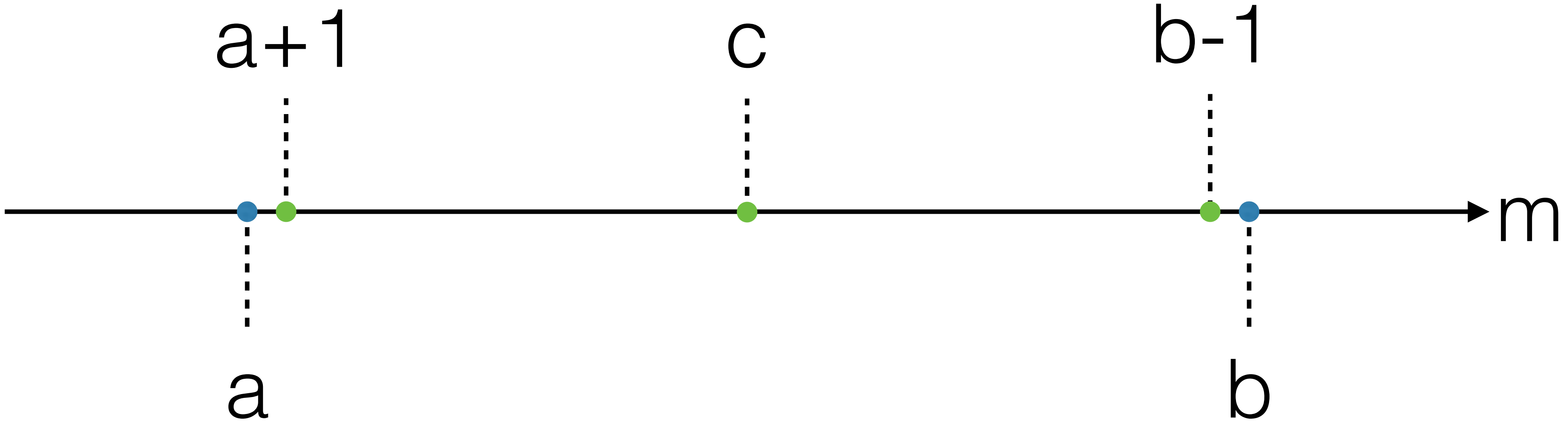
Boundary Value Testing

Venkatesh-Prasad Ranganath
Kansas State University

Normal Boundary Value Testing (1 variable)

a <= m <= b



The dots denote the test values for m

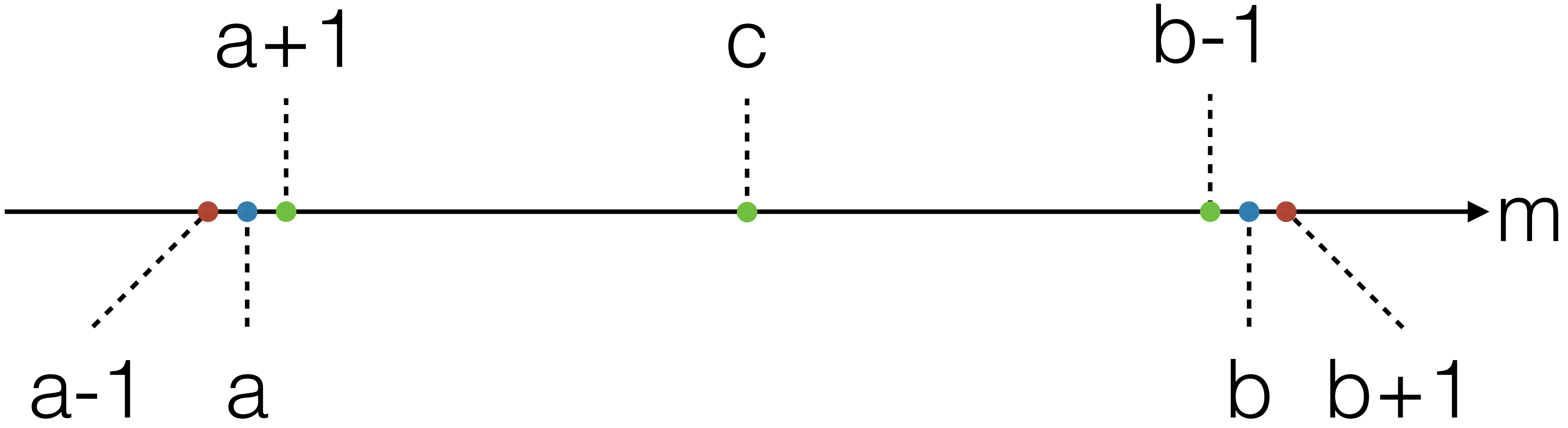
Normal Boundary Value Testing (1 variable)

