Equivalence Class Testing

Lecture Objectives:

1) Explain the concept of design by contract and test by contract.
2) Define equivalence class.
3) Compare and contrast defensive testing and testing by contract. (Text Reading Only)
4) Given a software description, define the equivalence classes for a given problem.
5) Given a software description, construct test cases using the equivalence partitioning method.
6) Based on Equivalence class testing, determine the minimum number of test cases necessary to test a given software system.

Homework

- See passed out sheet.

Design by contract, defensive testing, and testing by contract

Design by contract:

There is a specific interface that defines a contract and your implementation must meet it.

→ Testing based on valid numbers

ADD

⇒ Add accepts two integers A and B that are positive.

⇒ Add returns the sum of A+B.

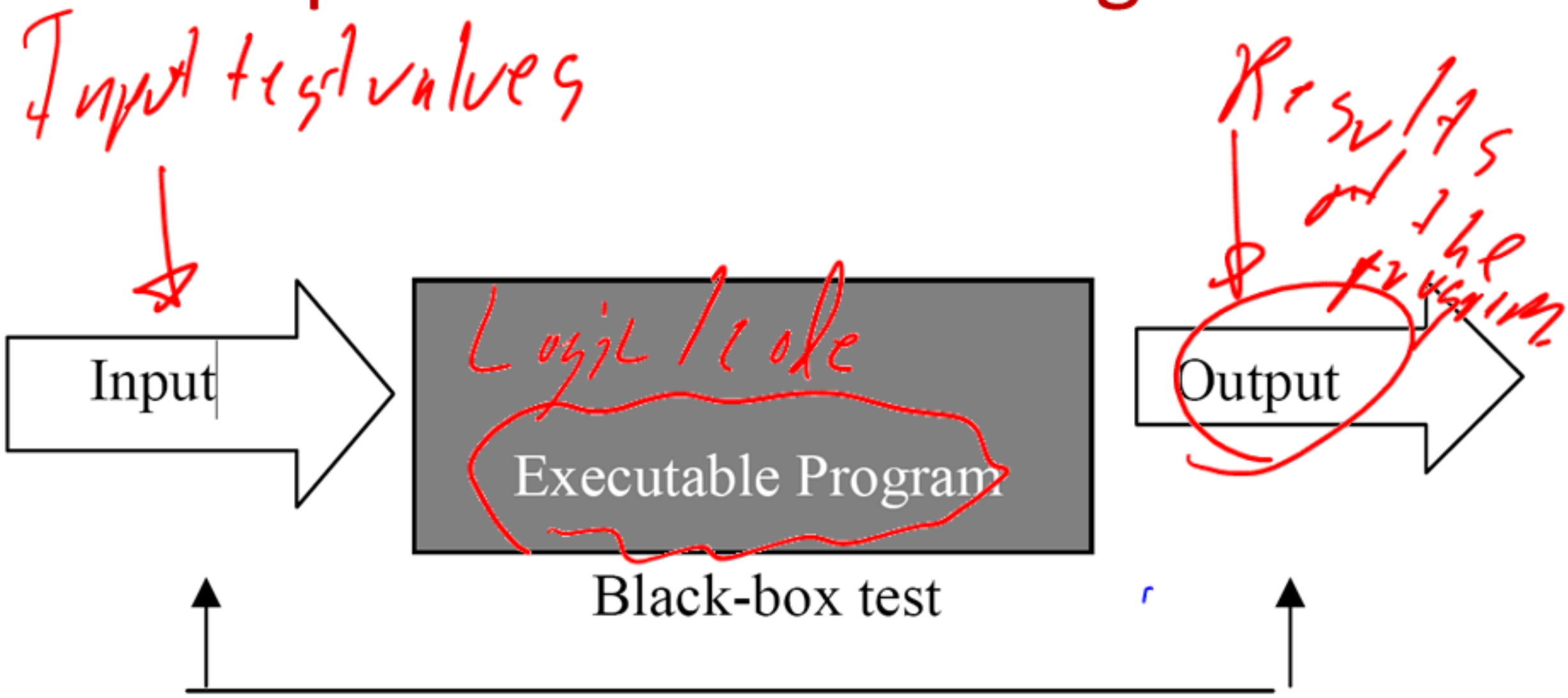
Design by contract, defensive testing, and testing by contract

Testing both valid and invalid values based on the contract.

Main Types of Black Box Tests

- Functional Testing
- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table Testing

