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. (T[redacted]y)
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.

Design by contract, defensive testing, and testing by contract

- Here's a one page handout. Read it through with your partner and be able to explain design by contract, defensive testing, and testing by contract.

Modules are defined by pre/post conditions.

→ Verify that module works based on what it is supposed to, not actual

Defensive Design:

⇒ Design your module to be able to handle both valid and ~~in~~ invalid entries.

Defensive Testing:

Testing both Normal and abnormal conditions

Main Types of Black Box Tests

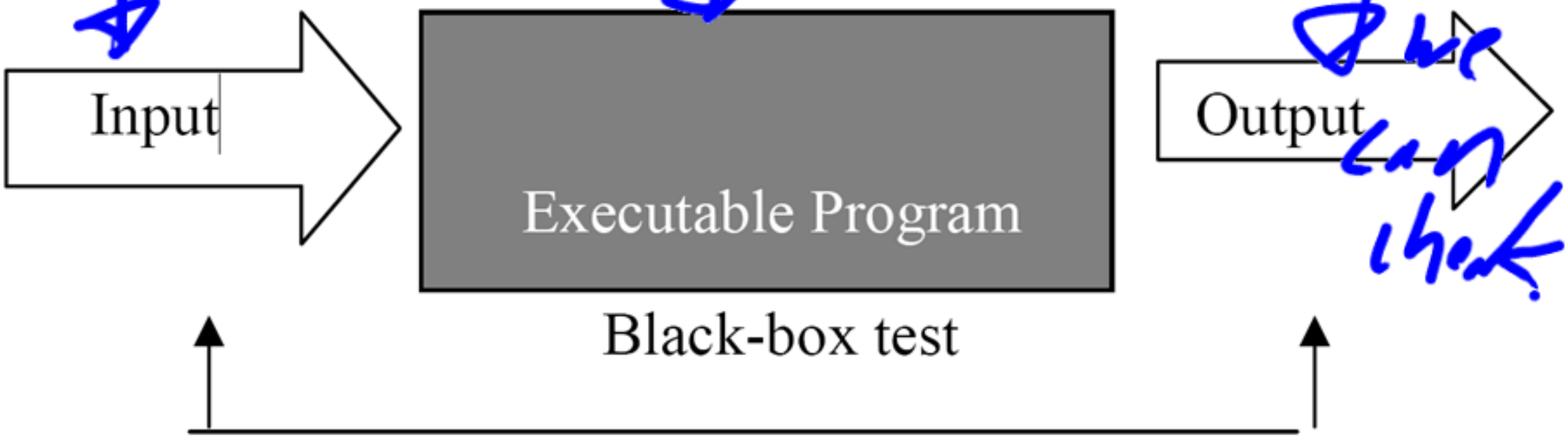
- Functional Testing
  - ↳ Functionality
- Equivalence Partitioning
  - ↳ Breaking our data into groups
- Boundary Value Analysis
  - ↳ Looking for changes in system.
- Decision Table Testing
  - ↳ state machine based

