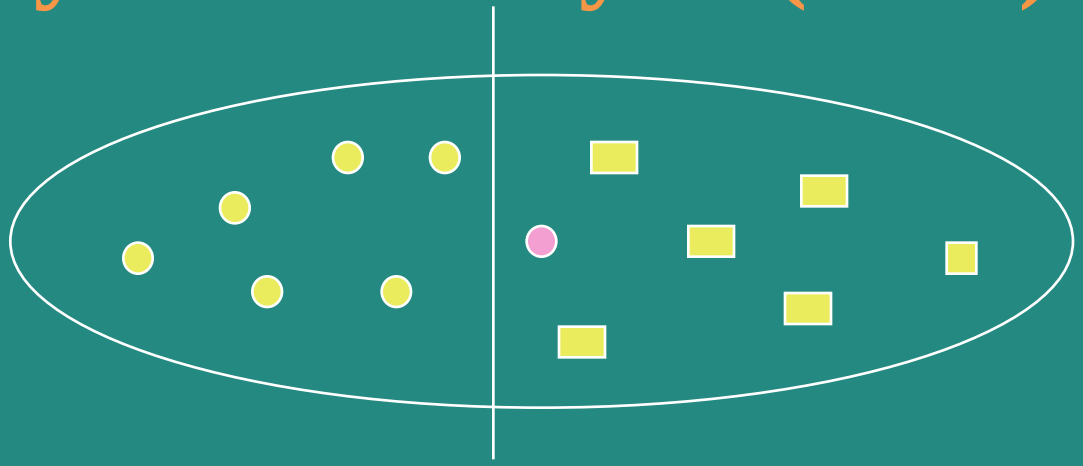
Equivalence partitioning (EP)



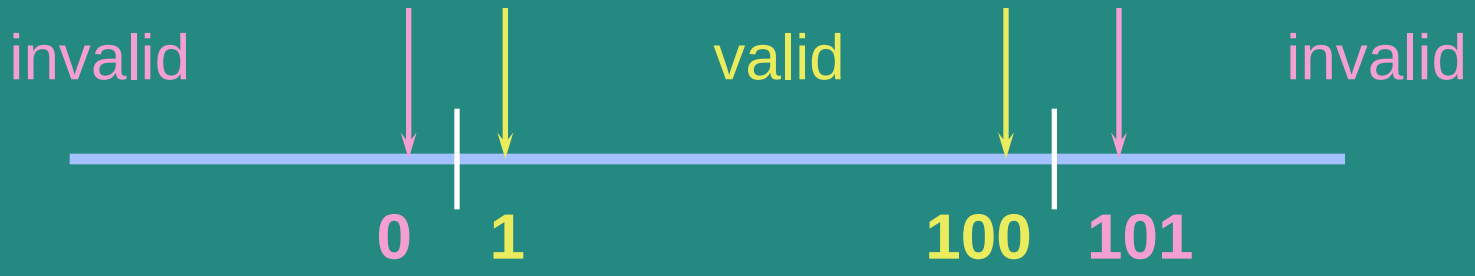
- divide (partition) the inputs, outputs, etc. into areas which are the same (equivalent)
- assumption: if one value works, all will work
- one from each partition better than all from one



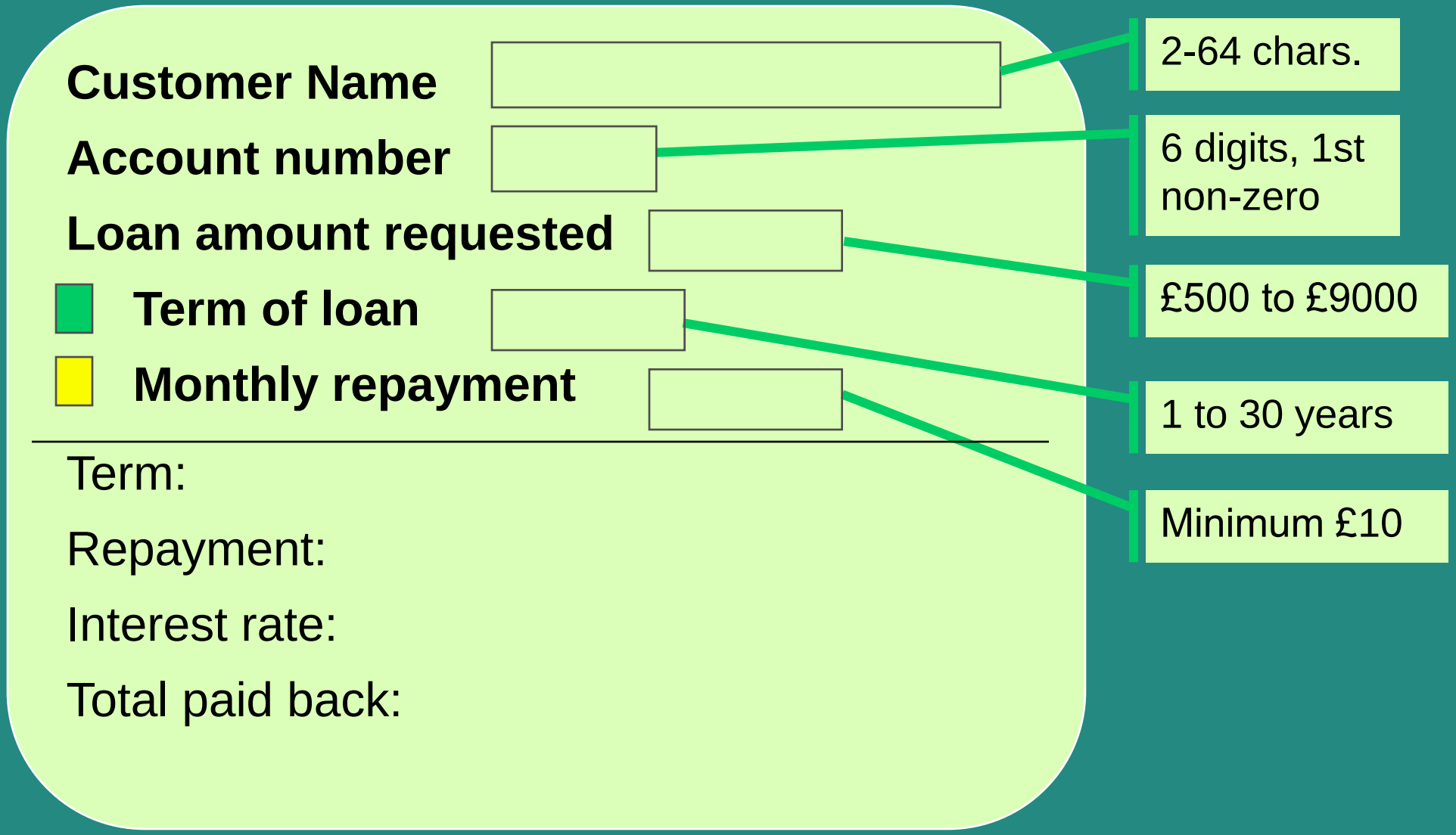
Boundary value analysis (BVA)