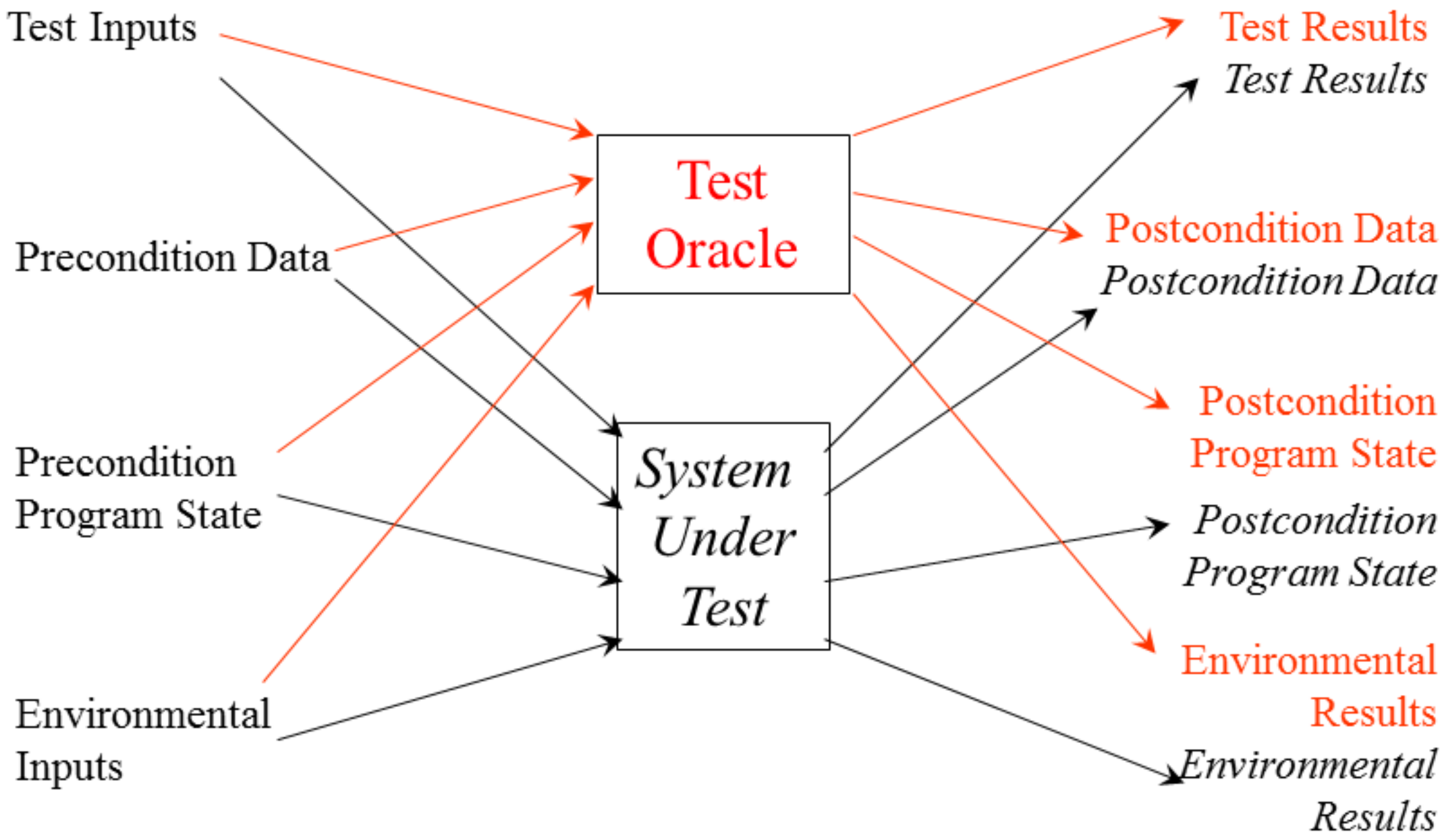
Simple Black Box Testing Model



Test Oracle

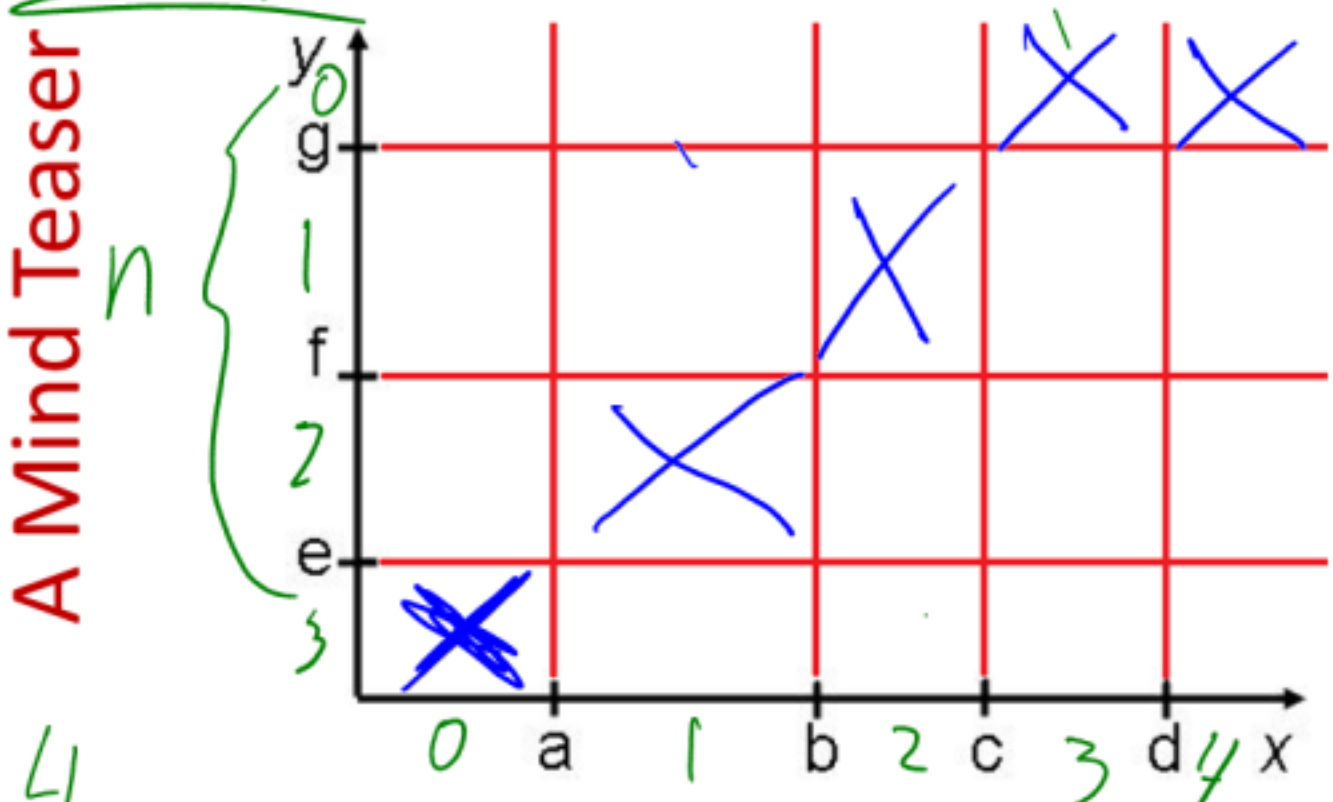
- An oracle is a mechanism for determining whether the program has passed or failed a test.
- A complete oracle would have three capabilities and would carry them out perfectly:
  - A *generator*, to provide predicted or expected results for each test.
  - A *comparator*, to compare predicted and obtained results.
  - An *evaluator*, to determine whether the comparison results are sufficiently close to be a pass.

The "Complete" Oracle



Reprinted with permission of Doug Hoffman

- Given a grid of size m x n, how many dots would be required to ensure that each row and each column contains at least one dot?



$Max(m, n)$

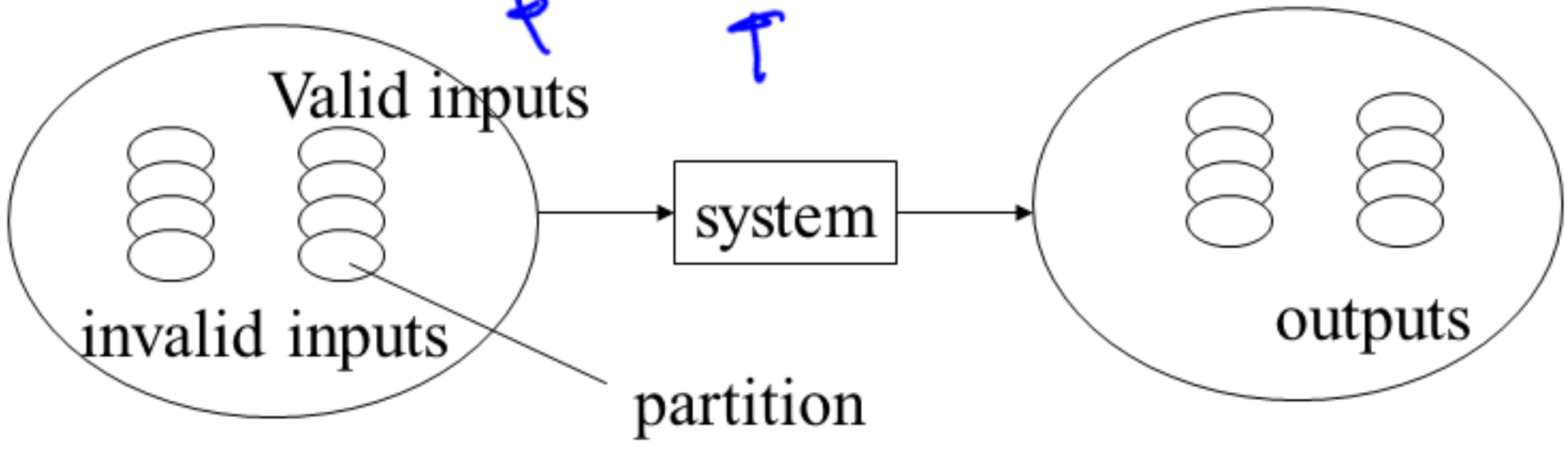
Equivalence Partitioning

Equivalence partitioning is a black-box testing method
- divide the input domain of a program into classes of data
- derive test cases based on these partitions.