Simple Black Box Testing Model

Values

No knowledge

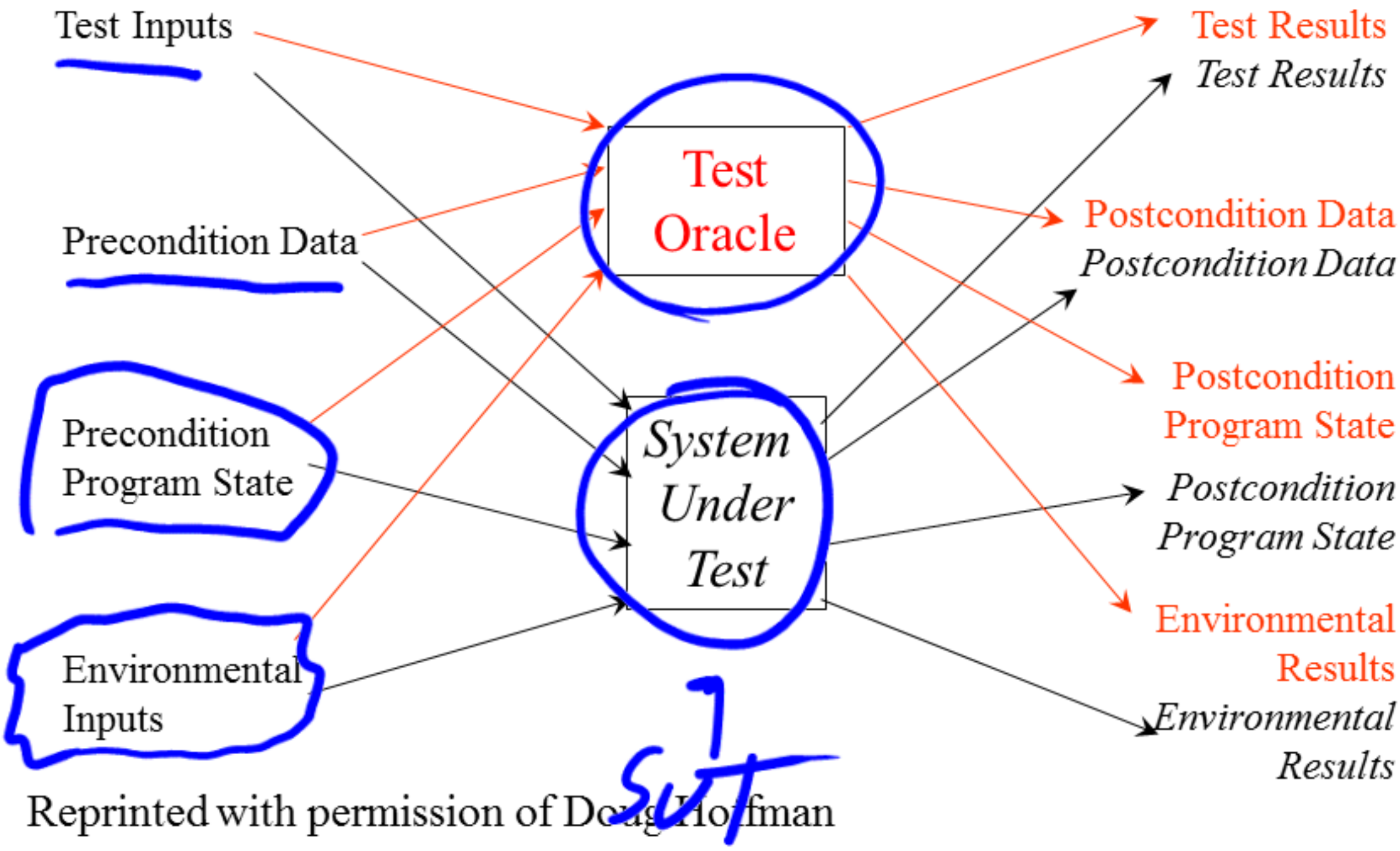
Things that we can check



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



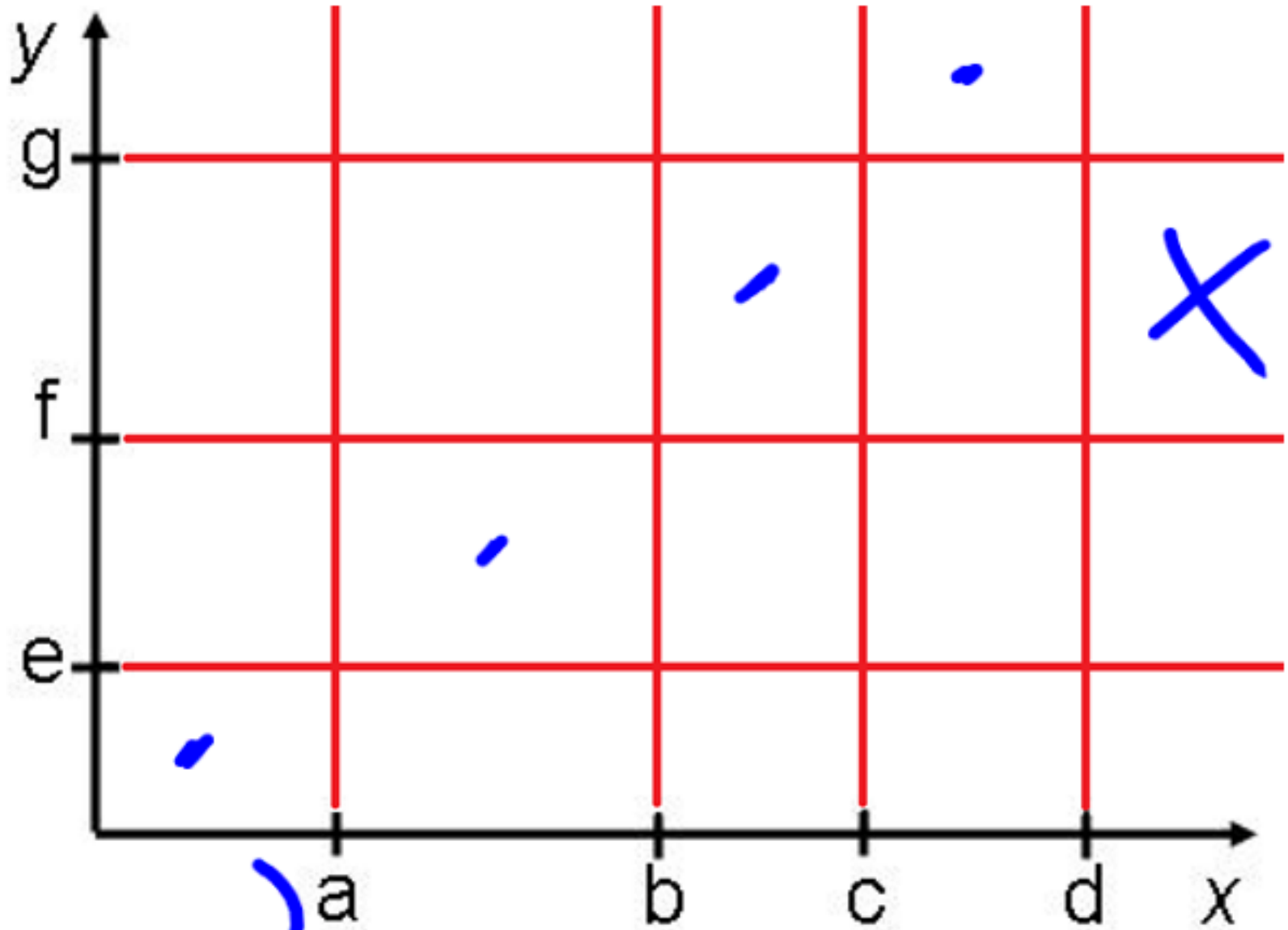
Reprinted with permission of Doug Hoffman

A Mind Teaser

- 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?

4

5 4



Max(m or n)