- faults tend to lurk near boundaries
- good place to look for faults
- test values on both sides of boundaries



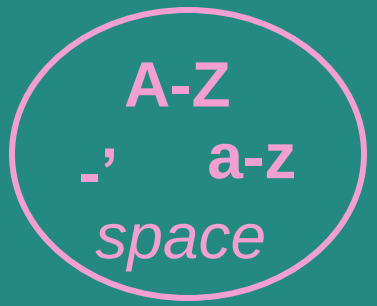
Example: Loan application



Customer name

Number of characters:

Valid characters:



Any other

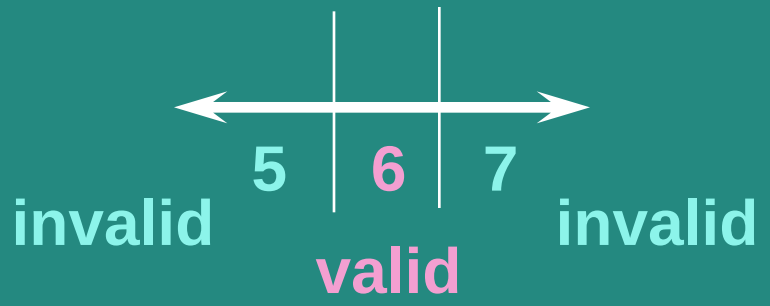
Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Customer name	2 to 64 chars	< 2 chars	2 chars	1 chars
	valid chars	> 64 chars	64 chars	65 chars
		invalid chars		0 chars

Account number

first character: valid: non-zero

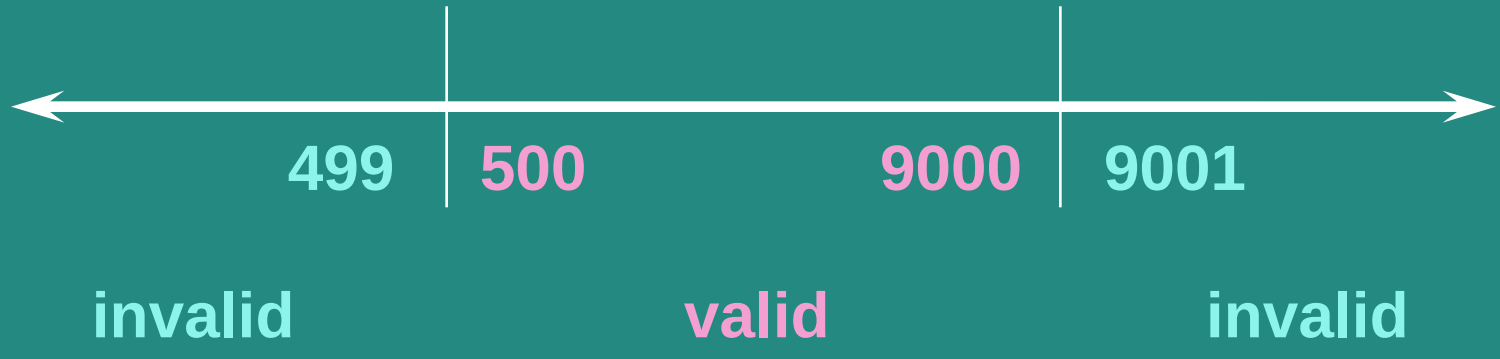
invalid: zero

number of digits:



Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Account number	6 digits	< 6 digits	100000	5 digits
	1st non-zero	> 6 digits	999999	7 digits
		1st digit = 0		0 digits
		non-digit		

Loan amount



Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Loan amount	500 - 9000	< 500 >9000 0 non-numeric null	500 9000	499 9001

Condition template

Conditions	Valid Partitions	Tag	Invalid Partitions	Tag	Valid Boundaries	Tag	Invalid Boundaries	Tag
Customer name	2 - 64 chars	V1	< 2 chars	X1	2 chars	B1	1 char	D1
	valid chars	V2	> 64 chars	X2	64 chars	B2	65 chars	D2
			invalid char	X3			0 chars	D3
Account number	6 digits	V3	< 6 digits	X4	100000	B3	5 digits	D4
	1st non-zero	V4	> 6 digits	X5	999999	B4	7 digits	D5
			1st digit = 0	X6			0 digits	D6
			non-digit	X7				
Loan amount	500 - 9000	V5	< 500	X8	500	B5	499	D7
			>9000	X9	9000	B6	9001	D8
			0	X10				
			non-integer	X11				
			null	X12				

Design test cases

Test Case	Description	Expected Outcome	New Tags Covered
1	Name: John Smith Acc no: 123456 Loan: 2500 Term: 3 years	Term: 3 years Repayment: 79.86 Interest rate: 10% Total paid: 2874.96	V1, V2, V3, V4, V5 .....
2	Name: AB Acc no: 100000 Loan: 500 Term: 1 year	Term: 1 year Repayment: 44.80 Interest rate: 7.5% Total paid: 537.60	B1, B3, B5, ...

Why do both EP and BVA?

- If you do boundaries only, you have covered all the partitions as well
  - technically correct and may be OK if everything works correctly!
  - if the test fails, is the whole partition wrong, or is a boundary in the wrong place - have to test mid-partition anyway
  - testing only extremes may not give confidence for typical use scenarios (especially for users)
  - boundaries may be harder (more costly) to set up

Test objectives?

Condition	Valid Partition	Tag	Invalid Partition	Tag	Valid Boundary	Tag	Invalid Boundary	Tag

- **For a thorough approach: VP, IP, VB, IB**
- **Under time pressure, depends on your test objective**
  - minimal user-confidence: VP only?
  - maximum fault finding: VB first (plus IB?)

Decision tables

- **explore combinations of inputs, situations or events,**
- **it is very easy to overlook specific combinations of input**
- **start by expressing the input conditions of interest so that they are either TRUE or FALSE**

  – record found
  – file exists
  – code valid
  – policy expired
  – account in credit
  – due date > current date

Example: student access

A university computer system allows students an allocation of disc space depending on their projects.

If they have used all their allotted space, they are only allowed restricted access, i.e. to delete files, not to create them. This is assuming they have logged on with a valid username and password.

What are the input and output conditions?

List the input and output conditions

- list the ‘input conditions’ in the first column of the table
- list the ‘output conditions’ under the input conditions

Input Conditions	
Valid username	
Valid password	
Account in credit	
Output Conditions	
Login accepted	
Restricted access	

Determine input combinations

- add columns to the table for each unique combination of input conditions.
- each entry in the table may be either ‘T’ for true, ‘F’ for false.