Test case design for equivalence partitioning is based on an evaluation of equivalence classes for an input domain.

An equivalence class represents a set of valid or invalid states for input condition.

An input condition is:
- a specific numeric value, a range of values
- a set of related values, or a Boolean condition



Problem

- Lets assume we have a method, doSomething, which has two parameters.

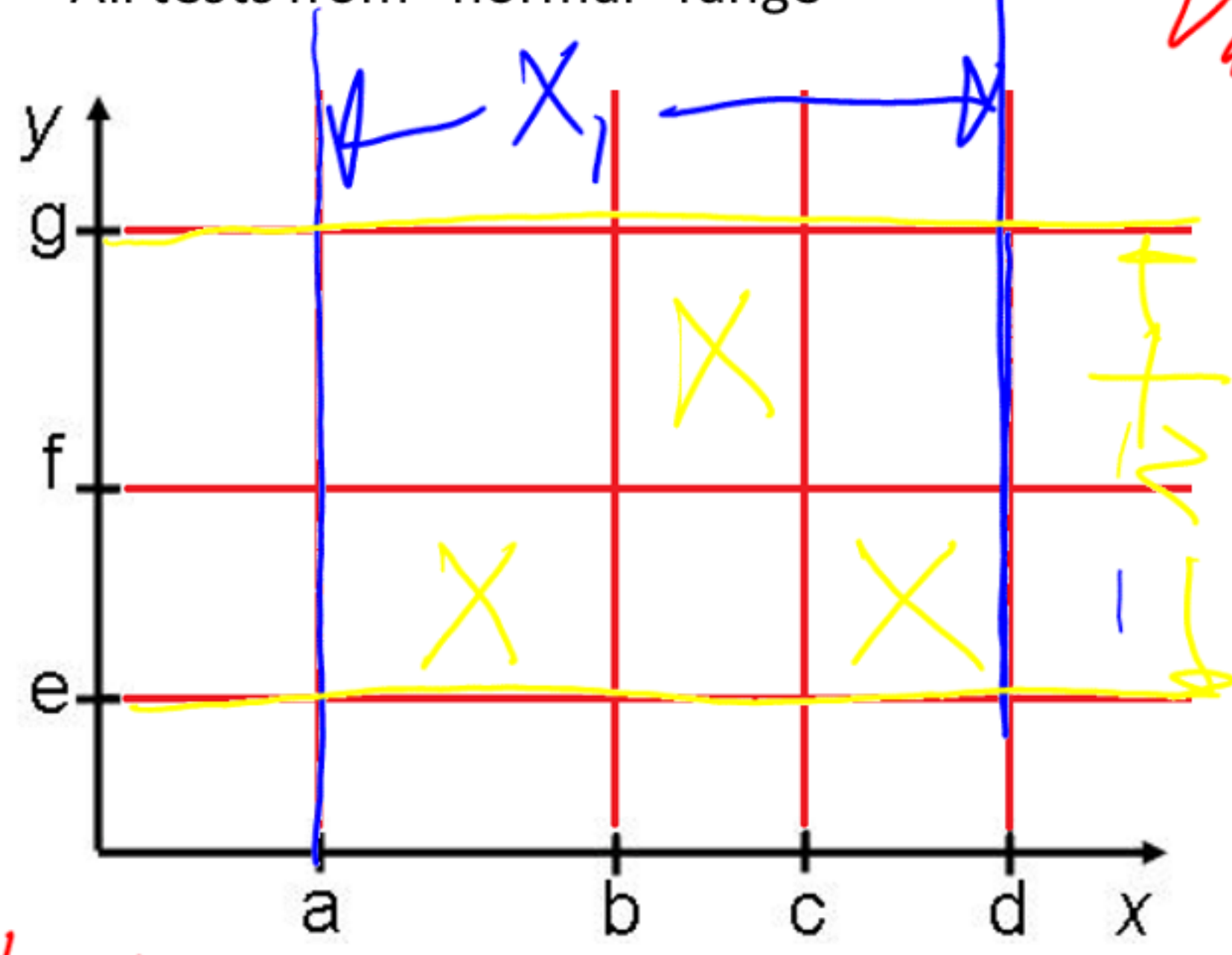
- Int doSomething(int x1, int x2);