Given a <= m <= b,

- Test cases within limits
  - m == a+1
  - m == c (a < c < b)
  - m == b-1
- Test cases at limits
  - m == a
  - m == b

Robust Boundary Value Testing (1 variable)

a <= m <= b



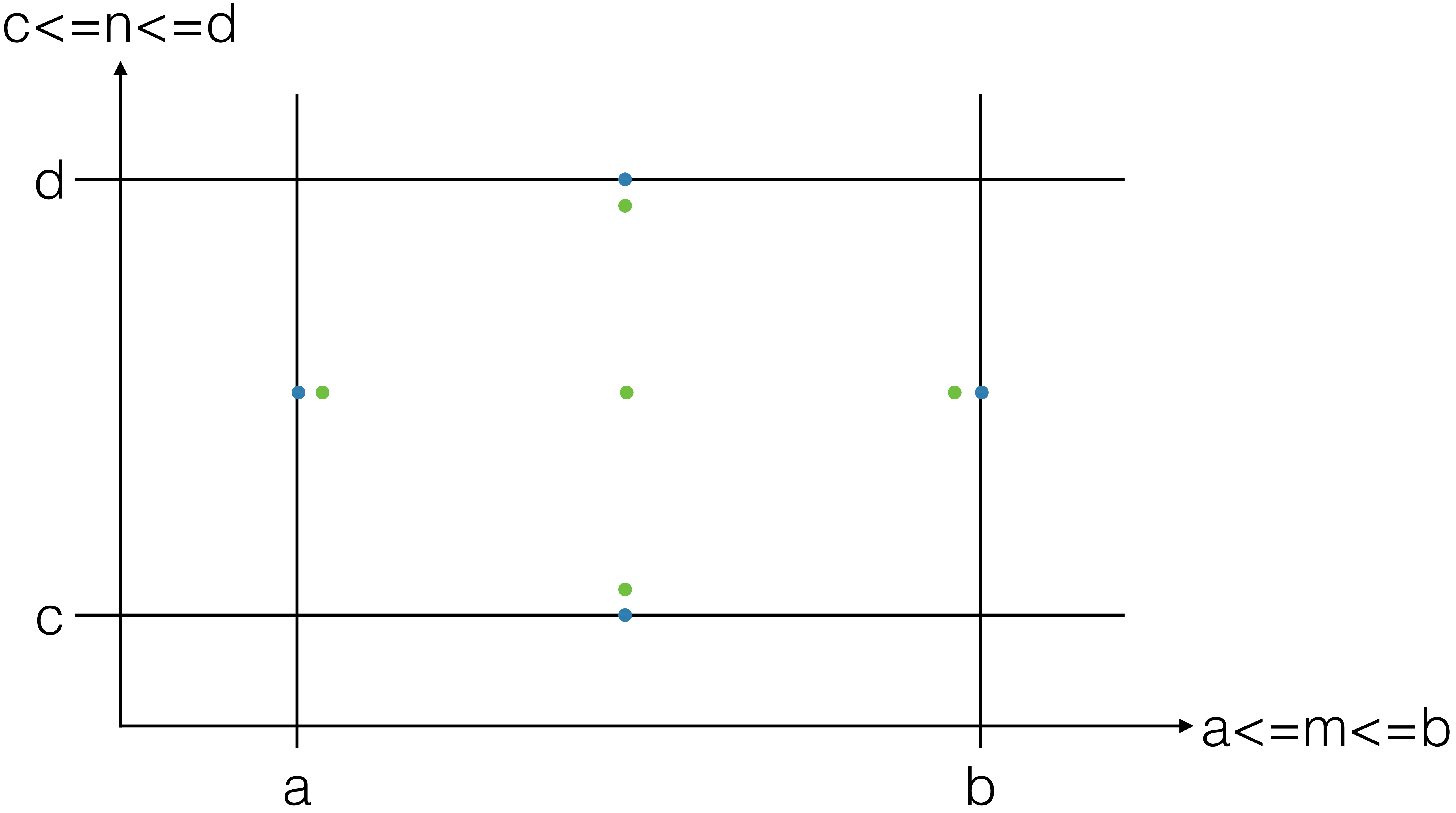
The dots denote the test values for m

Robust Boundary Value Testing (1 variable)