Input Conditions								
Valid username	T	T	T	T	F	F	F	F
Valid password	T	T	F	F	T	T	F	F
Account in credit	T	F	T	F	T	F	T	F

Rationalise input combinations

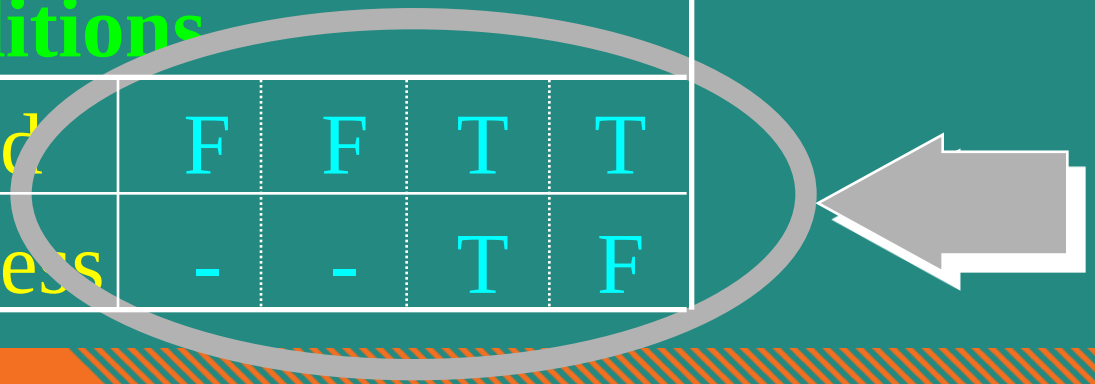
- some combinations may be impossible or not of interest
- some combinations may be ‘equivalent’
- use a hyphen to denote “don’t care”

Input Conditions				
Valid username	F	T	T	T
Valid password	-	F	T	T
Account in credit	-	-	F	T

Complete the table

- **determine the expected output conditions for each combination of input conditions**

Input Conditions				
Valid username	F	T	T	T
Valid password	-	F	T	T
Account in credit	-	-	F	T
Output Conditions				
Login accepted	F	F	T	T
Restricted access	-	-	T	F



Determine test case groups

- each column is at least one test case

Input Conditions				
Valid username	F	T	T	T
Valid password	-	F	T	T
Account in credit	-	-	F	T
Output Conditions				
Login accepted	F	F	T	T
Restricted access	-	-	T	F
Tags	**A**	**B**	**C**	**D**

Design test cases

- **usually one test case for each column but can be none or several**

Test	Description	Expected Outcome	Tag
1	Username BrbU	Invalid username	A
2	Username usernametoolong	Invalid username	A
3	Username BobU Password abcd	Invalid password	B
4	Valid user, no disc space	Restricted access	C
5	Valid user with disc space	Unrestricted access	D

Rationalising outputs

- if outputs or effects are mutually exclusive, I.e. T occurs in only one place in each column, we can combine them
- for example:

X	T	F	F
Y	F	T	F
Z	F	F	T

is equivalent to:

Output	X	Y	Z

Rationalising dangers

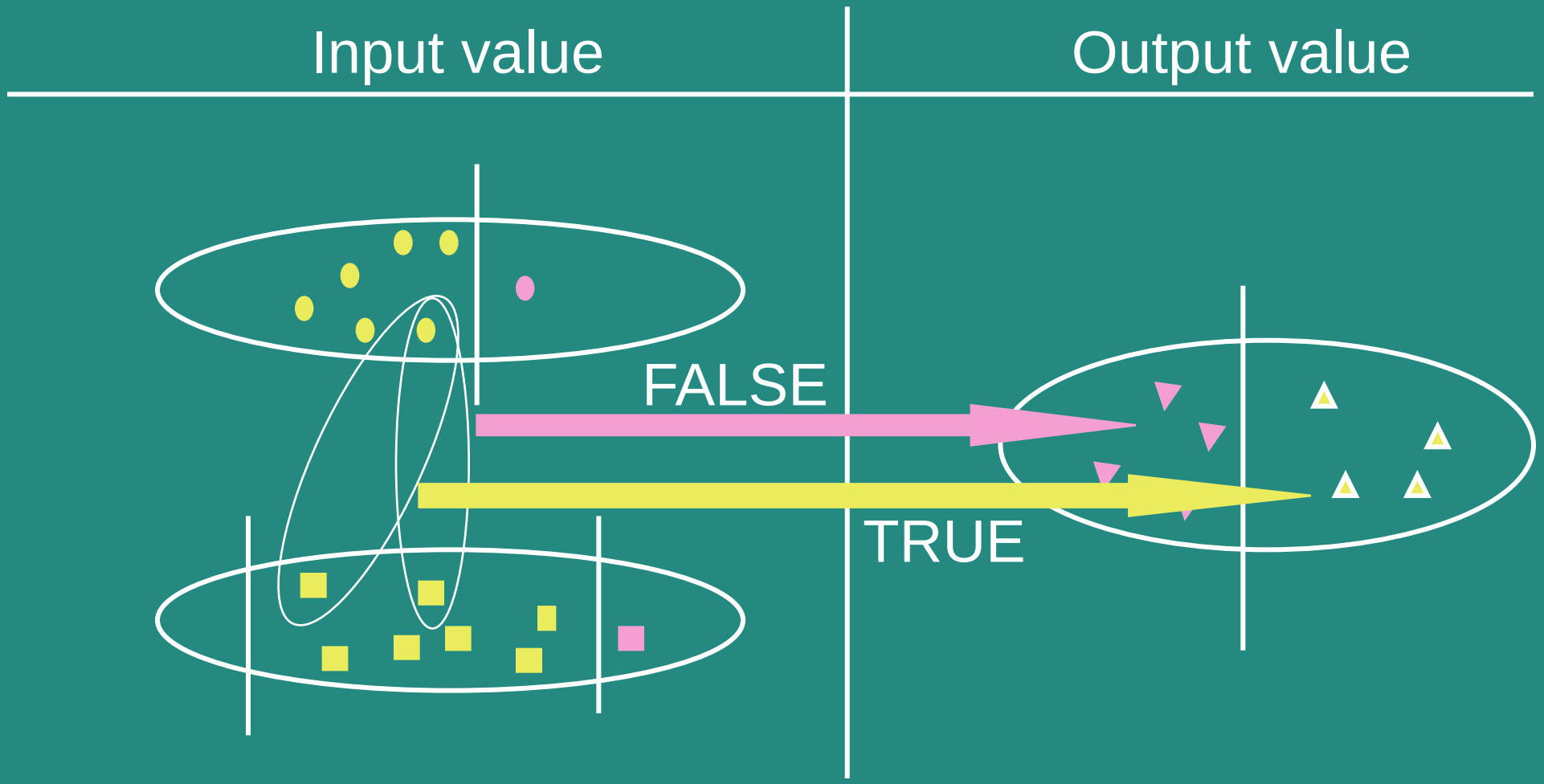
- Rationalising is based on assumptions
- Assumptions may be wrong!
- Assumptions should be stated
- Assumptions may change over time
- Be aware of the dangers
  - Filling in the full table may find errors which will be missed if you rationalise
  - It is possible to rationalise too far

Extending decision tables

- Entries can be more than just ‘true’ or ‘false’
  - completing table needs to be done carefully
  - rationalising becomes more important
- E.g.

