Software Testing

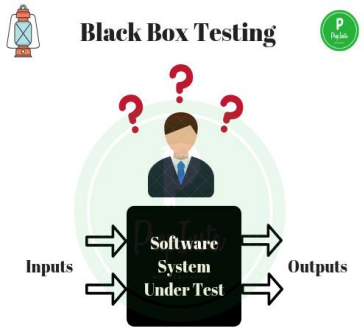
Lecture 19

Equivalence Class Portioning and Boundary Value Analysis

Chapter 5:
Dynamic Analysis-Test Design Techniques

Black Box Testing

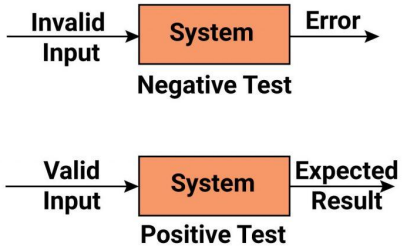
- **Motivation?**
- **Real life examples like Car, B**



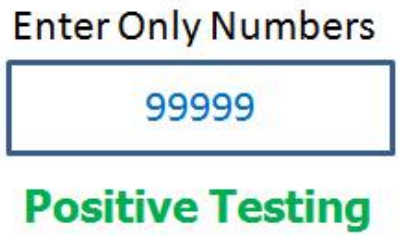
Black Box Testing

- Testing software without knowledge of internal implementation details
  - –Can be applied to software “units”
  - –External behavior is defined in functional requirements
- Focus on input and output
- Examines the functionality of an application
- Checks for valid and invalid conditions / inputs
- Finds defects like: Incorrect functions, behaviour etc.
- The Goal: Derive sets of input conditions (test cases) that fully exercise the external functionality

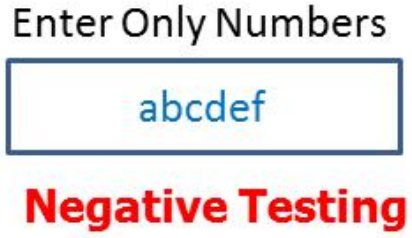
Positive and Negative testing



- **Positive testing to check that the product does what it is supposed to**
  - **- Behaves correctly when given right inputs**
  - **- Maps to a specific requirement“**
  - **- Coverage” is defined better**
  - **- Example: Text box in an application**



- **Negative testing to show that the product does not fail when given unexpected inputs**
  - **- Tries to break the system**
  - **- No direct mapping to a specific requirement“**
  - **- “Coverage” more challenging**
  - **- Example: Text box in an application**



Black Box Techniques

Equivalence Class Partitioning

Motivation:

- **Complete testing**
- **Avoid test redundancy**

Equivalence class partitioning:

- **Black box testing technique**
- **Partitions are frequently derived from the requirements**
- **Inputs to the software are divided into groups that are expected to exhibit similar behavior.**
- **Selecting one input from each group to design the test case.**
- **Can be used at any level of testing**

Equivalence Partitioning = Equivalence Class Partitioning = ECP

Equivalence Class Partitioning

- Test cases are designed to cover each partition at least once.
- Two test cases are equivalent
  - - Both follow same path
  - - Only select one test case
- It's technique where input values set into classes for testing.
  - - Valid Input Class = Keeps all valid inputs.
  - - Invalid Input Class = Keeps all Invalid inputs.