$$a \le x1 \le d, \text{intervals}[a, b), [b, c), [c, d]$$

$$e \le x2 \le g, \text{intervals}[e, f), [f, g]$$

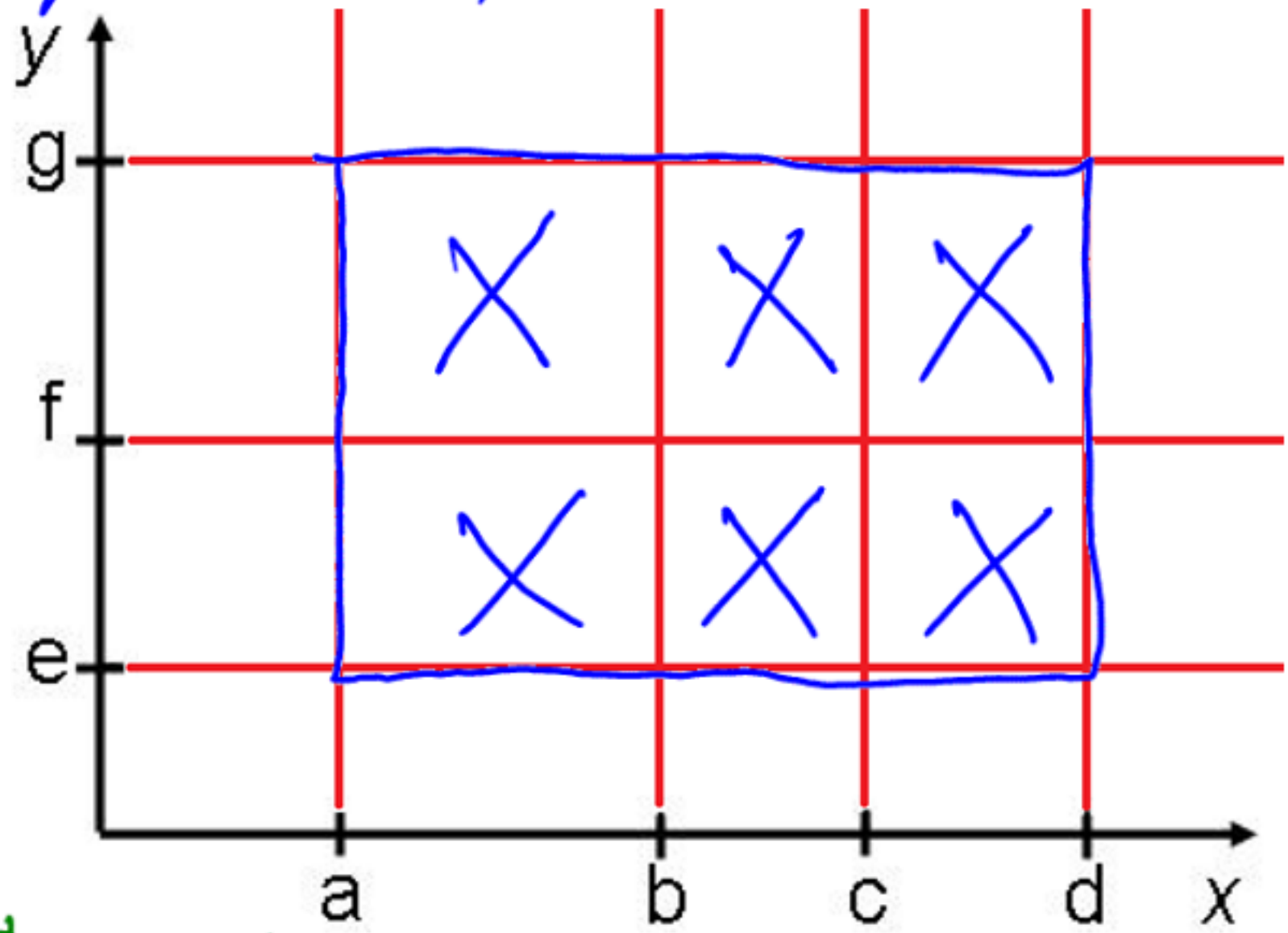
Weak normal equivalence classes

- Select a set of tests such that each equivalence class is tested at least once.
  - All tests from "normal" range



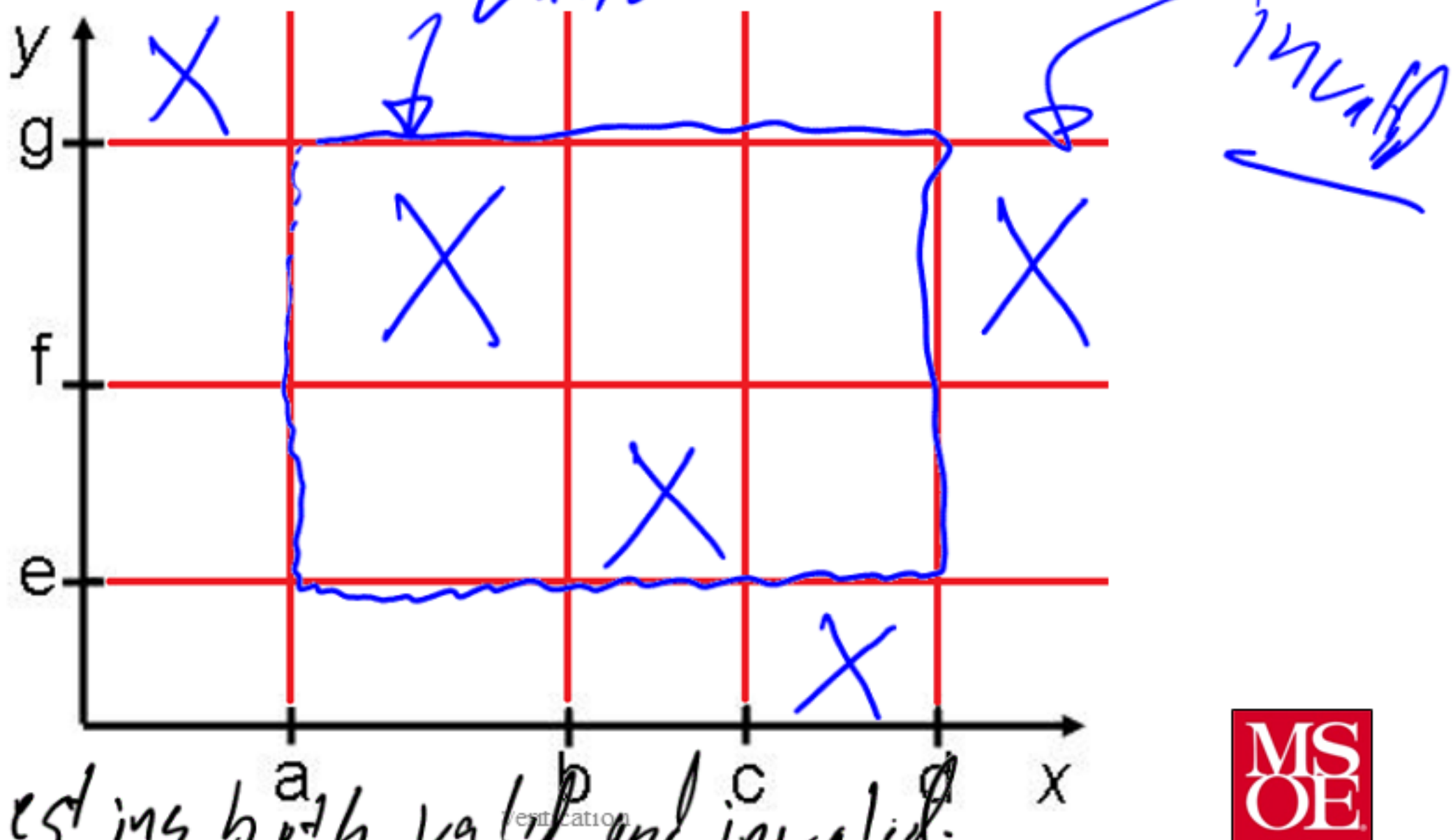
Strong Normal Equivalence Class Testing

- Select a set of tests such that each set of equivalence classes is tested at least once.
  - All tests from the "normal" range.