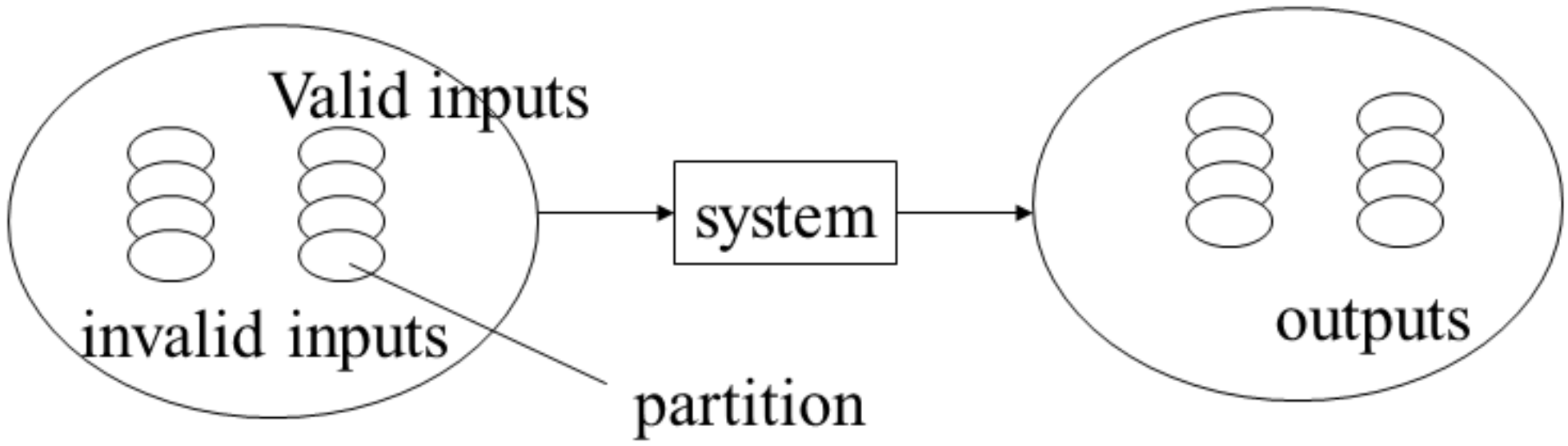
Equivalence Partitioning

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



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

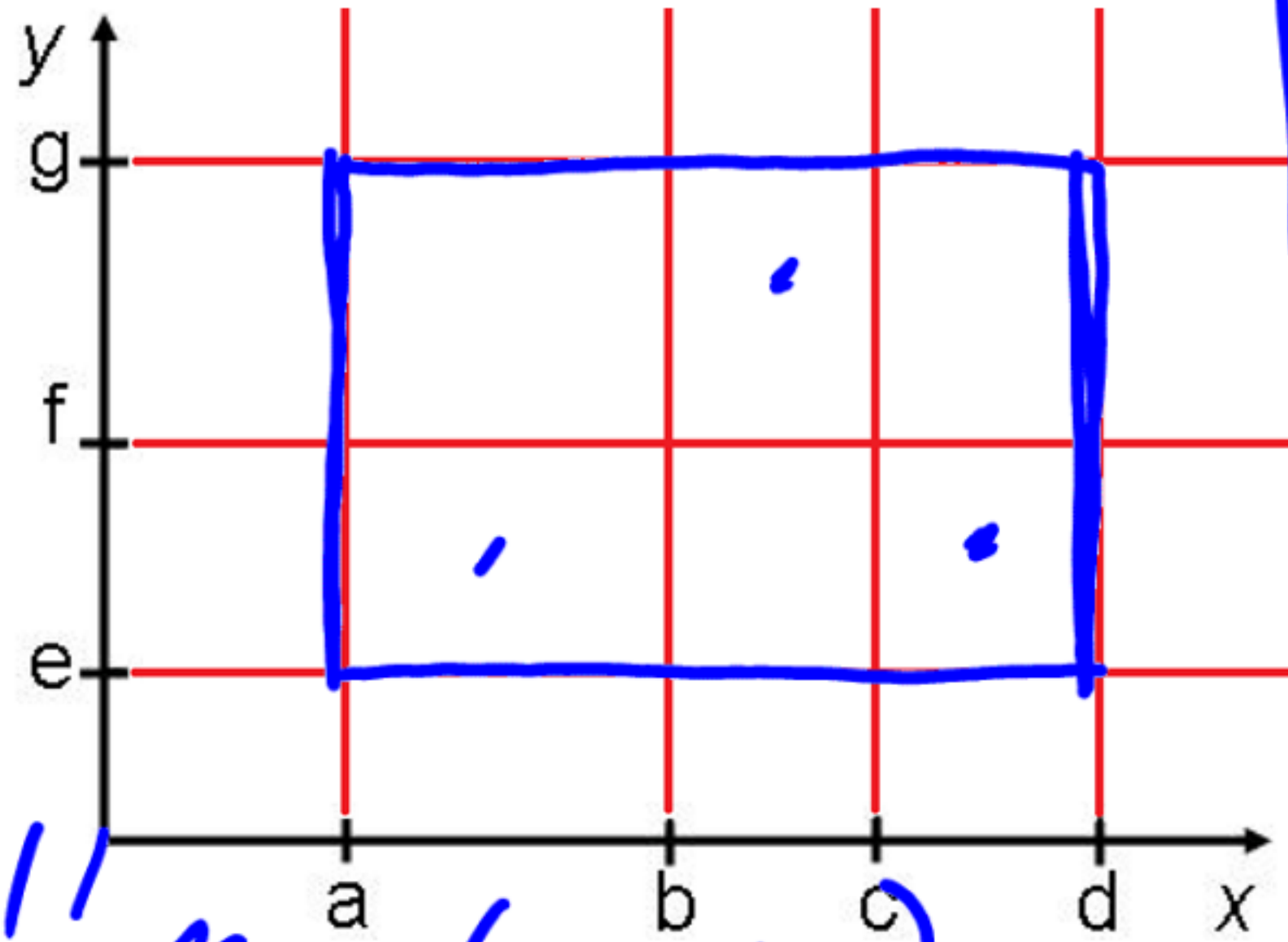
- Int doSomething(int x1, int x2);

Problem

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

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

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



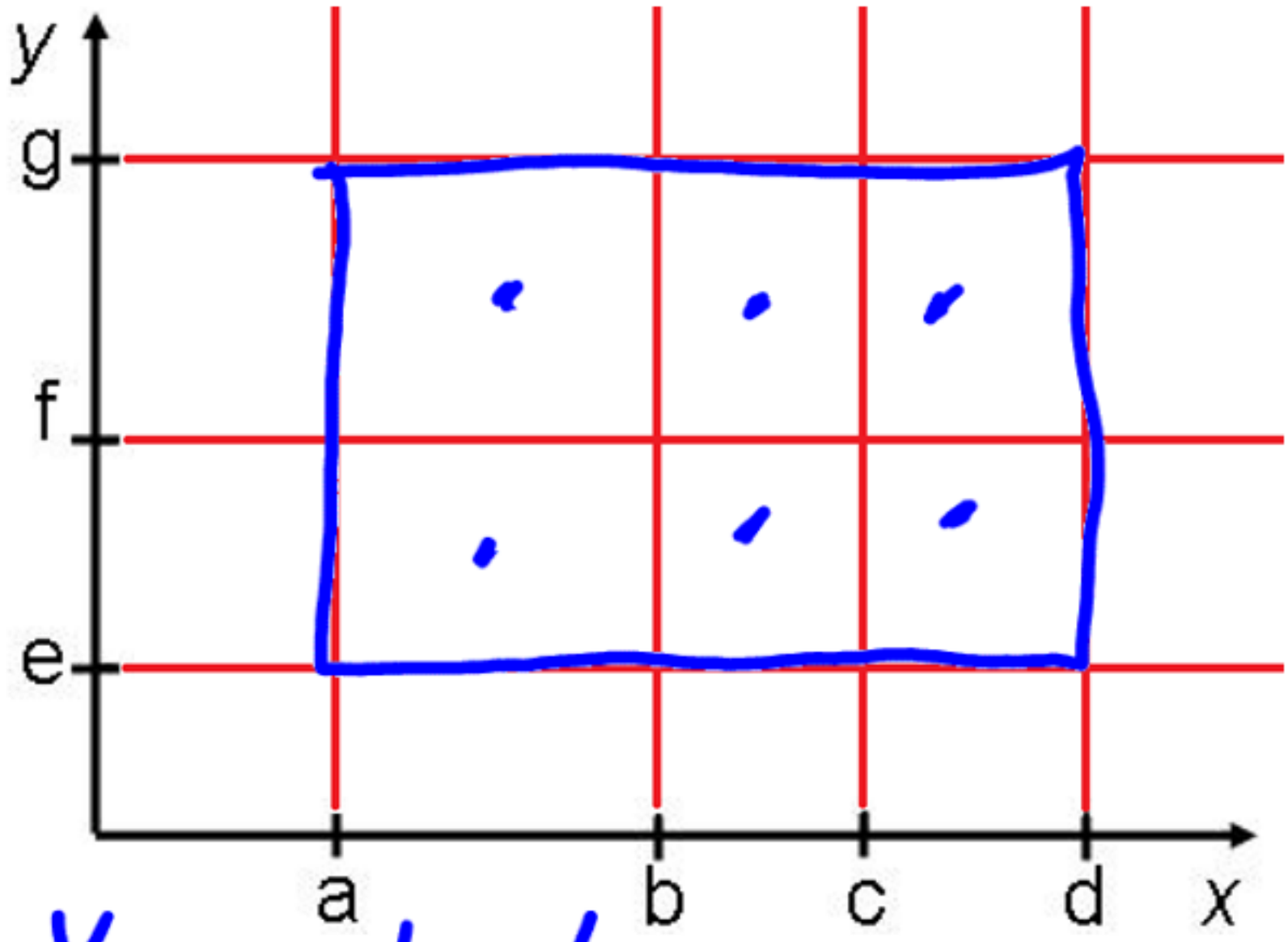
Weak normal equivalence classes Testing with minimal inputs

All tests from "normal" range

Test by contract.

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.