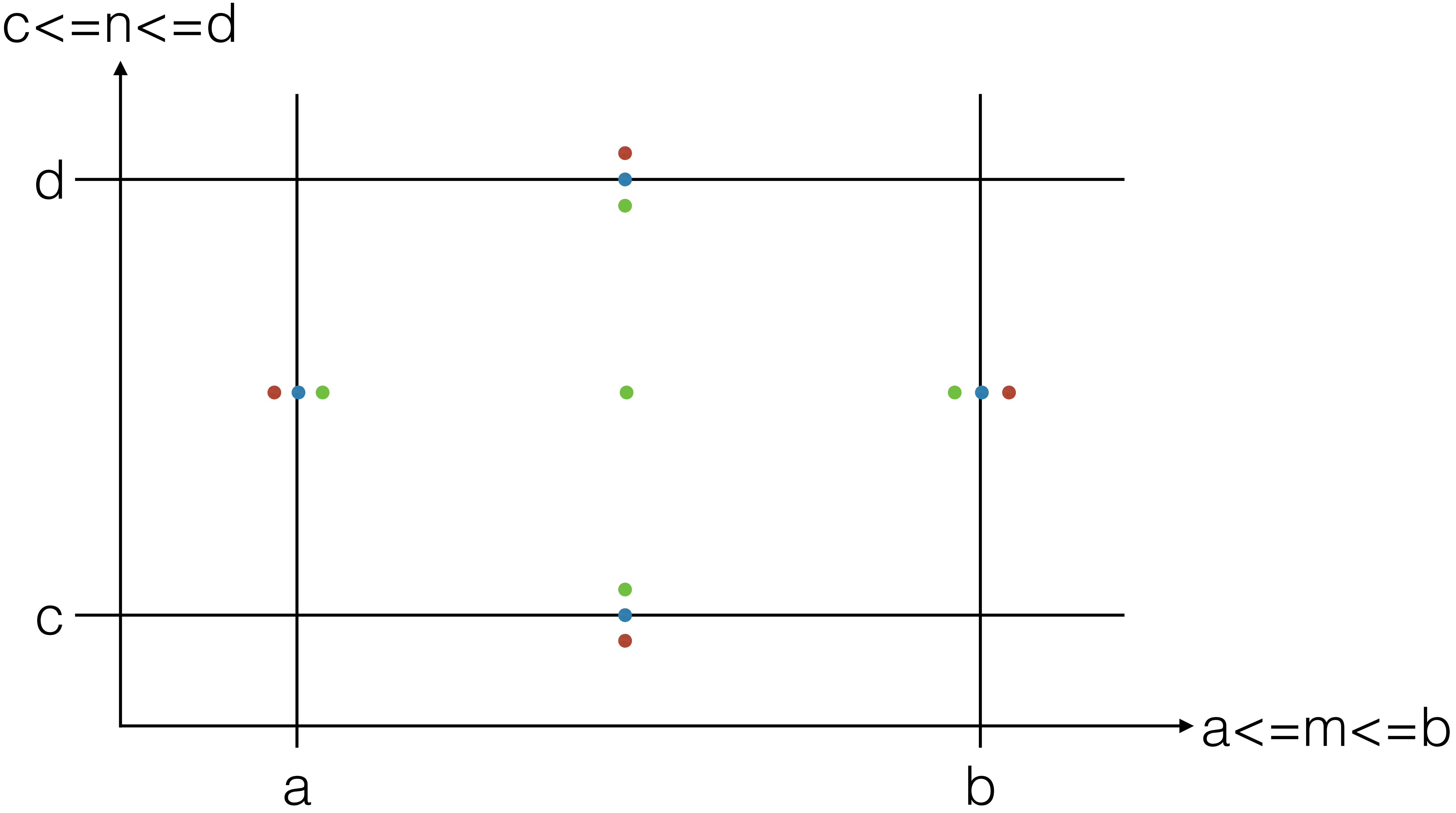
Given a <= m <= b,

- Test cases within limits
  - m == a+1
  - m == c (a < c < b)
  - m == b-1
- Test cases at limits
  - m == a
  - m == b
- Test case beyond limits
  - m == a-1
  - m == b+1