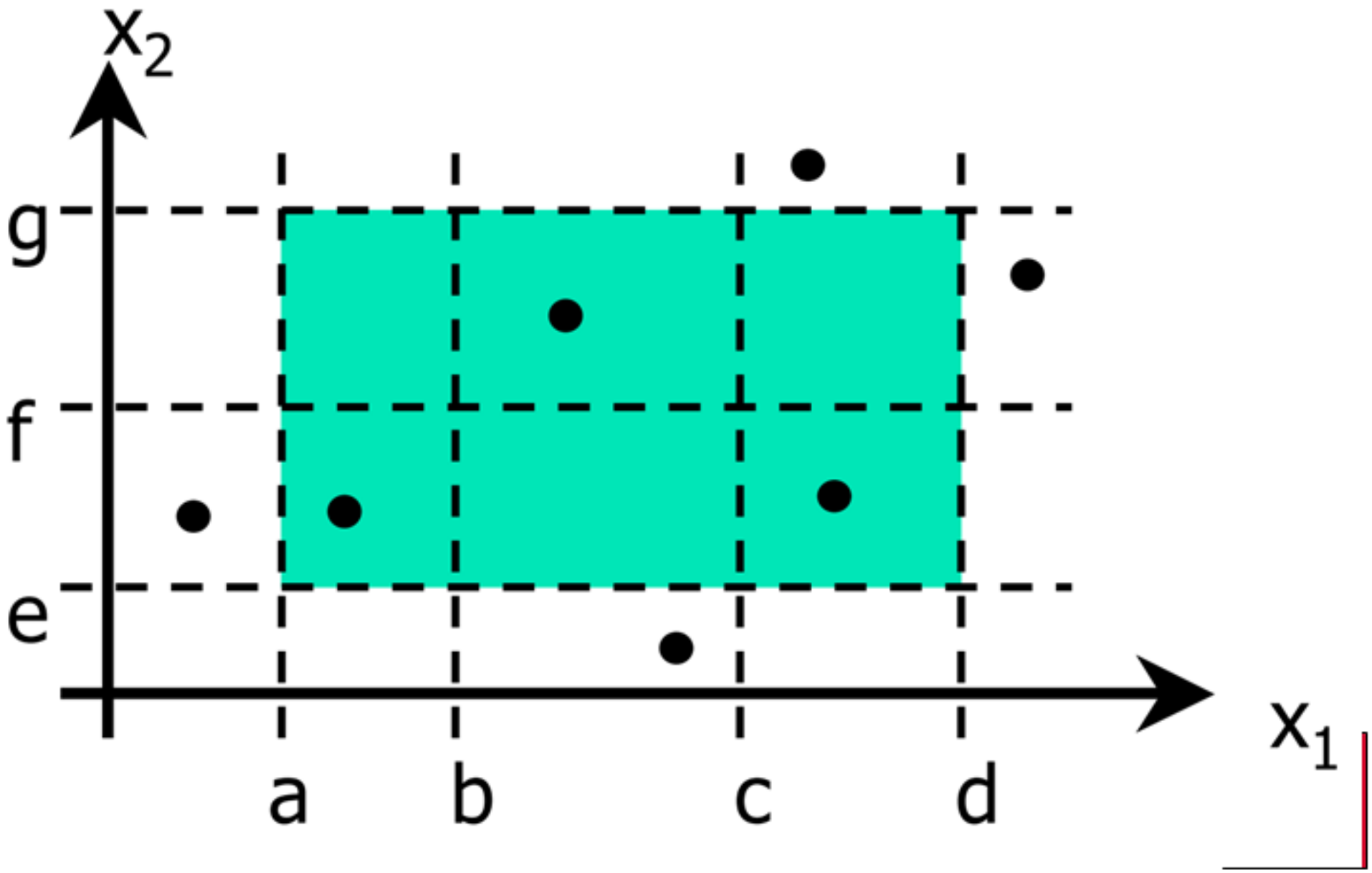
Weak Robust Equivalence Class Testing

- Valid inputs
  - Use 1 value from each class
- Invalid inputs
  - Test case will have one invalid input and all others will be valid



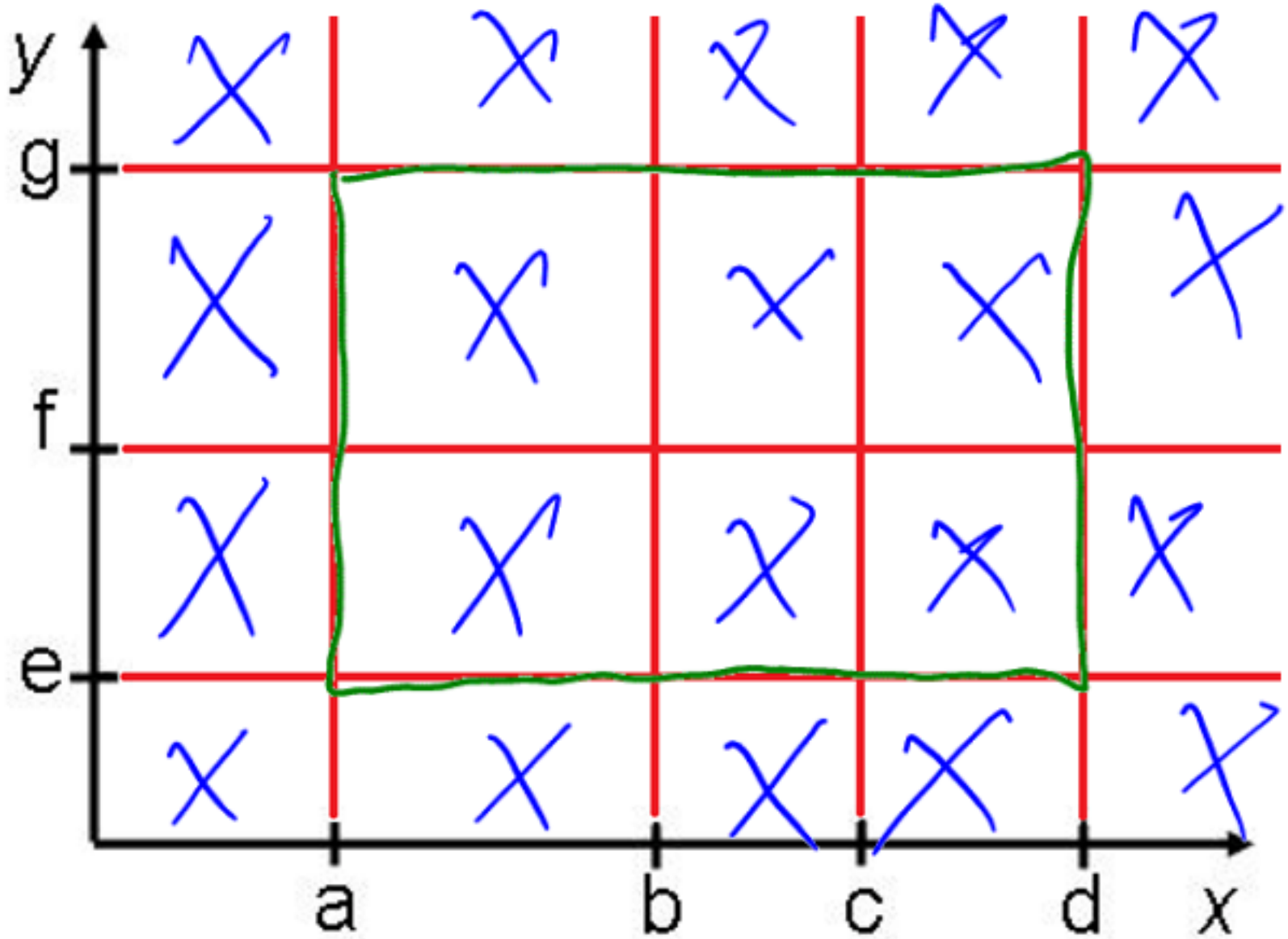
Weak Robust Equivalence Class Testing

- Valid inputs
  - Use 1 value from each class
- Invalid inputs
  - Test case will have one invalid input and all others will be valid