Code = 1, 2, or 3	1	1	1	1	2	2	2	2	3	3	3	3
Exp.date < now	T	T	F	F	T	T	F	F	T	T	F	F
Class A product	T	F	T	F	T	F	T	F	T	F	T	F

Decision Tables in relation to EP and BVA



Contents

White Box test design and measurement techniques

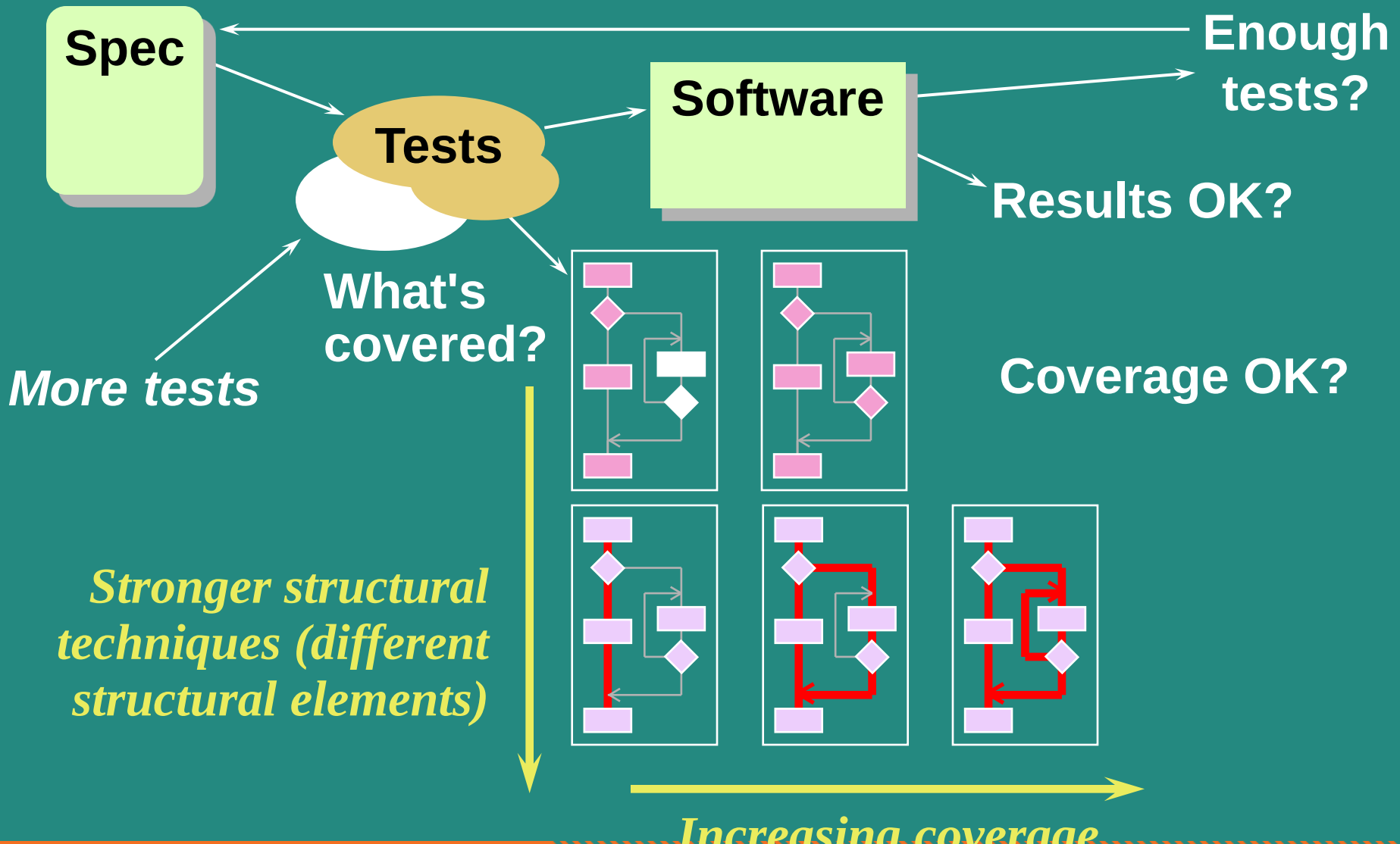
- Techniques defined in BS 7925-2
  - Statement testing ✓
  - Branch / Decision testing ✓
  - Data flow testing ✓
  - Branch condition testing ✓
  - Branch condition combination testing ✓
  - Modified condition decision testing ✓
  - LCSAJ testing ✓
- Also defines how to specify other techniques

Also a measurement technique?