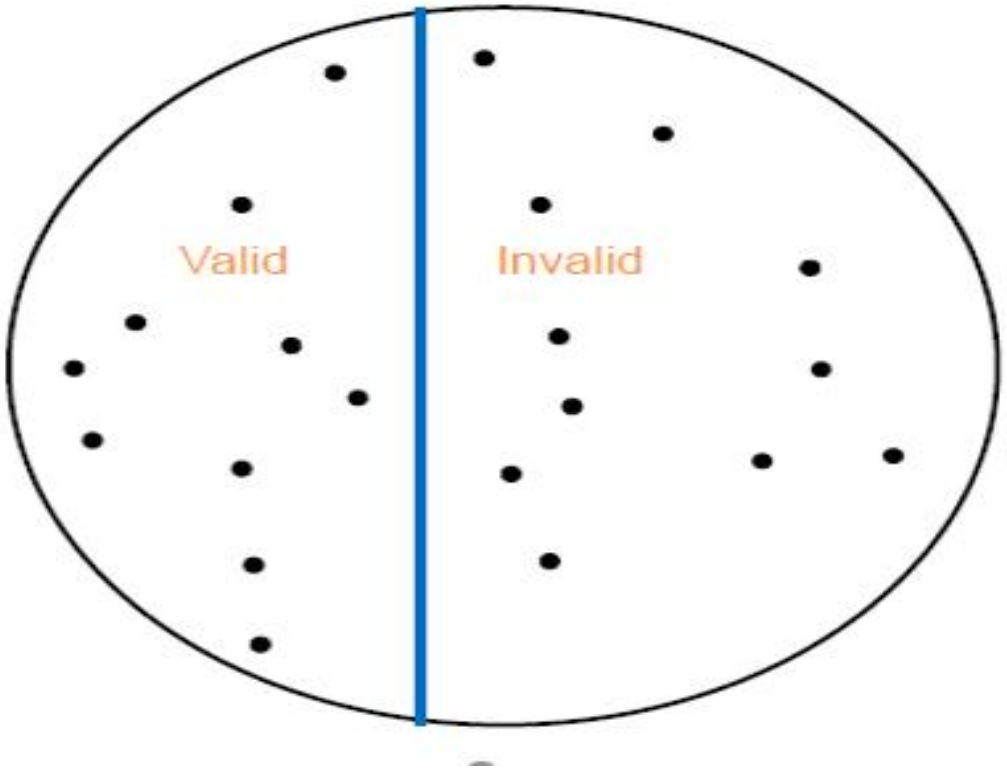
Equivalence Class Partitioning

- Partitions of the input set
- Entire input set is covered
- Disjoint classes
- Test cases
- Equivalence classes have to be chosen wisely
- All possible test cases is so large
- Select a relatively small number of test cases