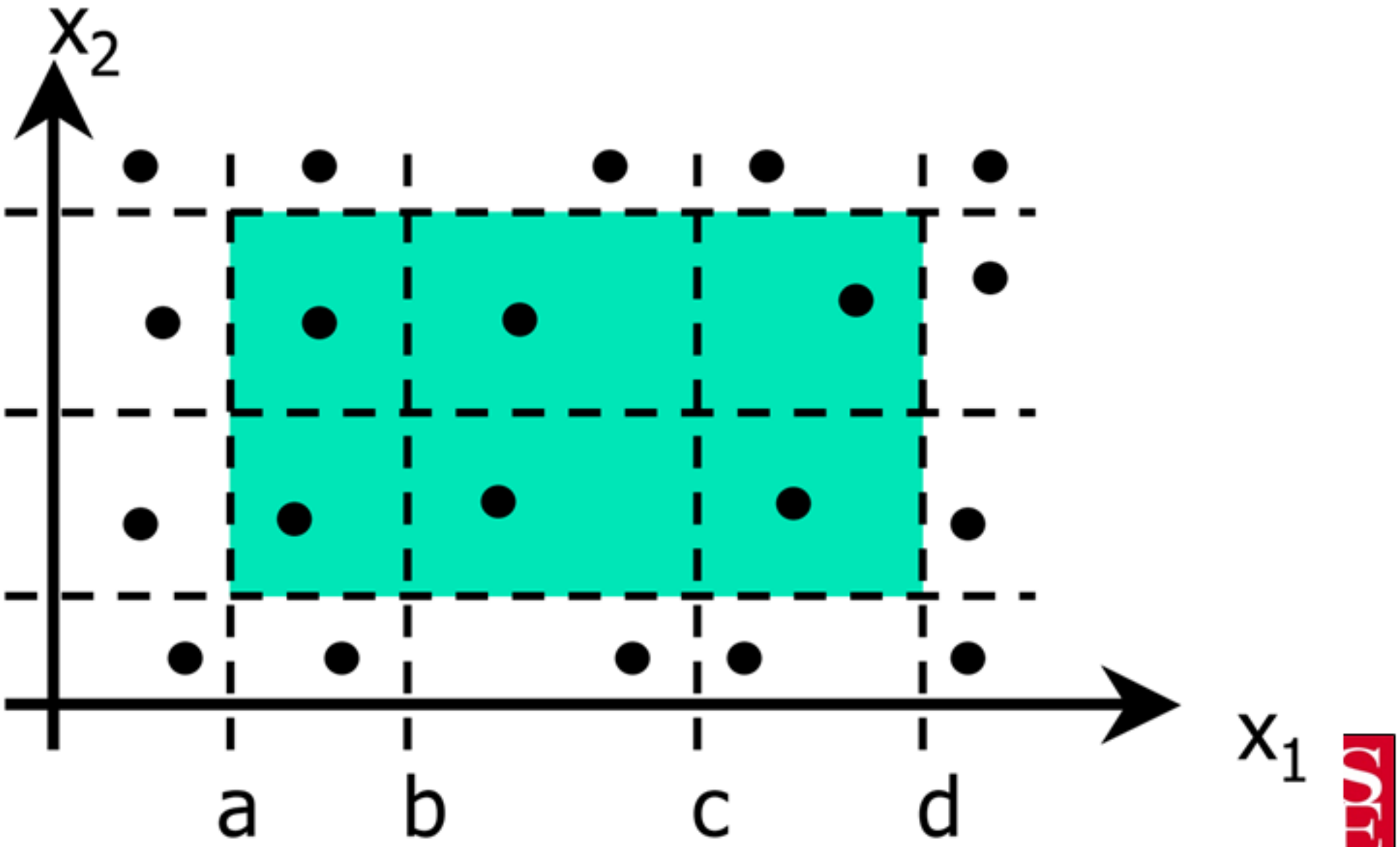
Strong Robust Equivalence Class Testing

- Valid inputs
  - Use value from each class intersection
- Invalid inputs
  - Test case will have one invalid input and all others will be valid



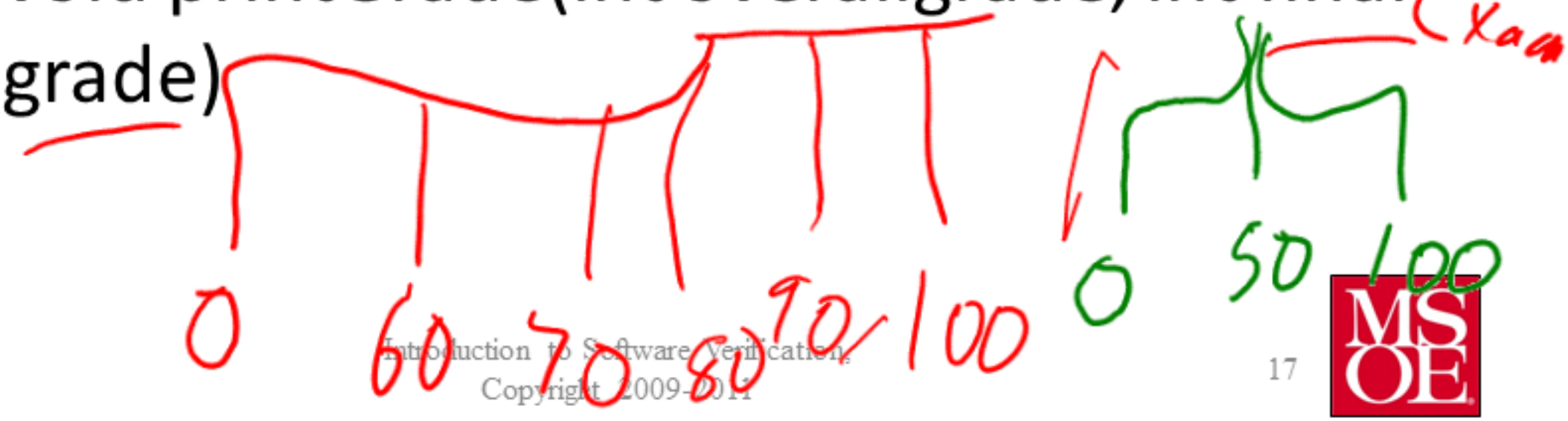
Strong Robust Equivalence Class Testing

- Valid inputs
  - Use value from each class intersection
- Invalid inputs
  - Test case will have one invalid input and all others will be valid