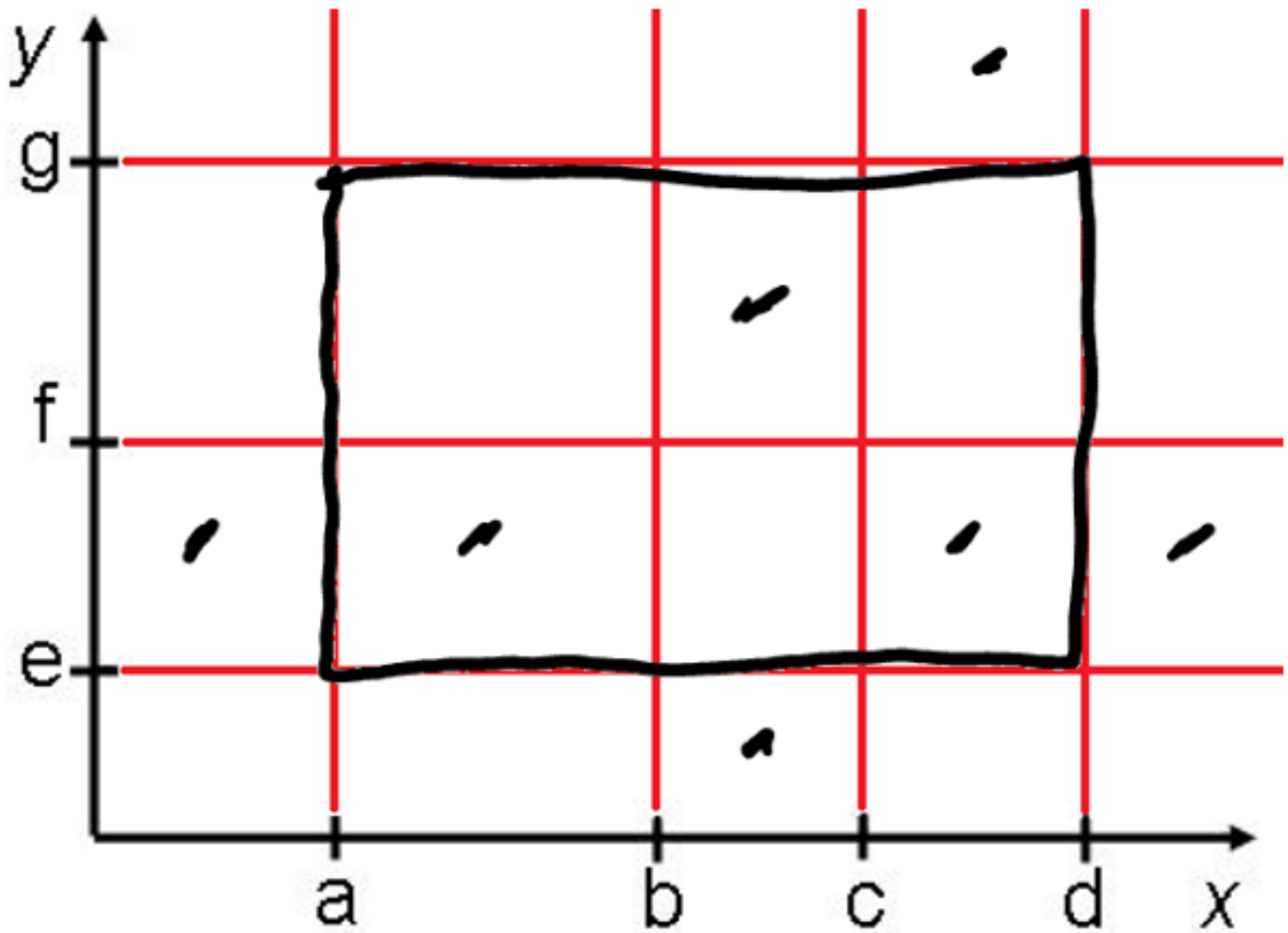


m * n test cases

Weak Robust Equivalence Class Testing

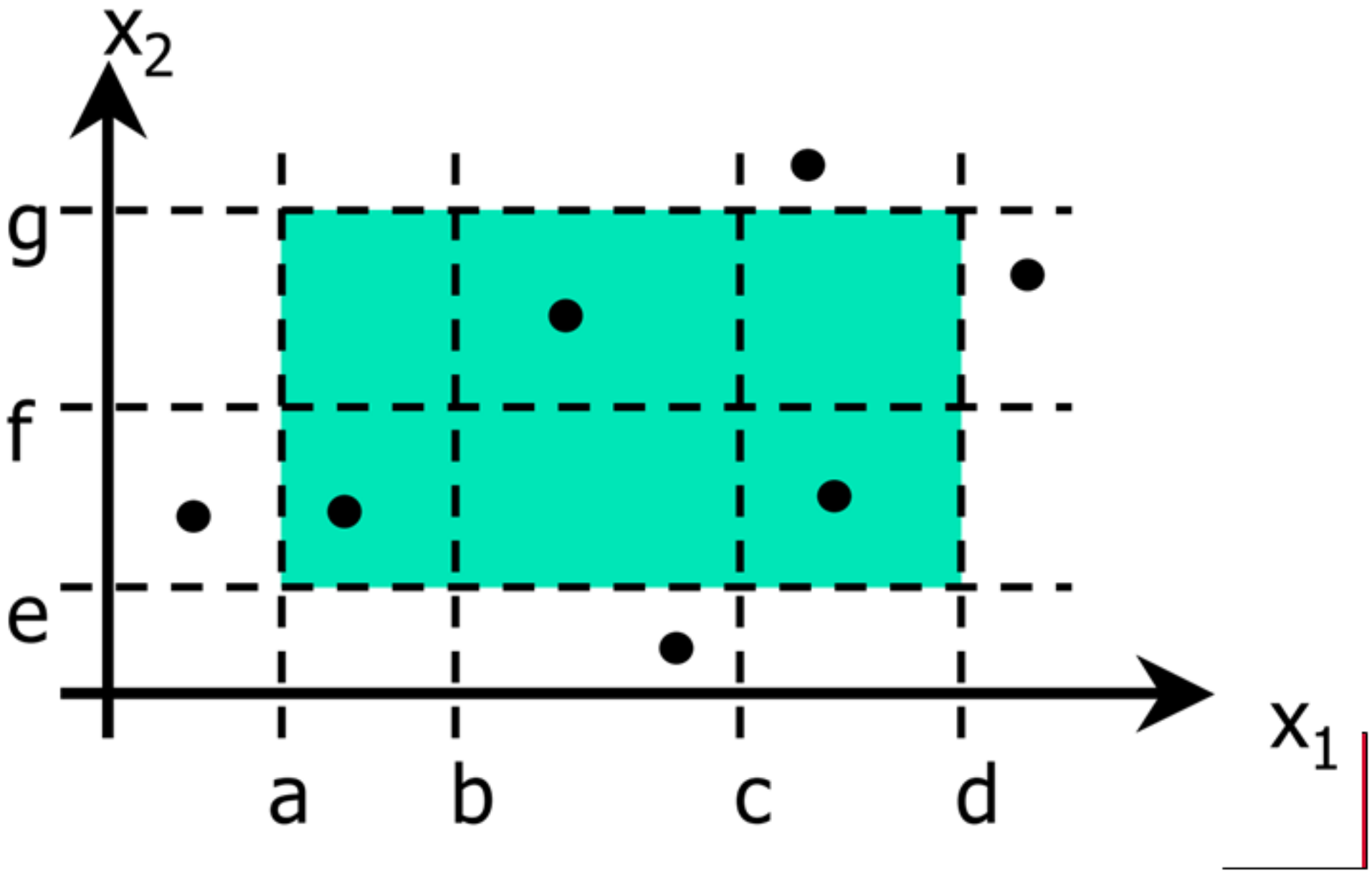
- Valid inputs
  - Use 1 value from each class
- Invalid inputs
  - Test case will have one invalid input and all others will be valid