- ***Which test cases should you choose?***
  - ***-Equivalence Classes***

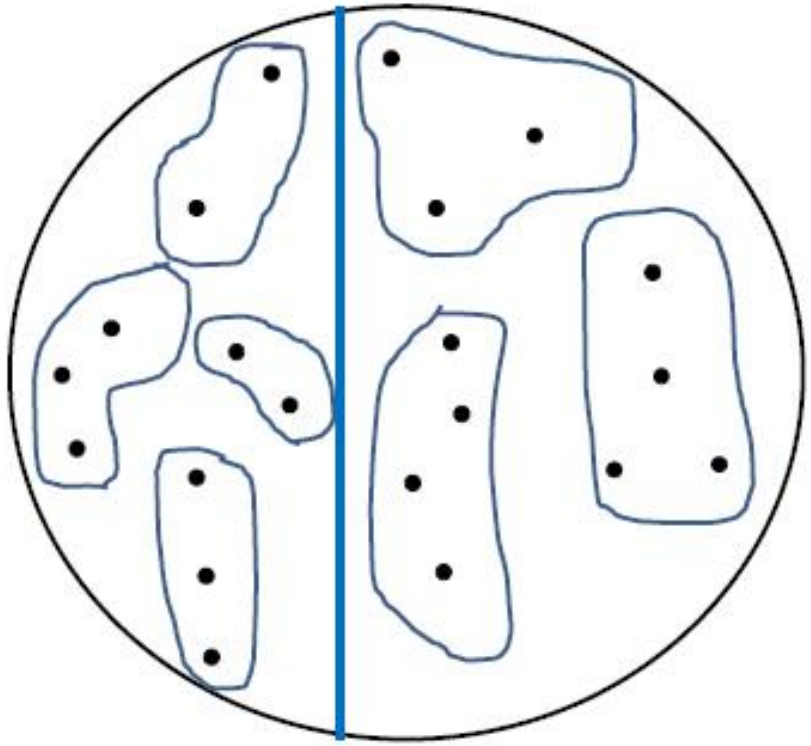
Equivalence Class Partitioning

- First level portioning: valid vs Invalid input/data



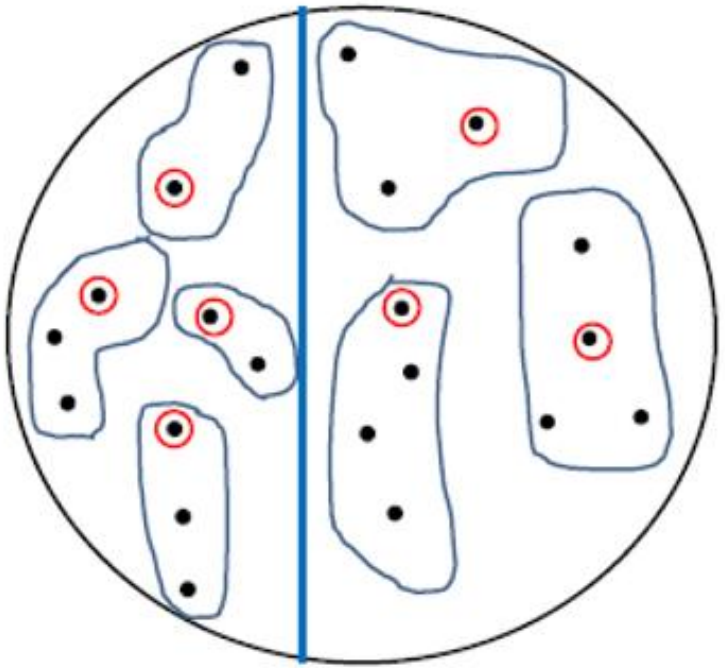
Equivalence Class Partitioning

- Partition valid and invalid data into equivalence classes



Equivalence Class Partitioning

- **Create a test case for atleast one value from each equivalence class**



Equivalence Class Partitioning Example 1

Assume, we have to test a field which accepts Age 18 – 56.

EQUIVALENCE PARTITIONING		
Invalid	**Valid**	**Invalid**
<=17	18-56	>=57

- **Valid Input: 18 – 56**
- **Invalid Input: <=17, >=57**
- **Valid Class 1: 18 – 56**
- **Invalid Class 2: <=17**
- **Invalid Class 3: >=57**
- **so we have one valid and 2 invalid classes**

Equivalence Class Partitioning Example 2

- Travel service offers discounts to travelers based on their age.
  - 0-4 years 100%
  - 5-15 years 50%
  - 16-64 years 0%
  - 64 years and older 25%
- Equivalence classes for age
  - 0, 1, 2, 3, 4
  - 5, 6, 7, ...15
  - 16, 17, 18, ...64
  - 65, 66, 67, ...120
- Similarly for destinations
  - E.g., destination can be grouped based on regions (that have same fair)
- Nothing special for name

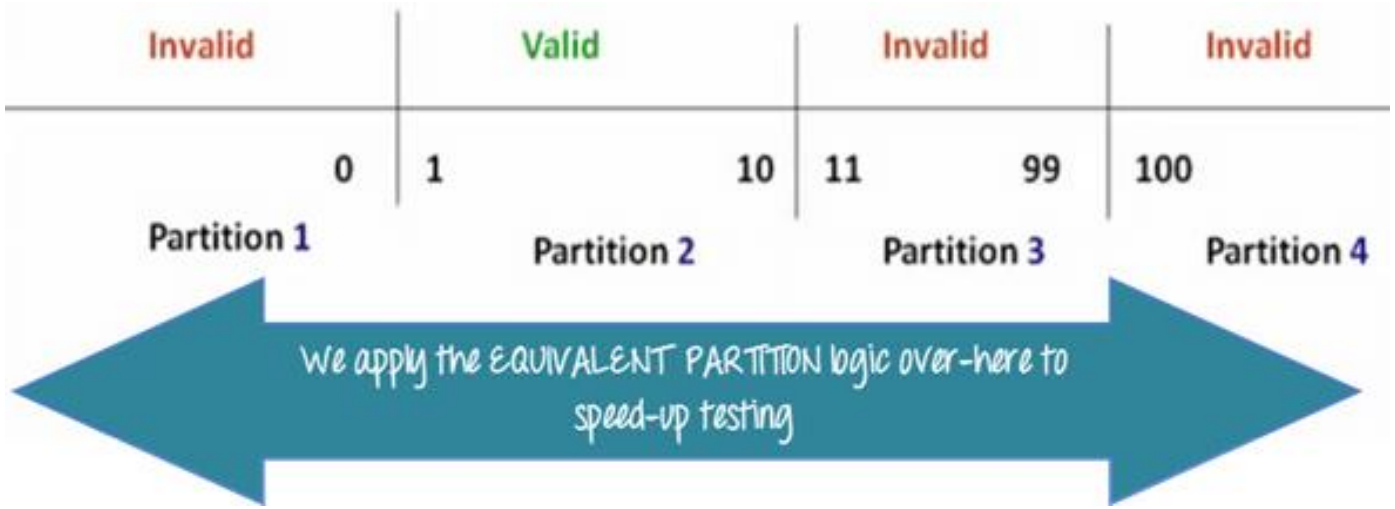
Equivalence Class Partitioning Example 3