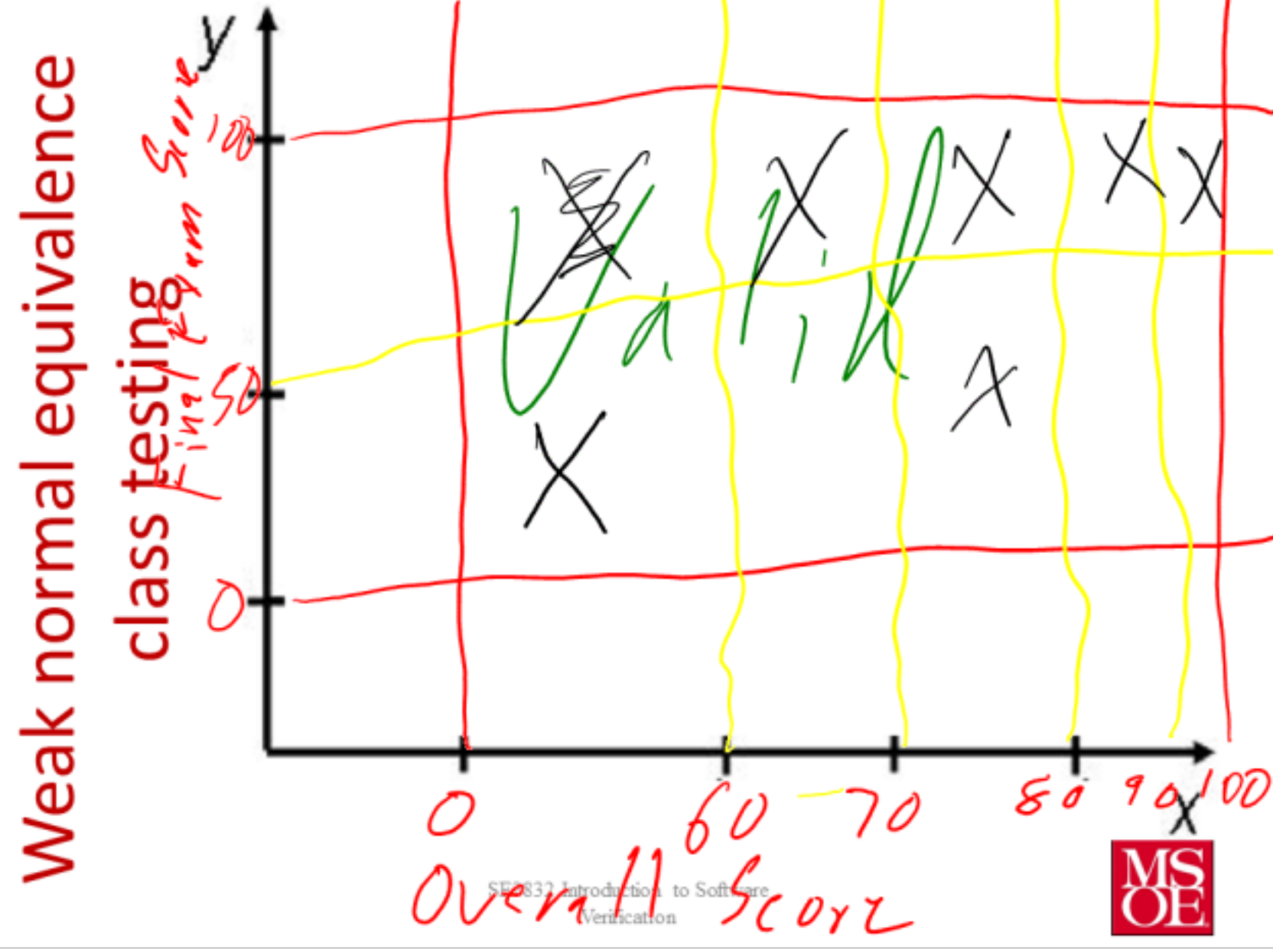


Equivalence Class Example

```
/**
This method will print a letter grade of
A(90), B(80), C(70), D(60), or F(<60) out to
the screen or throw an exception if an
invalid grade (<0, >100) is entered.  If the
final grade is less than 50, F will be
printed (assuming the value is in range.)
**/
void printGrade(int overallgrade, int final
grade)
```



Weak normal equivalence class testing



(40, 40, F)

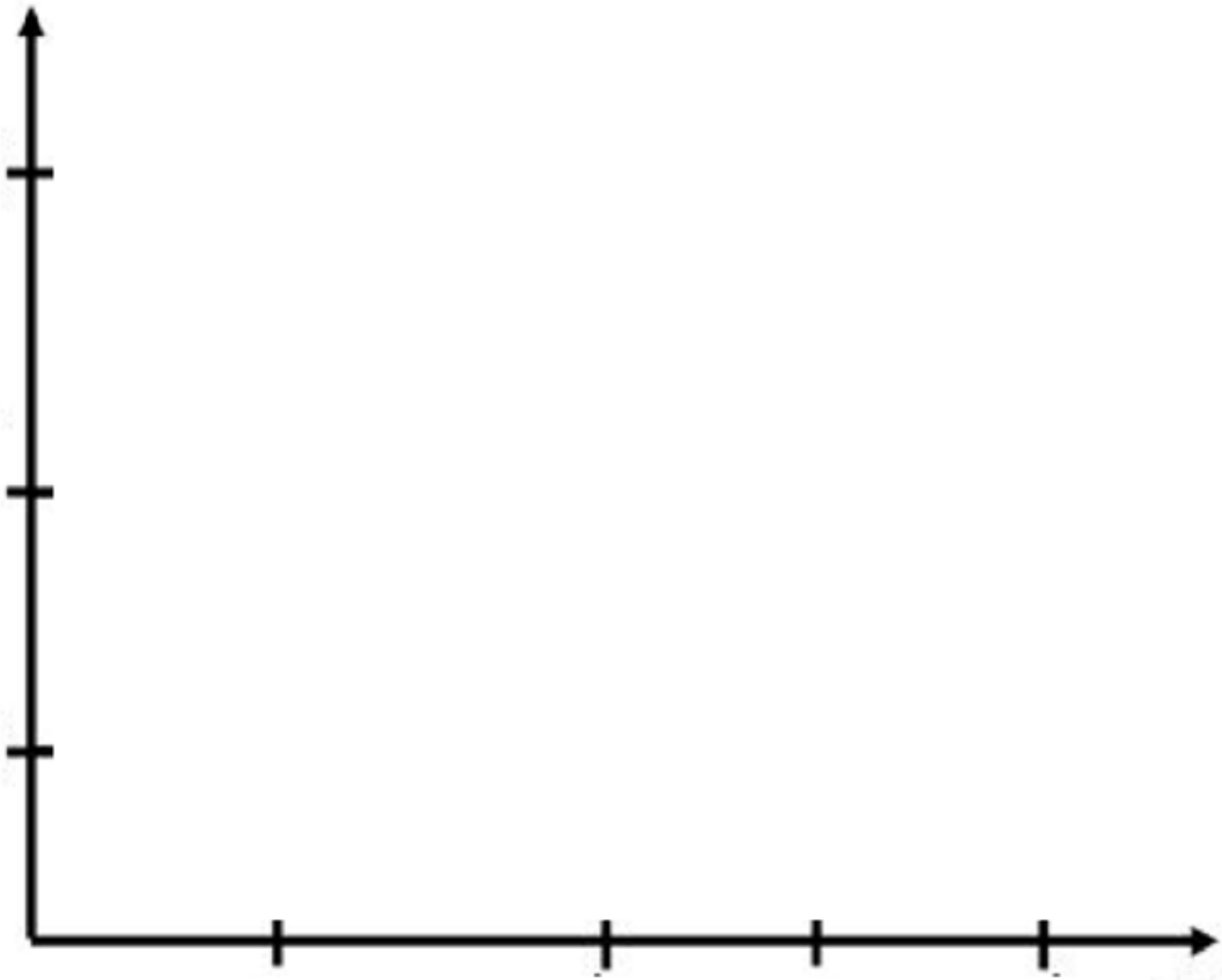
(65, 60, D)

(75, 60, C)