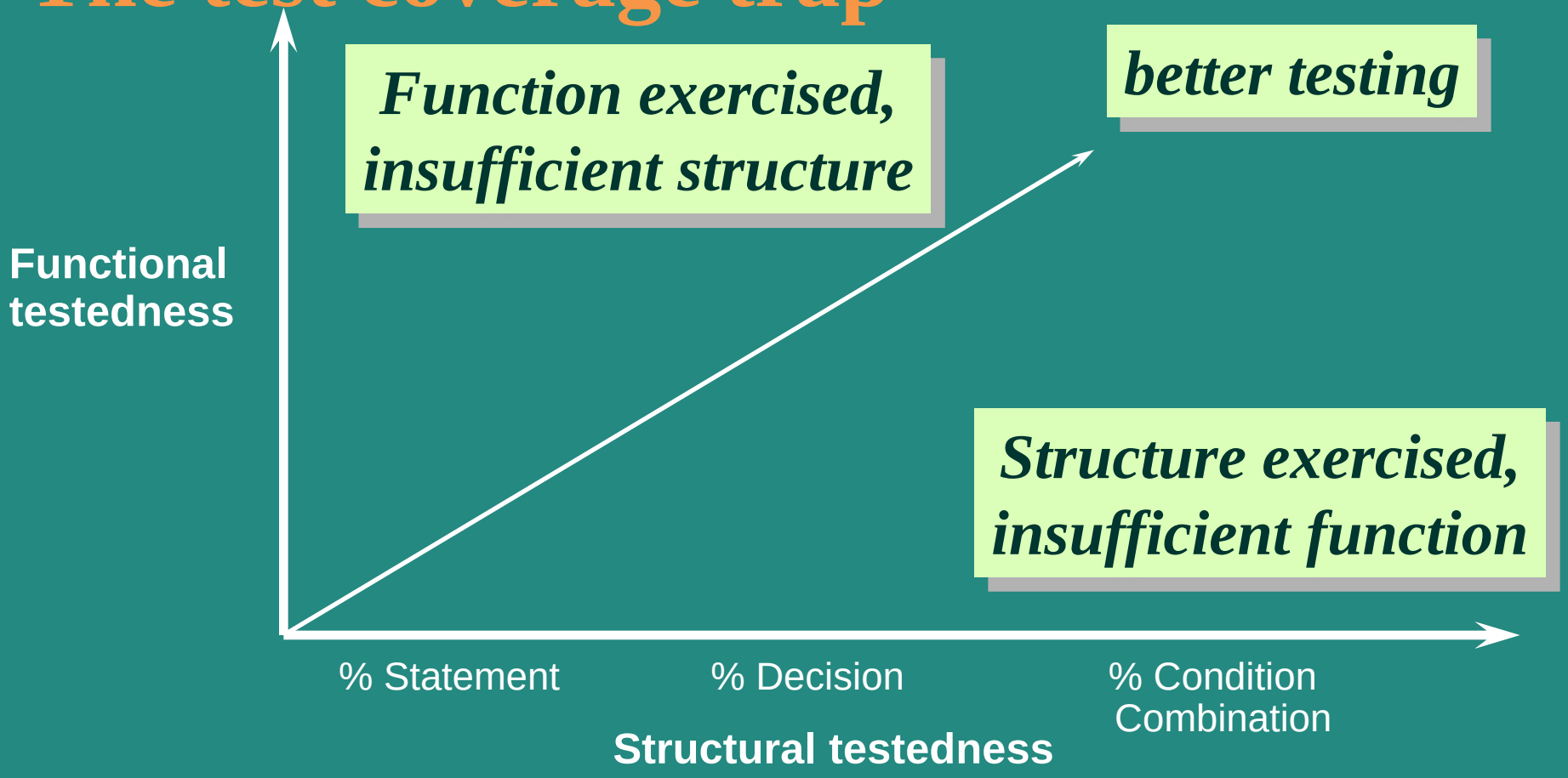
✓ = Yes

✗ = No

Using structural coverage



The test coverage trap



100% coverage does not mean 100% tested!

Coverage is not Thoroughness

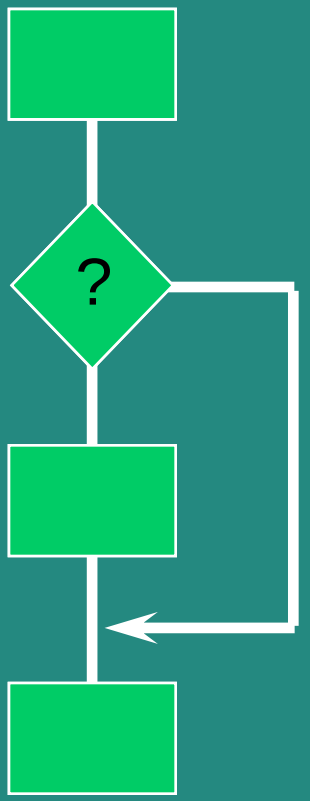
Statement coverage

Statement coverage is normally measured by a software tool.

- percentage of executable statements exercised by a test suite

$$= \frac{\text{number of statements exercised}}{\text{total number of statements}}$$

- example:
  - program has 100 statements
  - tests exercise 87 statements
  - statement coverage = 87%



Typical ad hoc testing achieves 60 - 75%

Example of statement coverage

1	**read(a)**
2	**IF a > 6 THEN**
3	**b = a**
4	**ENDIF**
5	**print b**

Test case	Input	Expected output
1	7	7

Statement numbers

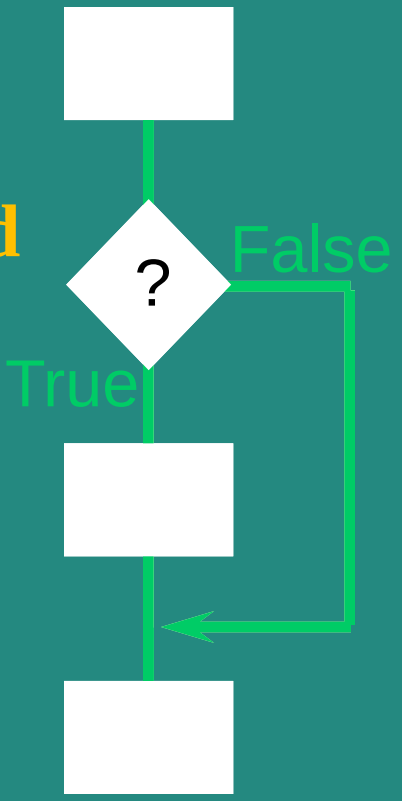
As all 5 statements are ‘covered’ by this test case, we have achieved 100% statement coverage

Decision coverage (Branch coverage)

Decision coverage is normally measured by a software tool.

- percentage of decision outcomes exercised by a test suite

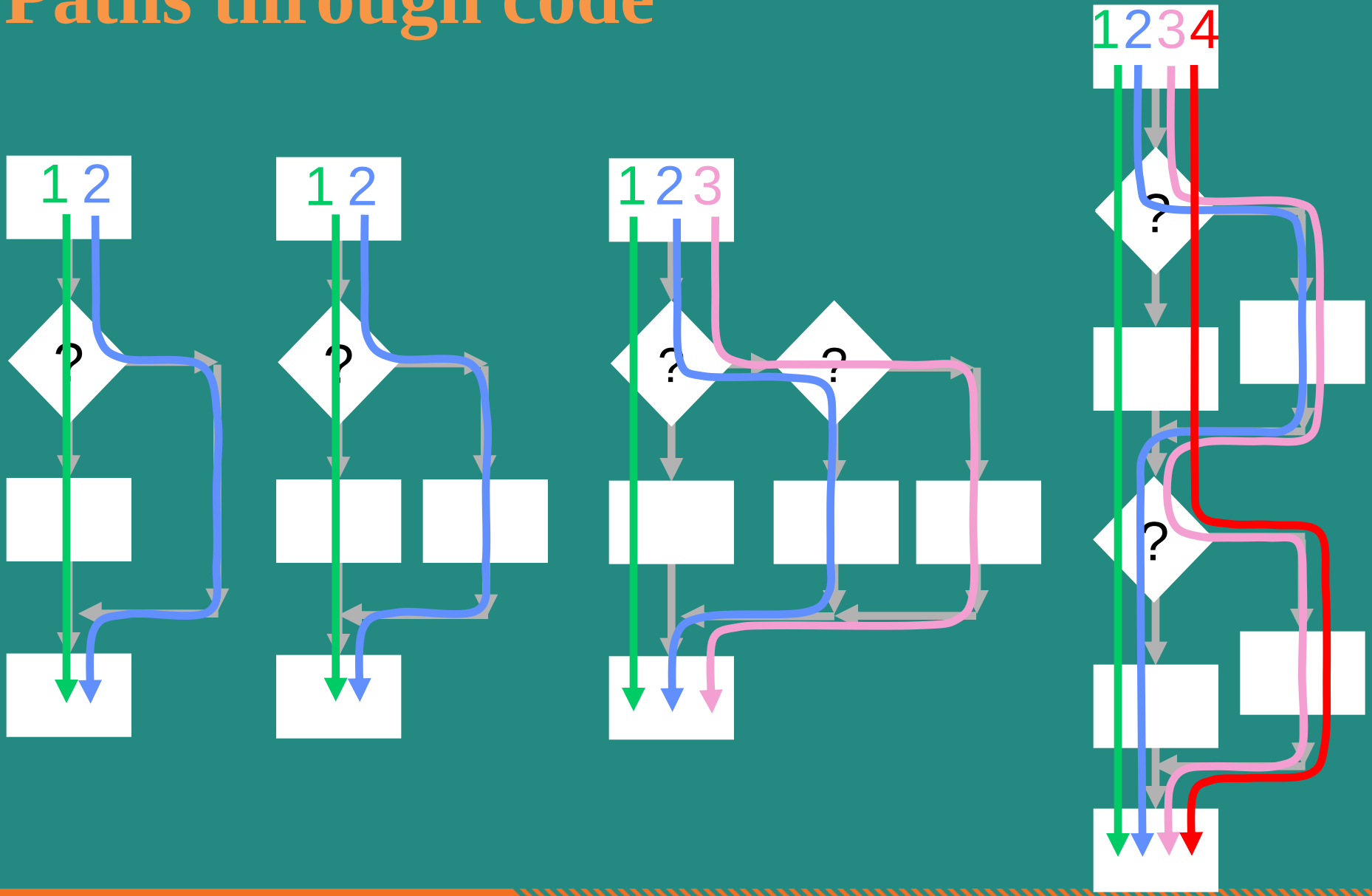
$$= \frac{\text{number of decisions outcomes exercised}}{\text{total number of decision outcomes}}$$