Defensive testing



Weak Robust Equivalence Class Testing

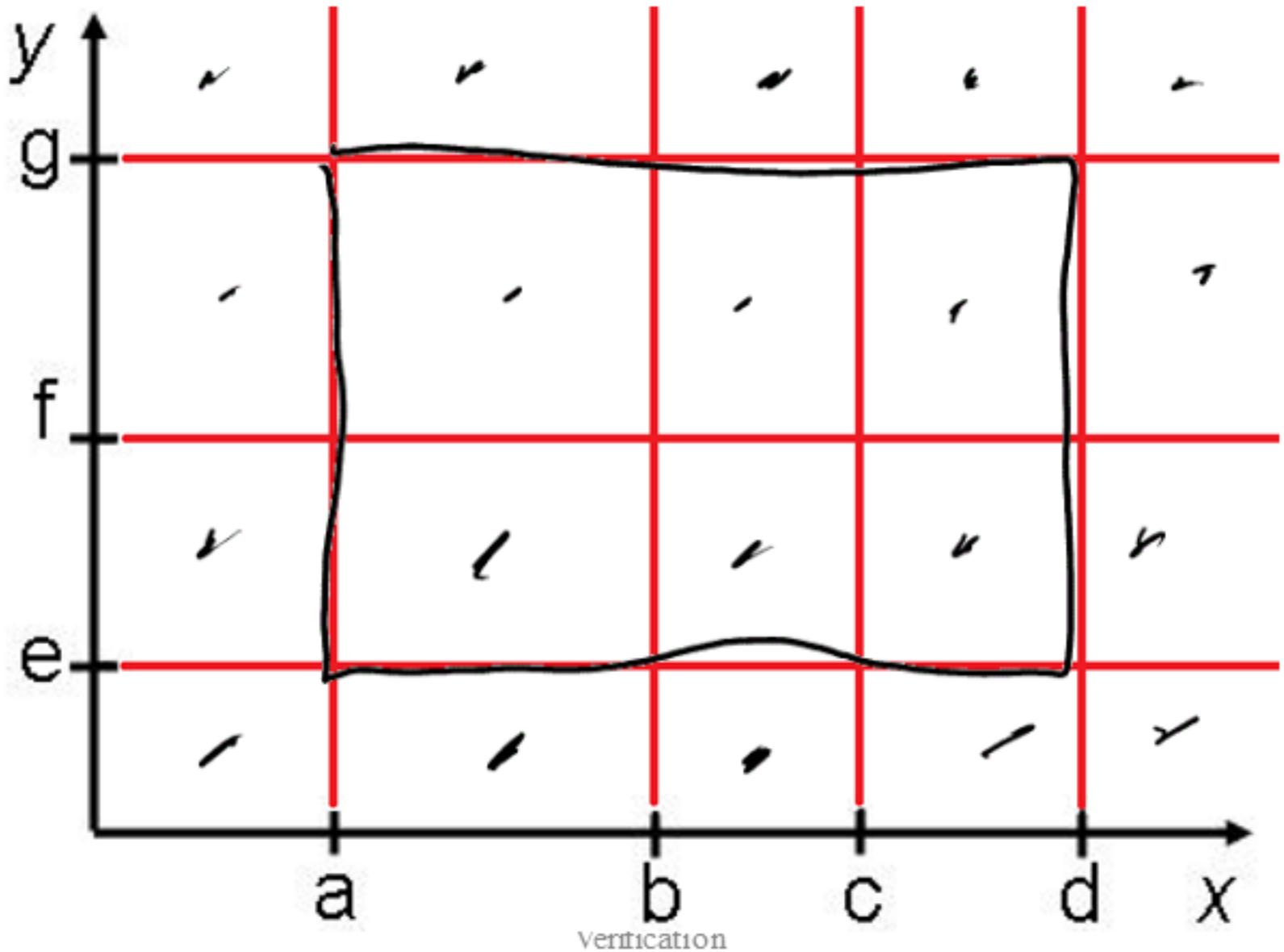
- Valid inputs
  - Use 1 value from each class
- Invalid inputs
  - Test case will have one invalid input and all others will be valid



Strong Robust Equivalence Class Testing

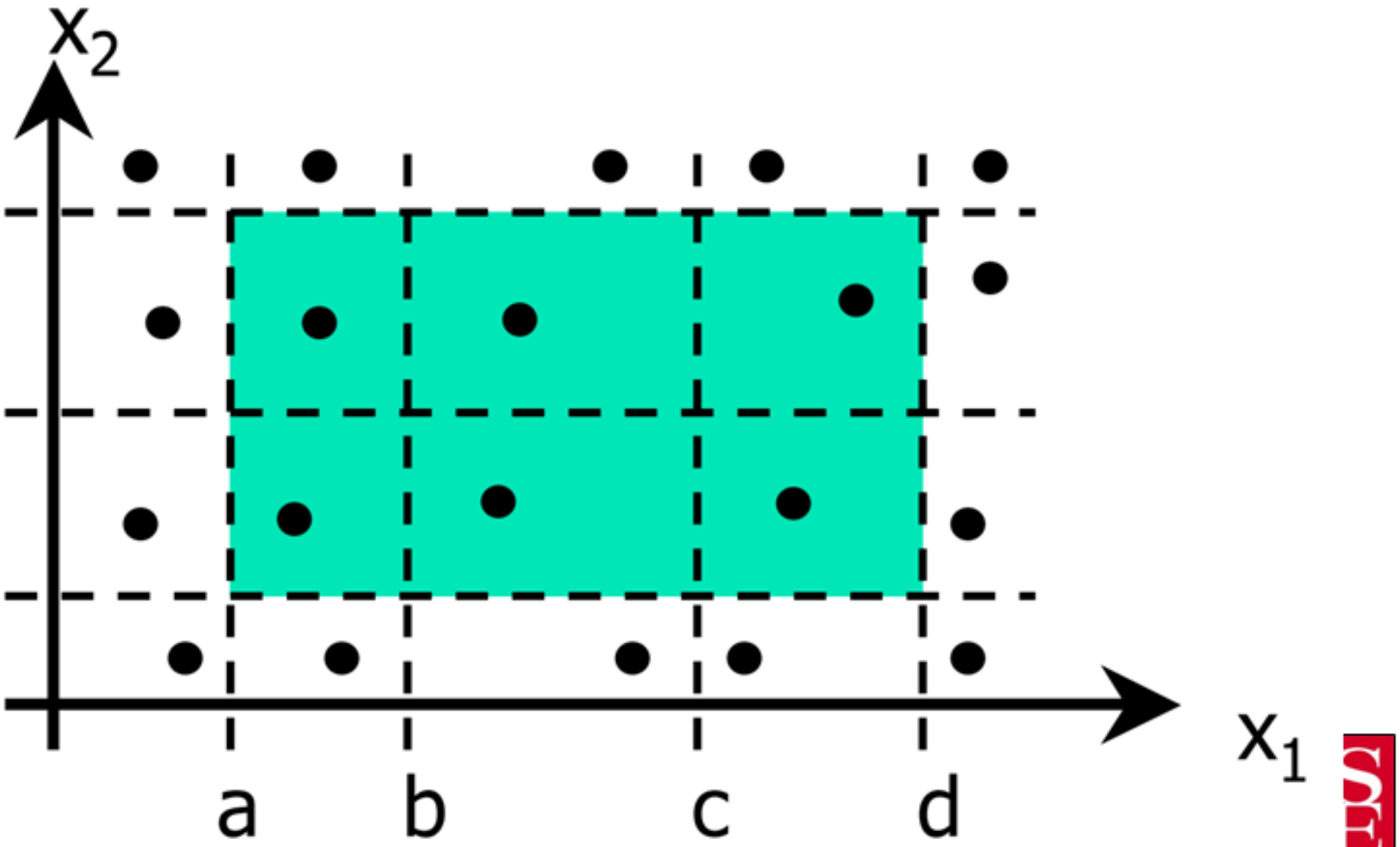
- Valid inputs
  - Use value from each class intersection
- Invalid inputs
  - Test case will have ~~one invalid input~~ and all others will be valid