- Pizza order application
- Requirements:
  - Pizza values 1 to 10 is considered valid. A success message is shown.
  - While value 11 to 99 are considered invalid for order and an error message will appear, "Only 10 Pizza can be ordered“

- Test Conditions and Partitions:
  - Any Number greater than 10 entered in the Order Pizza field(let say 11) is considered invalid.
  - Any Number less than 1 that is 0 or below, then it is considered invalid.
  - Numbers 1 to 10 are considered valid
  - Any 3 Digit Number say 100 is invalid.

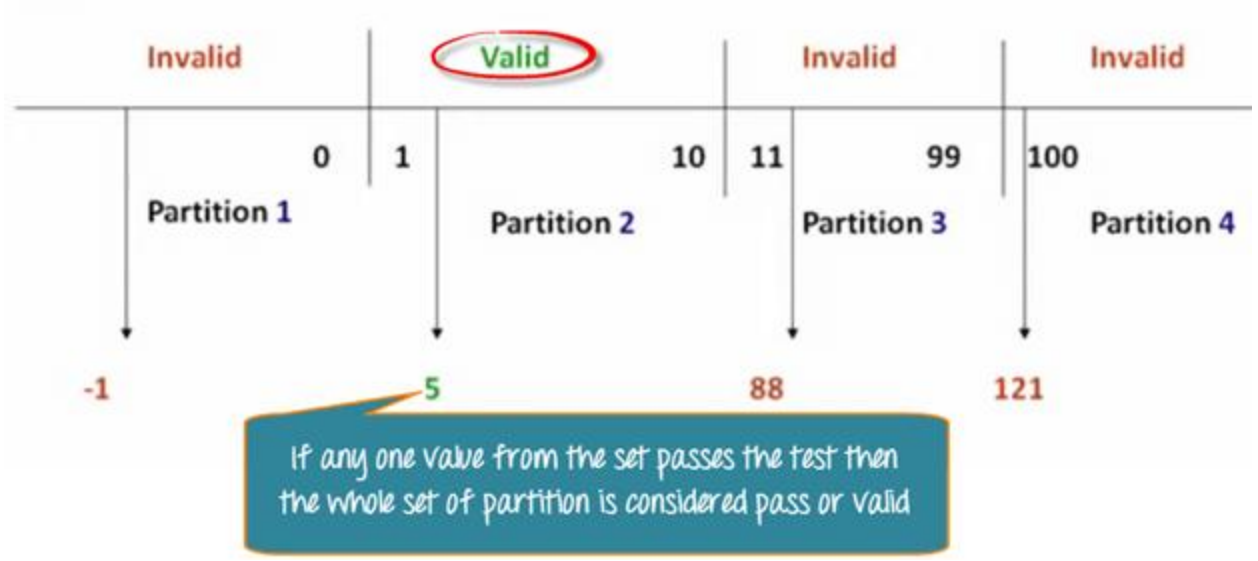
Equivalence Class Partitioning Example 3

- Divide the possible values of pizza number into groups or sets as shown below.