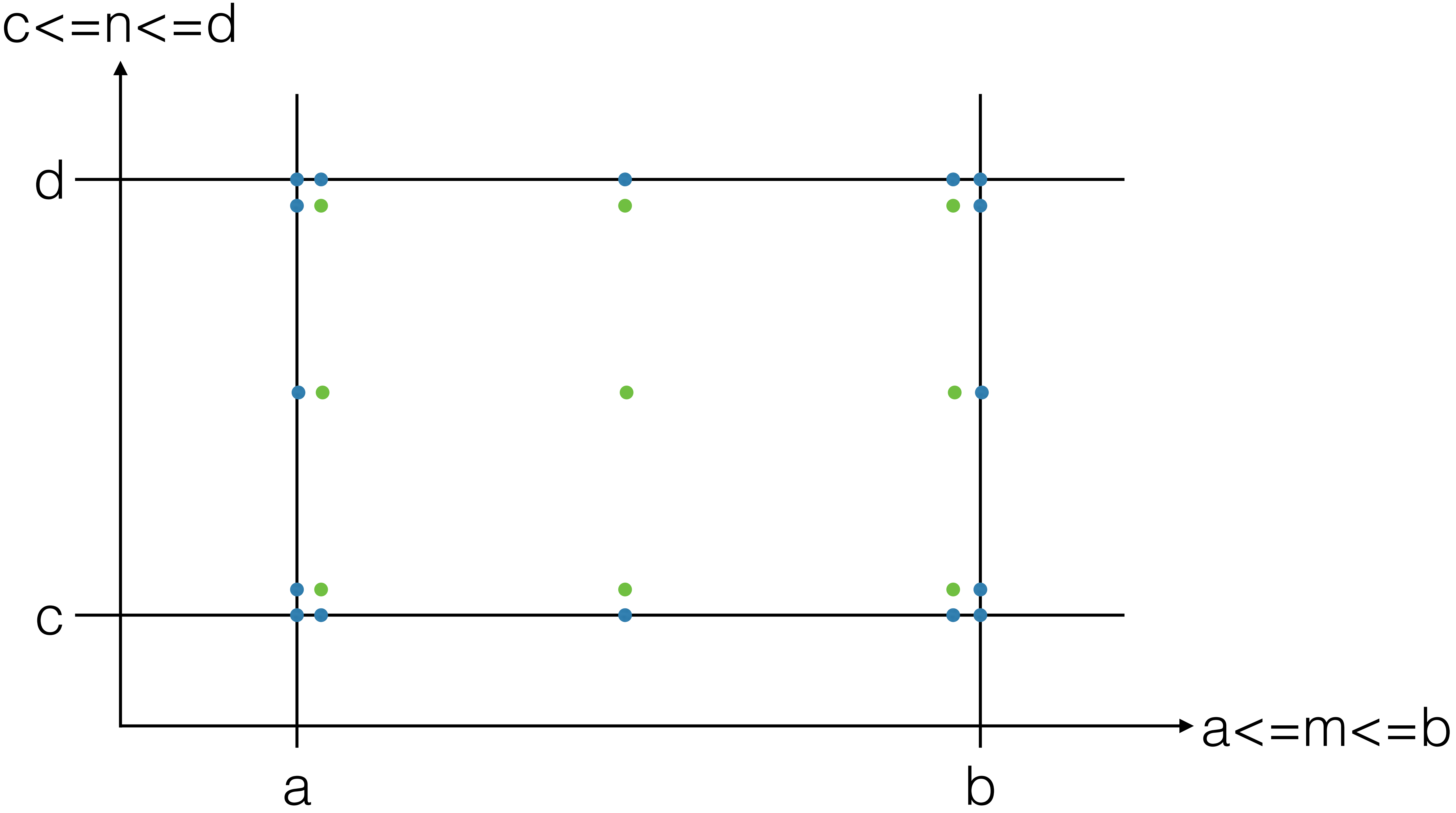
Normal Boundary Value Testing (2 variables)