Defensive Testing



Strong Robust Equivalence Class Testing

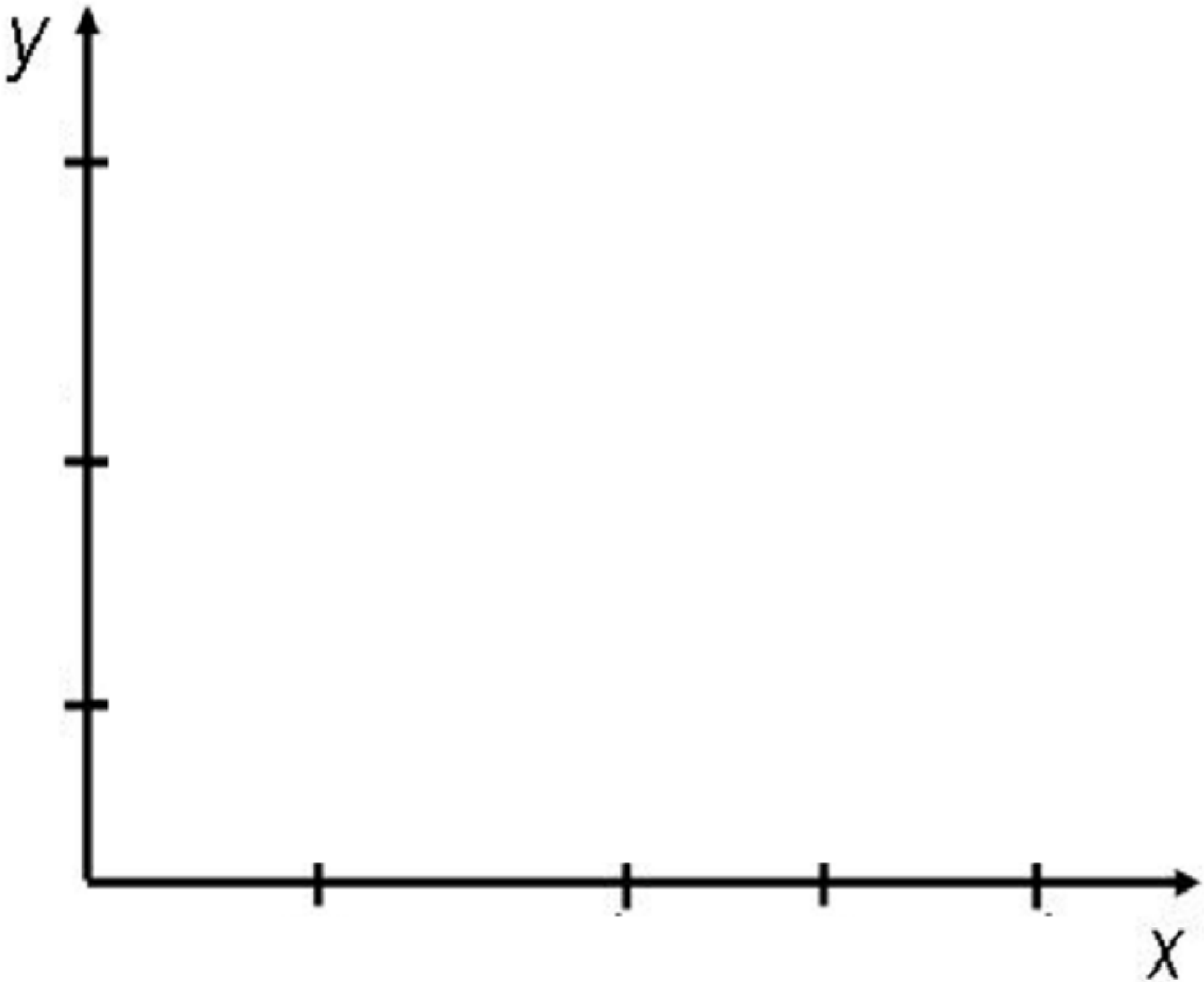
- Valid inputs
  - Use value from each class intersection
- Invalid inputs
  - Test case will have one invalid input and all others will be valid



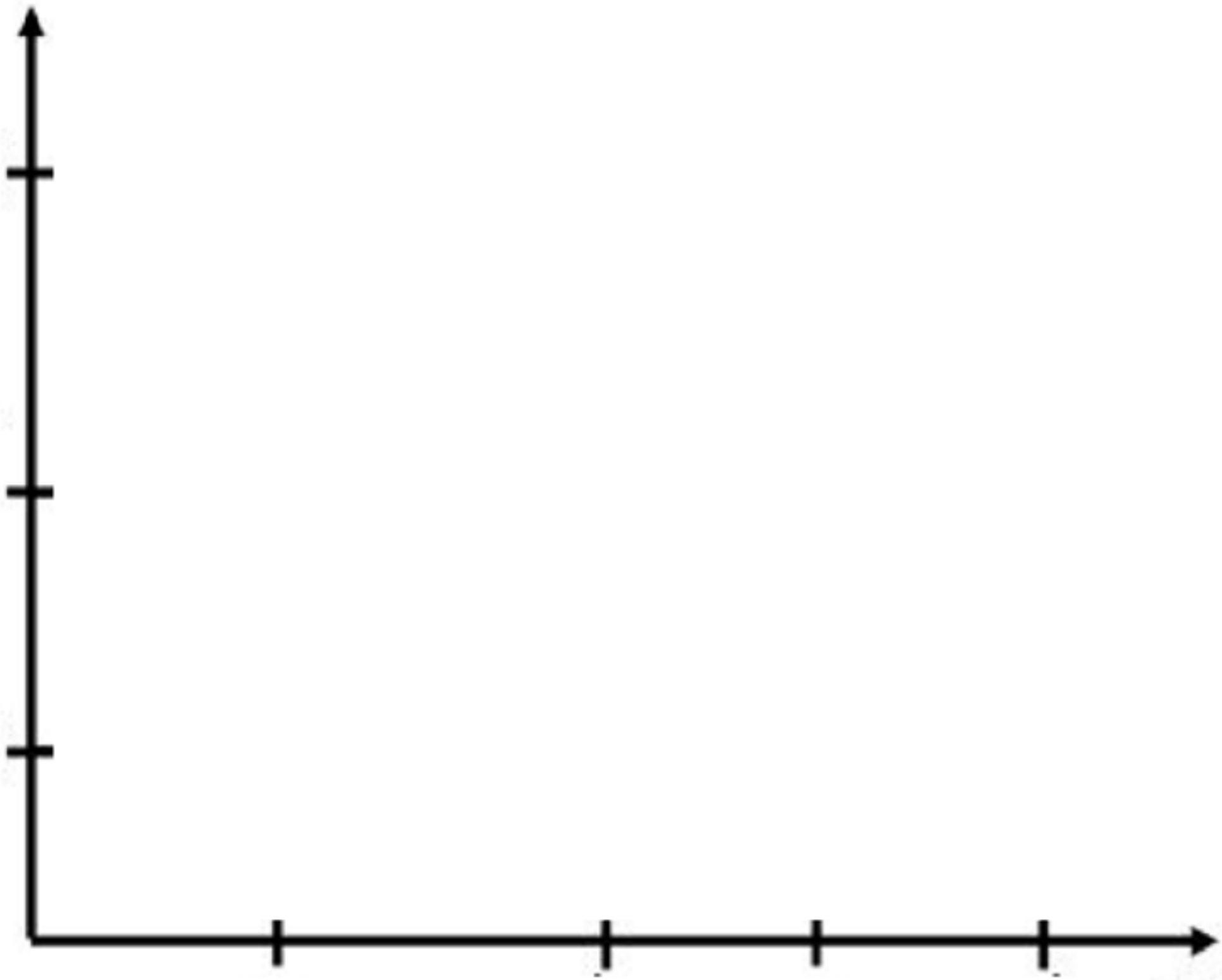
Equivalence Class Example

```
/**
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)
```