Equivalence Class Partitioning Example 3

- Pick only one value from each partition for testing



Equivalence Class Partitioning Example 3

S#	Equivalence Partition	Type of Input	Test Data	Expected Result
1	Number 1-10	Valid	10	“Successfully Ordered 10 Pizza”
2	Number < 1 or 0	Invalid	-1	“Invalid Order”
3	10<Number<=99	Invalid	88	“Only 10 Pizza can be ordered”
4	Number>100	Invalid	121	“Only 10 Pizza can be ordered”

Table. Equivalence partitioning for Pizza order application

Summary of the process of ECP

- Choose criteria for ECP (range, list of values, etc.).
- Identify the valid and invalid EC .
- Select a sample data from that partition.
- Write the expected result based on the requirements given.
- Identify special values, if any, and include them in the table.
- Check to have expected results for all the cases prepared
- If the expected result is not clear for any particular test case, mark appropriately and escalate for corrective actions.
- If you cannot answer a question, or find an inappropriate answer, consider whether you want to record this issue on your log.

Advantages and Disadvantages of ECP

Advantages:

- Reduces the number of test cases.
- Reduce execution time
- Can be applied at any level of testing
- Used where exhaustive testing is not possible
- Maintain good test coverage.

Disadvantages:

- Not consider the boundary conditions.
- Equivalence classes relies heavily on the expertise of the tester
- Too large partitions leads to risk of missing defects
- Assumption for result of correct input data set

Class Task

Consider a software module that is intended to accept the name of a grocery item and a list of the different sizes the item comes in, specified in ounces.

The specification states that:

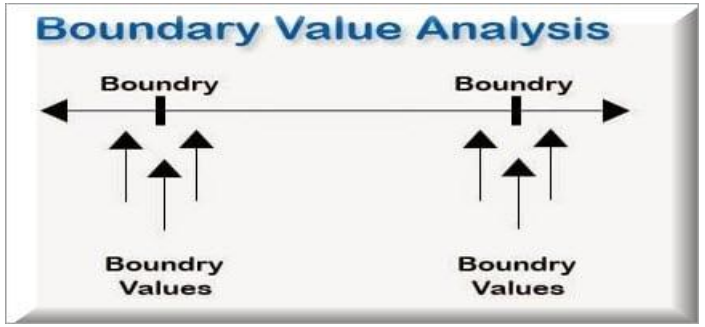
1. the item name is to be alphabetic characters 2 to 15 characters in length.
2. Each size may be a value in the range of 1 to 48, whole numbers only.
3. The sizes are to be entered in ascending order (smaller sizes first).
4. A maximum of five sizes may be entered for each item.
5. The item name is to be entered first, followed by a comma, then followed by a list of sizes.
6. A comma will be used to separate each size.
7. Spaces (blanks) are to be ignored anywhere in the input

Class Task (Solution)

Derived Equivalence Classes

1. Item name is alphabetic (valid)
2. Item name is not alphabetic (invalid)
3. Item name is less than 2 characters in length (invalid)
4. Item name is 2 to 15 characters in length (valid)
5. Item name is greater than 15 characters in length (invalid)
6. Size value is less than 1 (invalid)
7. Size value is in the range 1 to 48 (valid)
8. Size value is greater than 48 (invalid)
9. Size value is a whole number (valid)
10. Size value is a decimal (invalid)
11. Size value is numeric (valid)
12. Size value includes nonnumeric characters (invalid)
13. Size values entered in ascending order (valid)
14. Size values entered in nonascending order (invalid)
15. No size values entered (invalid)
16. One to five size values entered (valid)
17. More than five sizes entered (invalid)
18. Item name is first (valid)
19. Item name is not first (invalid)
20. A single comma separates each entry in list (valid)
21. A comma does not separate two or more entries in the list (invalid)
22. The entry contains no blanks (???)
23. The entry contains blanks (????)