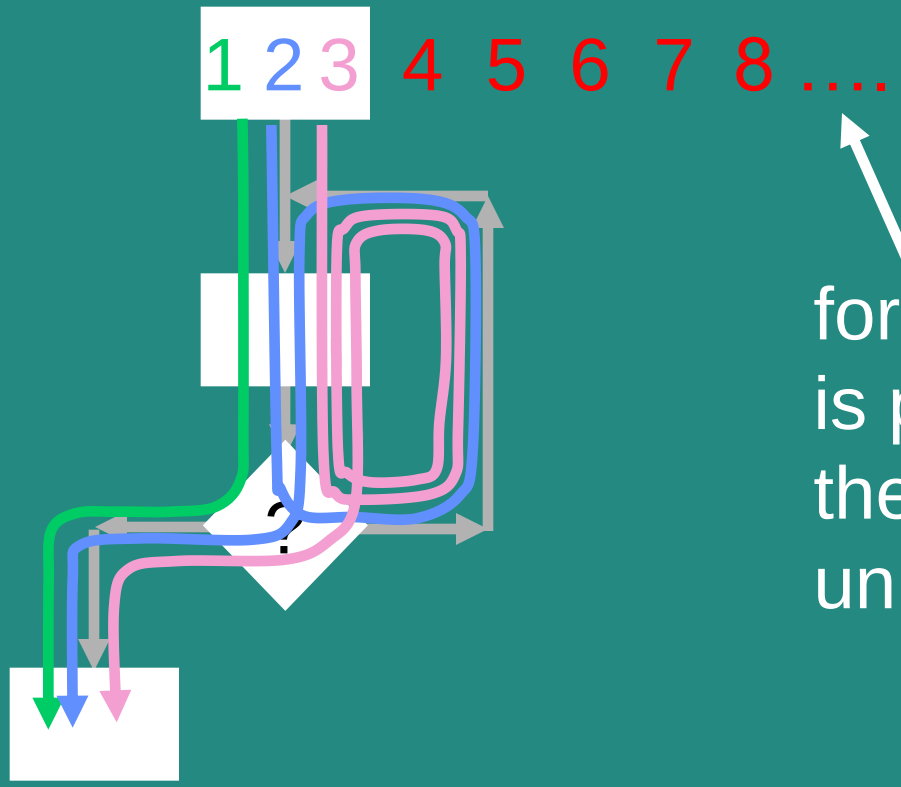
- example:
  - program has 120 decision outcomes
  - tests exercise 60 decision outcomes
  - decision coverage = 50%

Typical ad hoc testing achieves 40 - 60%

Paths through code



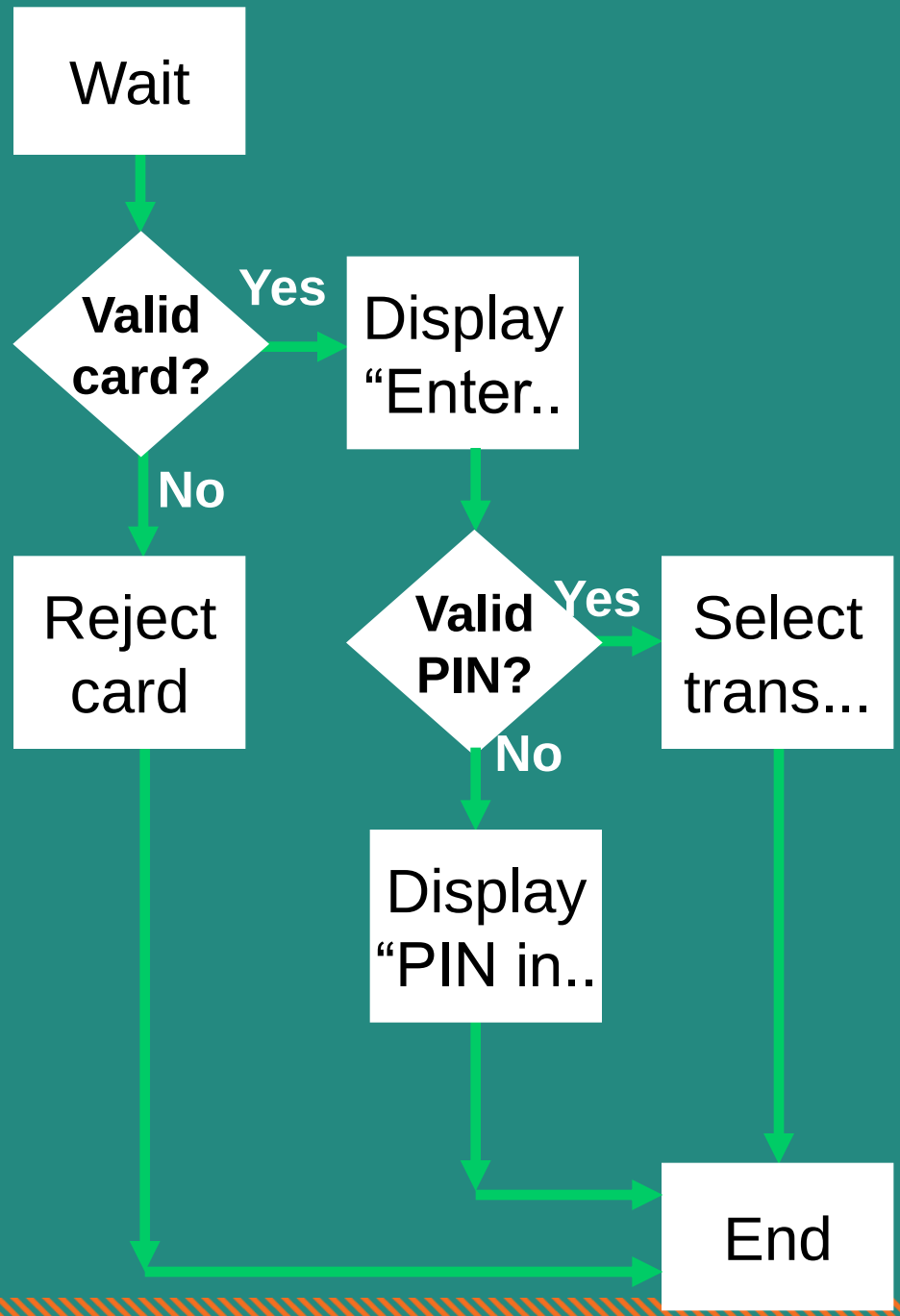
Paths through code with loops



for as many times as it is possible to go round the loop (this can be unlimited, i.e. infinite)