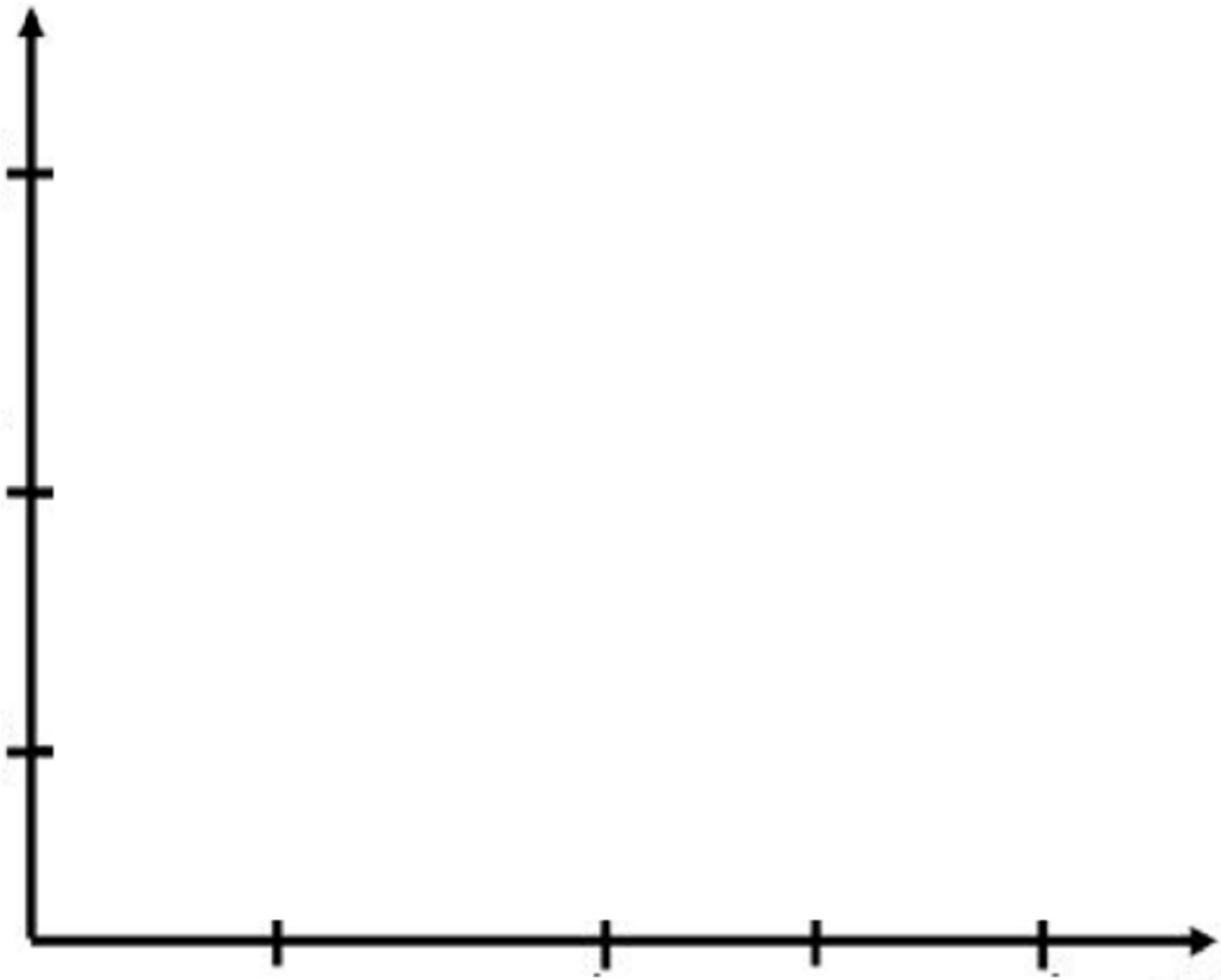
Weak normal equivalence class testing



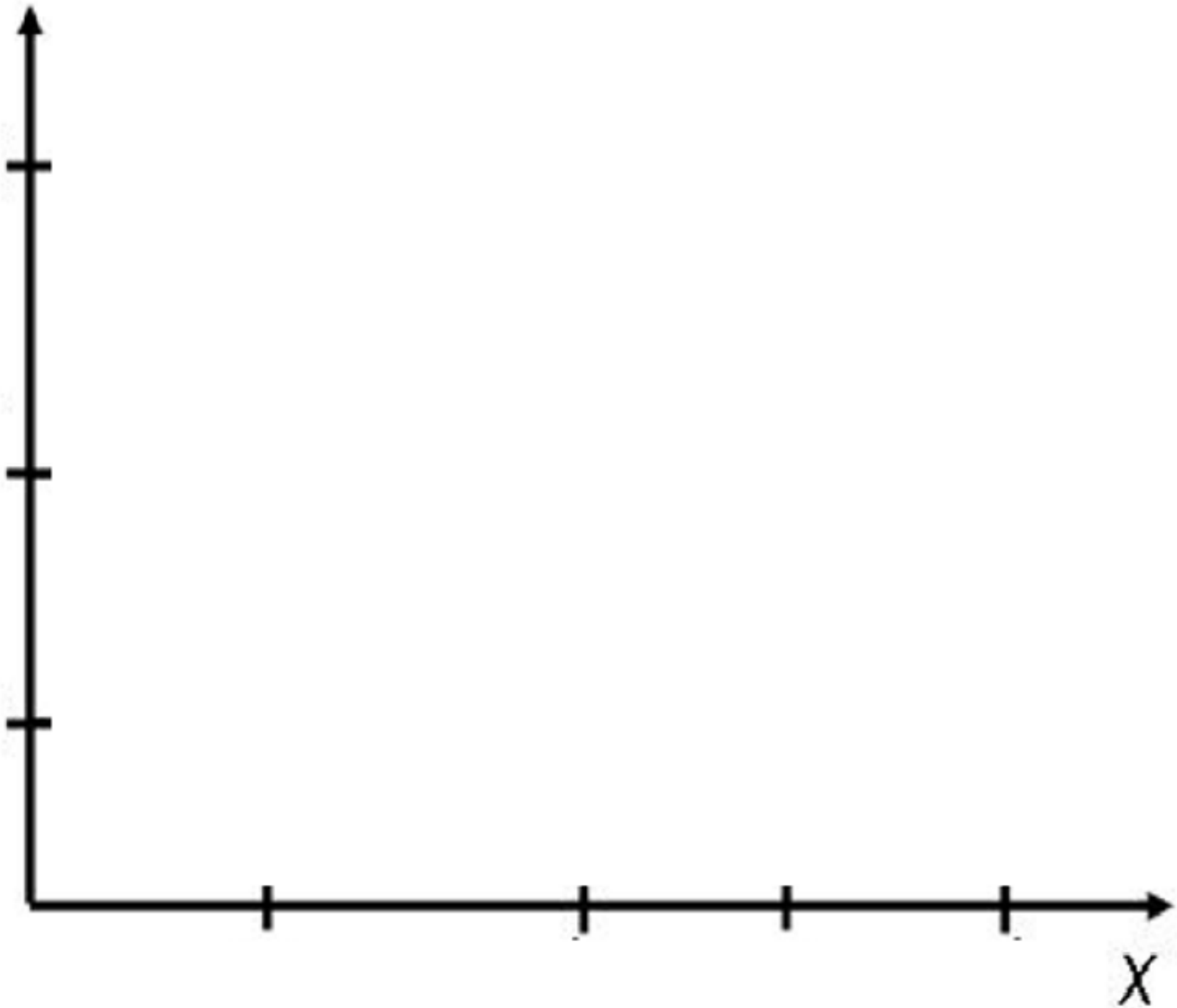
Strong normal equivalence class testing



Weak robust equivalence class testing



Strong robust equivalence class testing



General test case counts

For 2 variables

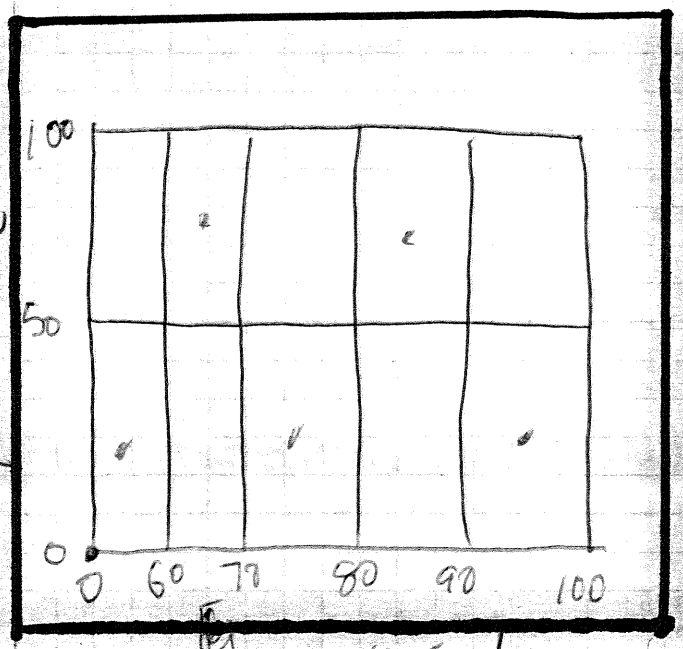
- Weak normal
  - Max (N,M)
- Strong Normal