Boundary Value Analysis



Motivation:

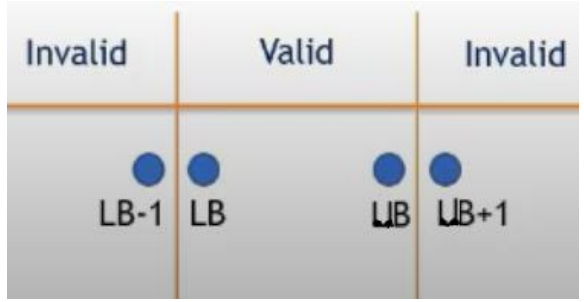
- Errors at extreme ends.

Boundary Value Analysis:

- Based on testing boundary values
- Maximum and minimum of every partition
- Covers both valid and invalid boundary values
- Used to identify errors that arise due to the limits of input data.

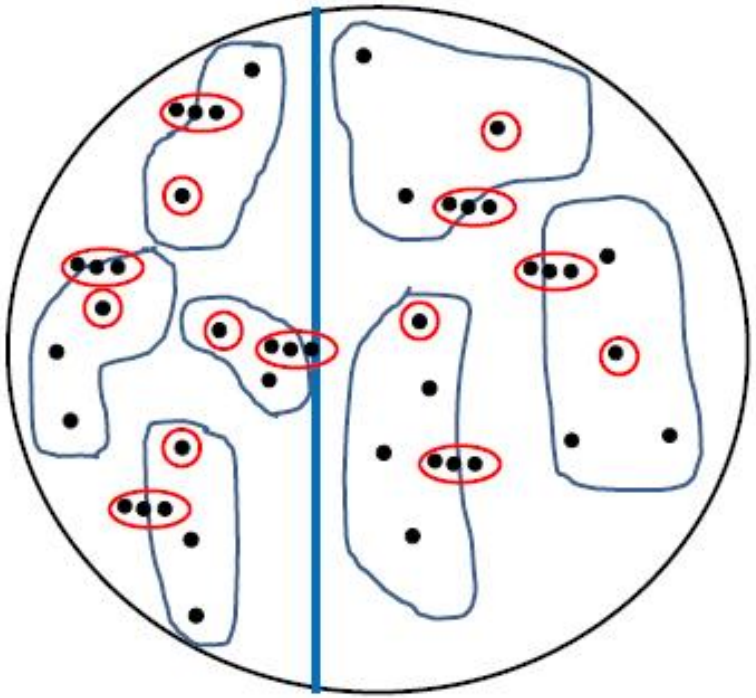
Boundary Value Analysis

- Choose the values that cause the program to fail
- Errors occur at boundary rather in center
- Test boundary values
  - Right on a boundary
  - Close to a boundary on either side
- For each boundary, we test +/-1 in the least significant digit of either side of the boundary.
- Boundary values are calculated as LB,LB-1, UB, UB+1



Boundary Value Analysis

- Create test cases to test boundaries of equivalence classes



Boundary Value Analysis Example 1

- **Assume, we have to test a field which accepts Age 18 – 56**

AGE [Enter Age] *Accepts value 18 to 56

BOUNDARY VALUE ANALYSIS		
Invalid (min -1)	**Valid (min, +min, -max, max)**	**Invalid (max +1)**
17	18, 19, 55, 56	57

- Minimum boundary value is 18
- Maximum boundary value is 56
- Valid Inputs: 18,19,55,56
- Invalid Inputs: 17 and 57

Test Cases for Example 1

- **Test cases for input box accepting numbers between 18 and 56 using Boundary value analysis:**
  - **- Test cases with test data exactly as the input boundaries of input**
  - **domain i.e. values 18 and 56 in our case.**
  - **- Test data with values just below the extreme edges of input domains**
  - **i.e. values 17 and 55.**
  - **- Test data with values just above the extreme edges of the input**
  - **domain i.e. values 19 and 57**

Test Cases for Example 1

- Test case 1: Enter the value 17 (18-1) = Invalid
- Test case 2: Enter the value 18 = Valid
- Test case 3: Enter the value 19 (18+1) = Valid
- Test case 4: Enter the value 55 (56-1) = Valid
- Test case 5: Enter the value 56 = Valid
- Test case 6: Enter the value 57 (56+1) =Invalid