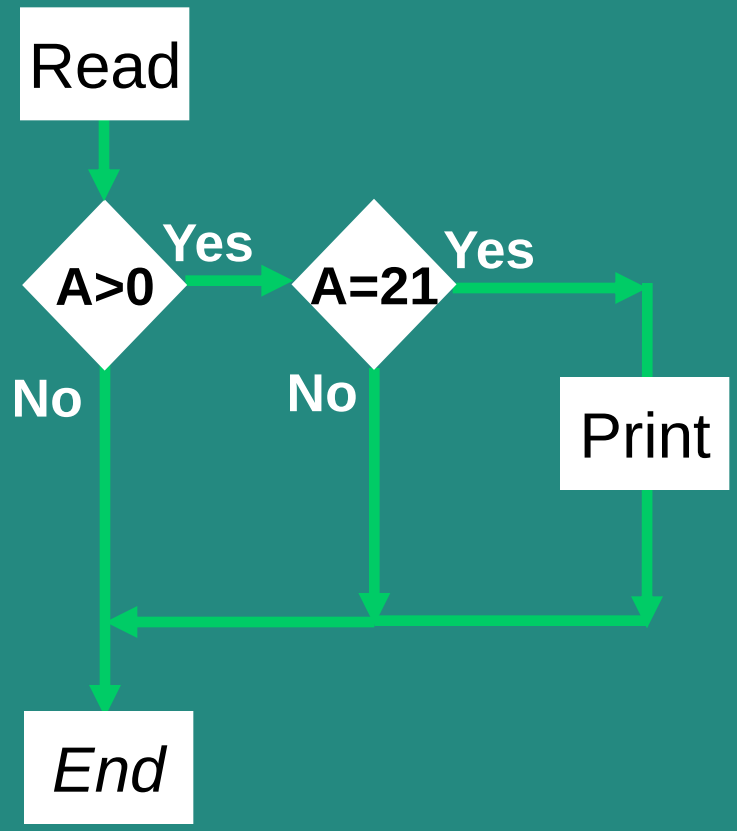
Example 1

Wait for card to be inserted
IF card is a valid card THEN
    display “Enter PIN number”
    IF PIN is valid THEN
        select transaction
    ELSE (otherwise)
        display “PIN invalid”
ELSE (otherwise)
    reject card
End



Example 2

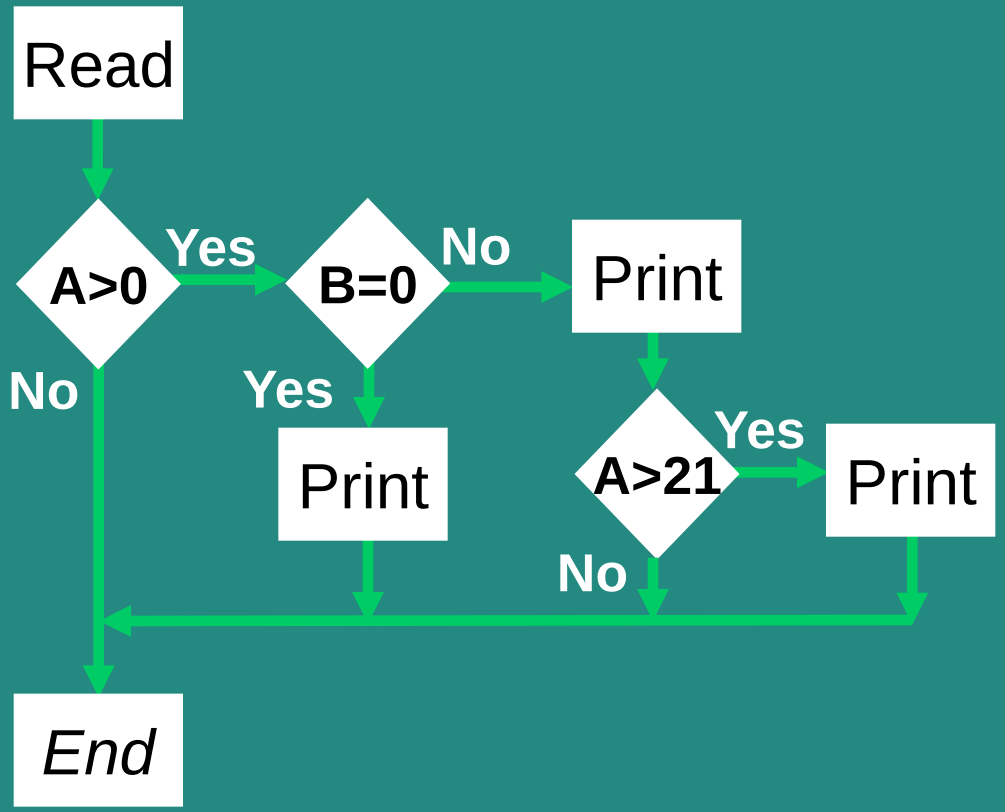
```
Read A
IF A > 0 THEN
    IF A  = 21 THEN
        Print “Key”
    ENDIF
ENDIF
```



- **Cyclomatic complexity:** 3
- **Minimum tests to achieve:**
  - Statement coverage: 1
  - Branch coverage: 3

Example 3

```
Read A
Read B
IF A > 0 THEN
    IF B  = 0 THEN
        Print “No values”
    ELSE
        Print B
        IF A > 21 THEN
            Print A
        ENDIF
    ENDIF
ENDIF
```