  - M * N
- Weak Robust
  - Max(N,M) + 4 –
- Strong robust
  - (M+2)*(N+2)

least + C's

Most + C's

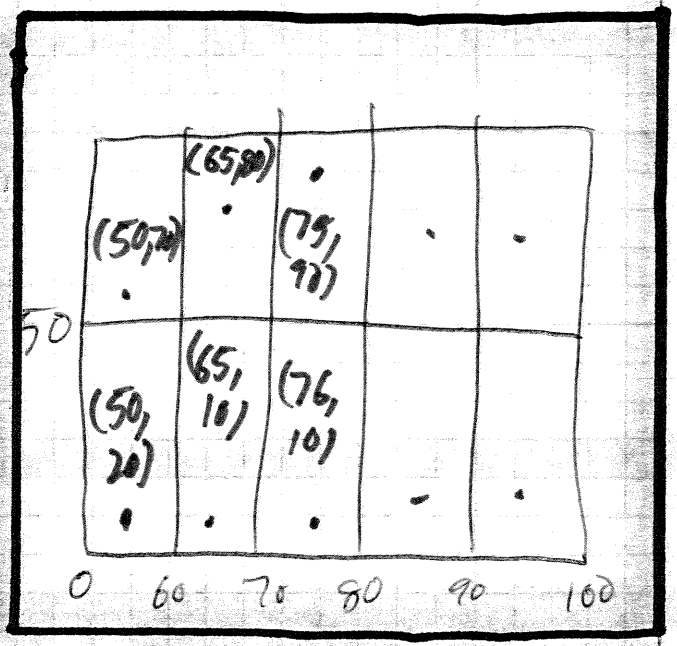
Test by Contract

Defensive Testing

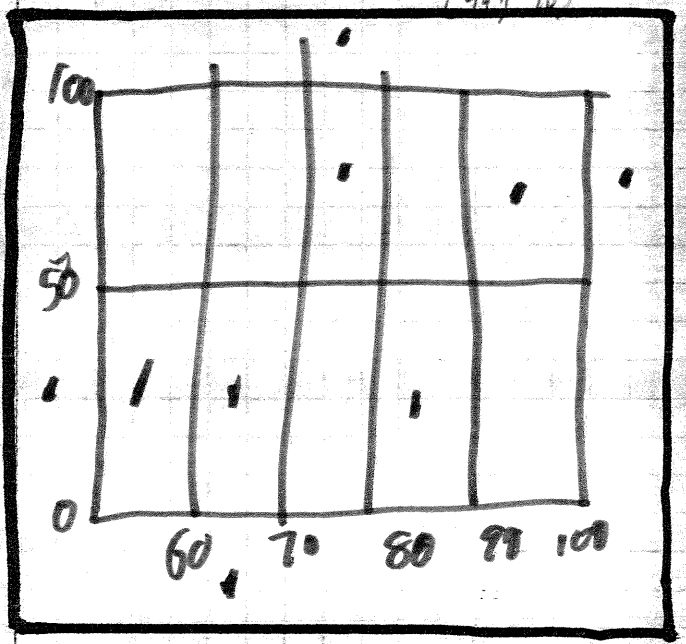


Weak Normal

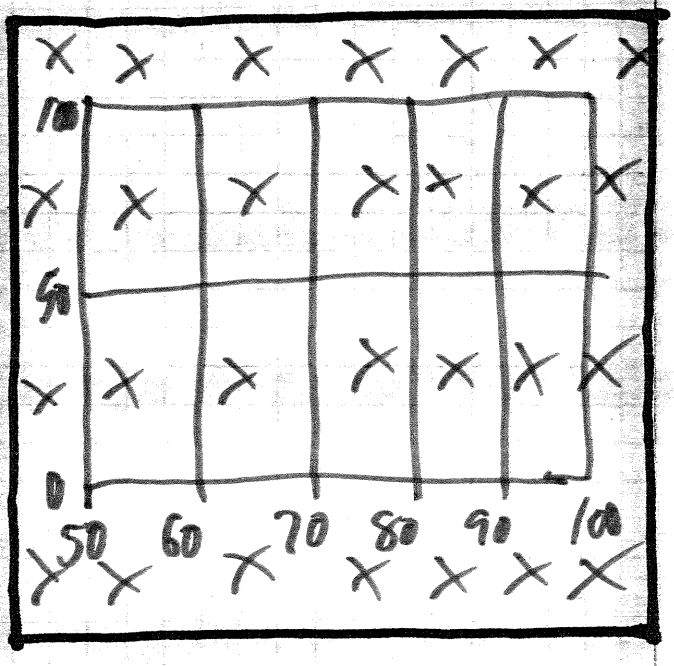
(55, 35)
(65, 80)
(75, 25)
(85, 45)
(95, 10)



Strong Normal



Weak Robust



Strong Robust