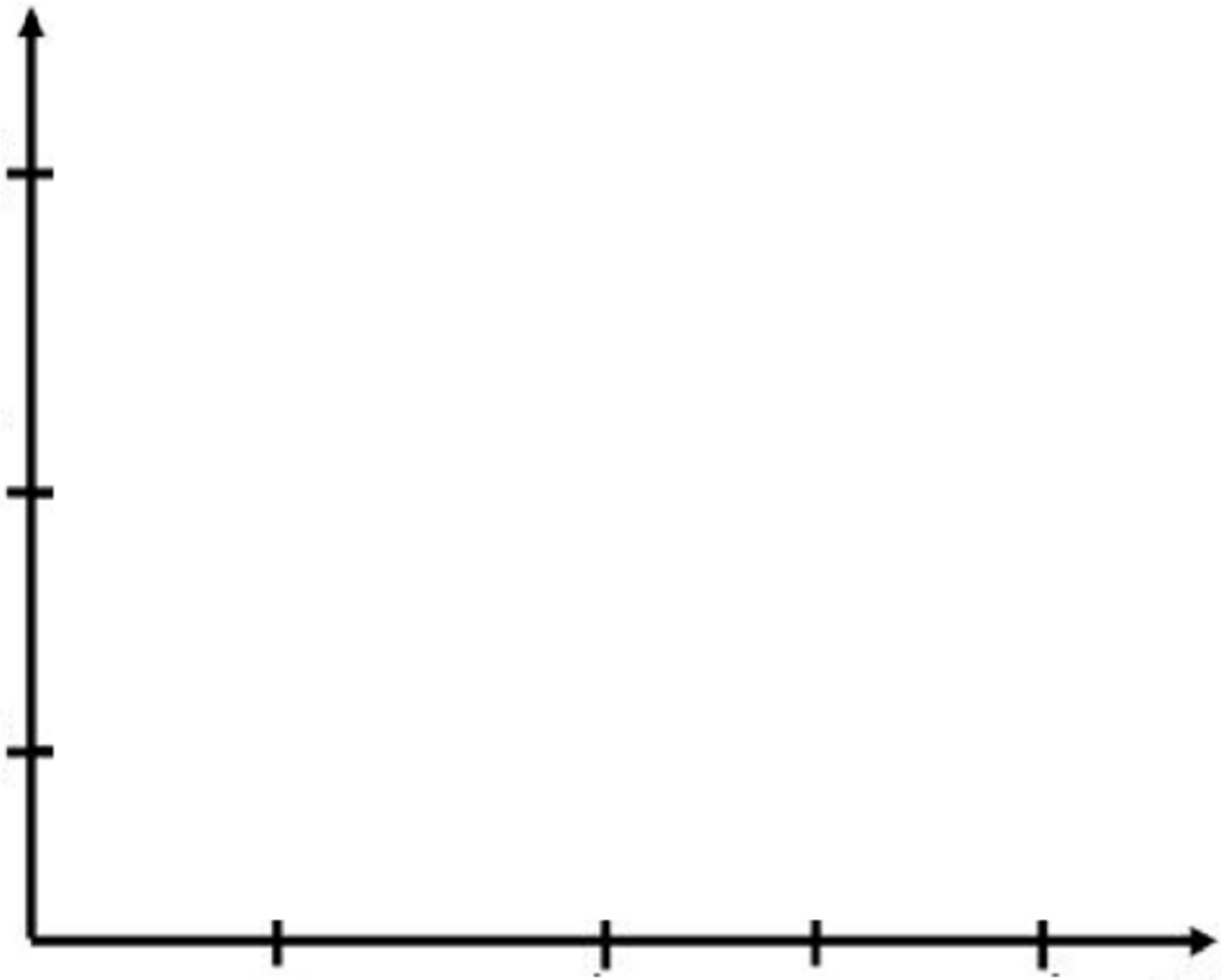
(85, 60, B)

(95, 60, A)

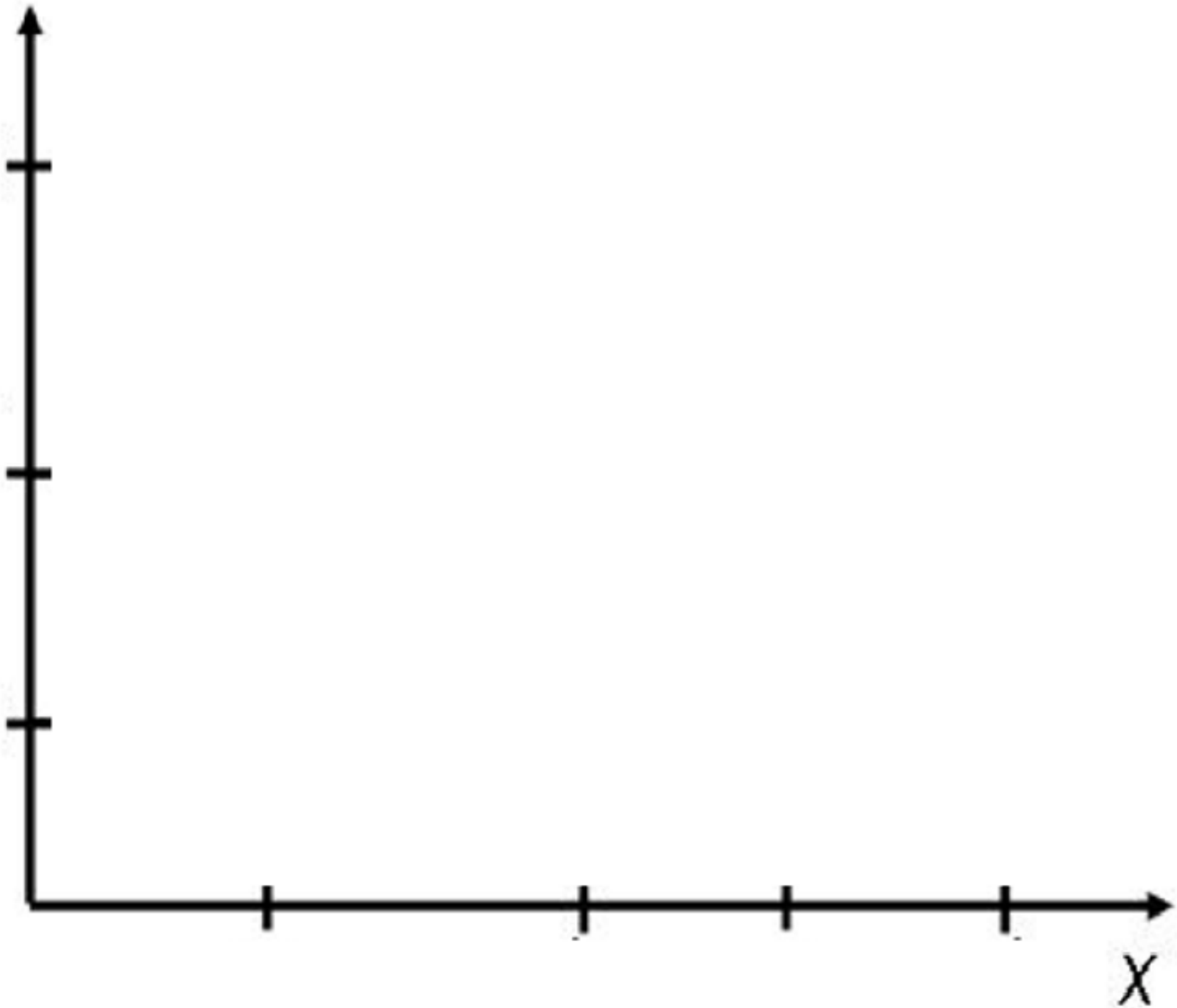
Strong normal equivalence class testing



Weak robust equivalence class testing



Strong robust equivalence class testing



case counts

- Weak normal
  - Max (N,M)
- Strong Normal
  - M * N
- Weak Robust
  - Max(N,M) + 4

General test

- Strong robust
  - (M+2)*(N+2)