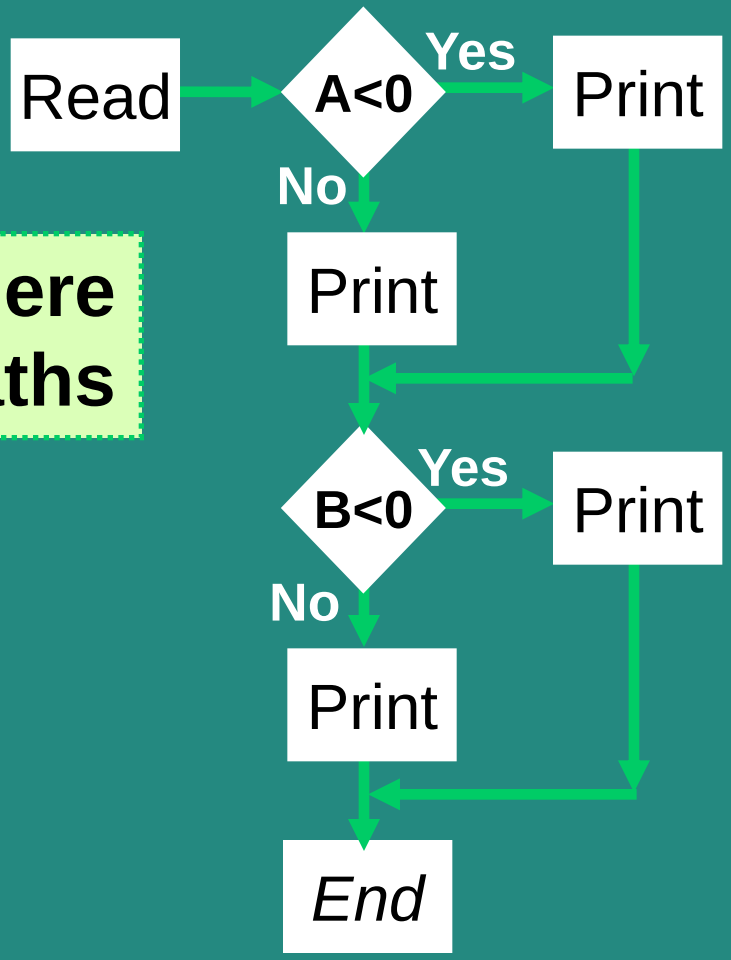


- **Cyclomatic complexity:** 4
- **Minimum tests to achieve:**
  - Statement coverage: 2
  - Branch coverage: 4

Example 4

```
Read A
Read B
IF A < 0 THEN
    Print  “A negative”
ELSE
    Print  “A positive”
ENDIF
IF B < 0 THEN
    Print  “B negative”
ELSE
    Print  “B positive”
ENDIF
```

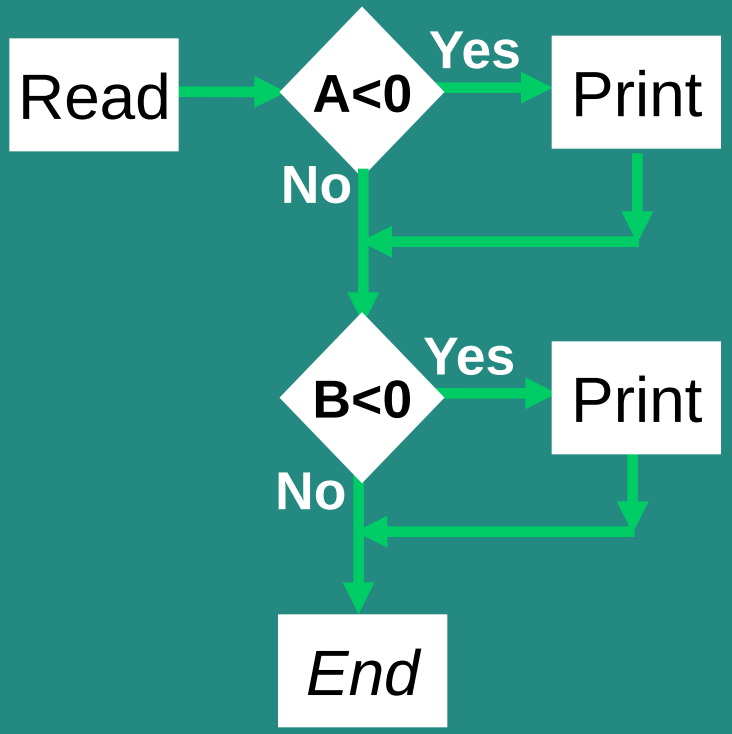
Note: there are 4 paths



- **Cyclomatic complexity:** 3
- **Minimum tests to achieve:**
  - Statement coverage: 2
  - Branch coverage: 2

Example 5

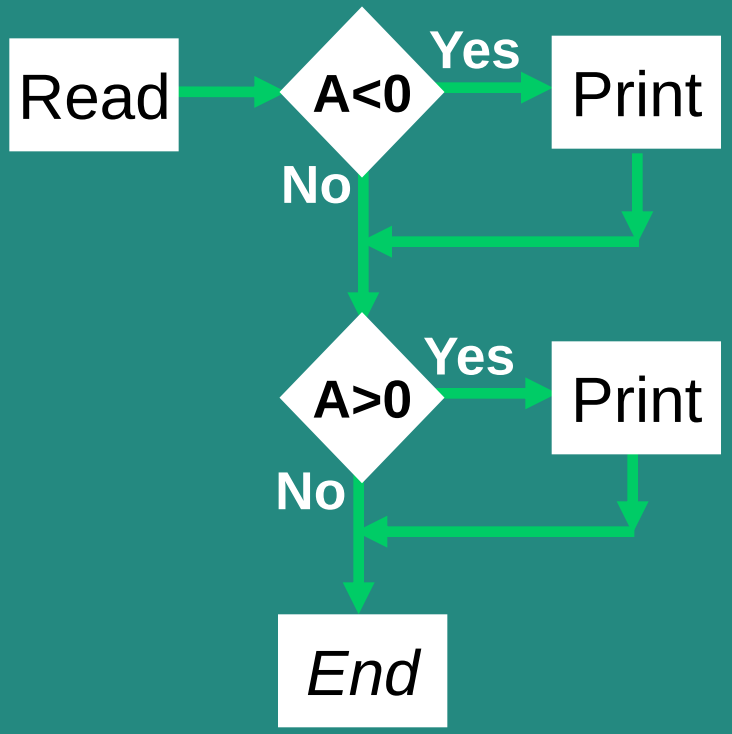
```
Read A
Read B
IF A < 0 THEN
    Print  “A negative”
ENDIF
IF B < 0 THEN
    Print  “B negative”
ENDIF
```



- **Cyclomatic complexity:** 3
- **Minimum tests to achieve:**
  - Statement coverage: 1
  - Branch coverage: 2

Example 6

```
Read A
IF A < 0 THEN
    Print  “A negative”
ENDIF
IF A > 0 THEN
    Print  “A positive”
ENDIF
```



- **Cyclomatic complexity:** 3
- **Minimum tests to achieve:**
  - Statement coverage: 2
  - Branch coverage: 2

Contents

Experience-based techniques

- **Error guessing and fault attacks:** Error guessing is a technique that should always be used as a complement to other more formal techniques.
- **Exploratory testing:** is a hands-on approach in which testers are involved in minimum planning and maximum test execution.

Contents

Choosing a test technique

- The internal factors that influence the decision about which technique to use are:
  - ***Models used***
  - ***Tester knowledge/experience***
  - ***Likely defects***
  - ***Test objective***
  - ***Documentation***
  - ***Life cycle model***

Choosing a test technique

- The external factors that influence the decision about which technique to use are:
  - ***Risk***
  - ***Customer/contractual requirements***
  - ***Type of system***
  - ***Regulatory requirements***
  - ***Time and budget***

Summary: Key Points

The test development process

Categories of test design techniques

Black and White box testing

Black box test techniques

White box test techniques

Experience-based techniques

Choosing a test technique