Boundary Value Analysis Example 2

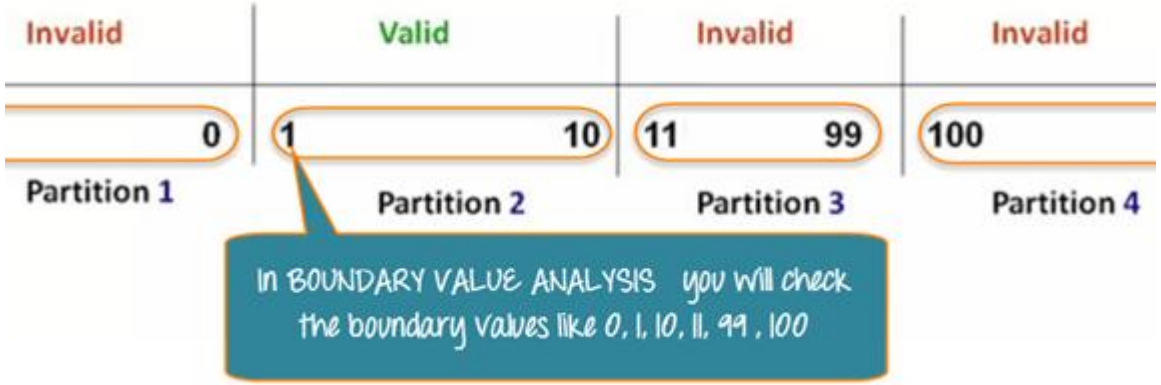
Pizza order application

- **Requirements:**
  - **Pizza values 1 to 10 is considered valid. A success message is shown.**
  - **While value 11 to 99 are considered invalid for order and an error message will appear, "Only 10 Pizza can be ordered“**

- **Test Conditions and Partitions:**
  - **Any Number greater than 10 entered in the Order Pizza field(let say 11) is considered invalid.**
  - **Any Number less than 1 that is 0 or below, then it is considered invalid.**
  - **Numbers 1 to 10 are considered valid**
  - **Any 3 Digit Number say 100 is invalid.**

Boundary Value Analysis Example 2

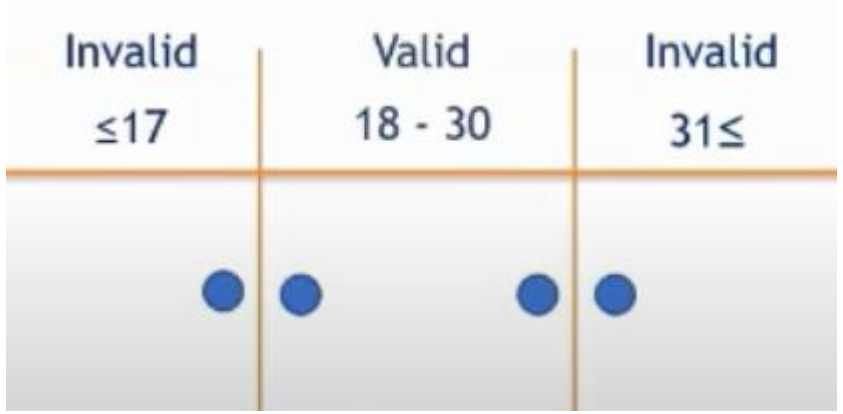
- In BVA,we test boundaries between equivalence partitions



Boundary Value Analysis Example 3

- A text field in an application accepts input as the age of the user. Here, the values allowed to be accepted by the field is between 18to 30 years, inclusive of both the values. By applying Boundary value analysis what is the minimum number of test cases required for maximum coverage.

- 2
- 3
- 1
- 4 Ans



Advantages and Disadvantages of BVA

Advantages:

- Density of defects at boundaries is more
- Overall execution time reduces
- Best for physical quantities

Disadvantages:

- Incorrect EC leads to incorrect BVA
- Not suitable for input value depends on the decision of another value
- Not fit for Boolean and logical

Questions and Answers


Thank
you