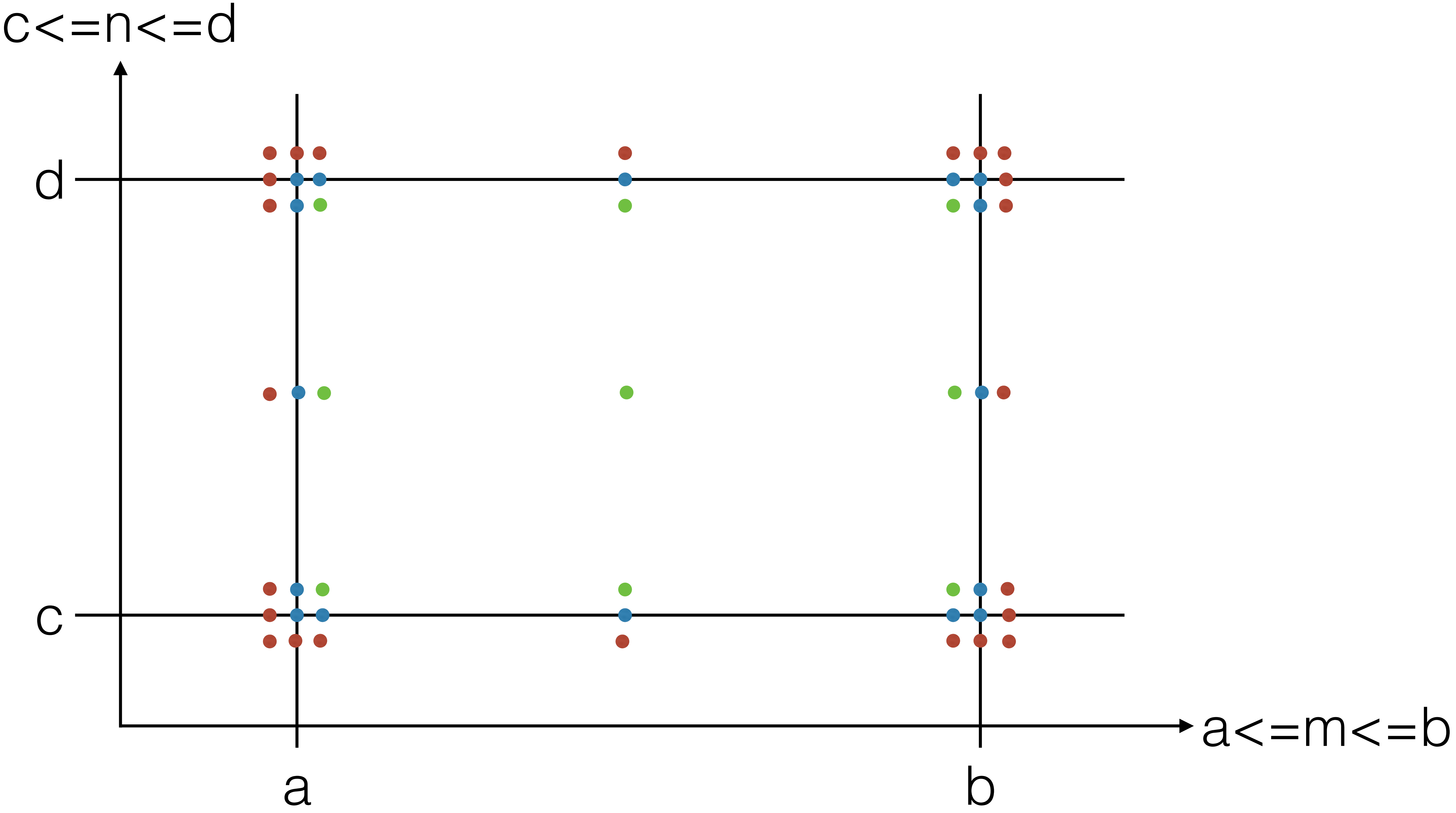
Robust Boundary Value Testing (2 variables)



Worst-case Boundary Value Testing (2 variables)



Robust Worst-case Boundary Value Testing



Boundary Value Testing

- Normal boundary value testing (including robust variant) relies on the *single-fault assumption*
    - *failures are rarely the result of the simultaneous occurrence of two (or more) faults*
- To generate test cases, while holding all variables at a valid value, consider valid and boundary values for a variable
- 4n+1 (6n+1) test cases result from normal (robust) boundary value testing where n is the number of variables

Boundary Value Testing

- Worst-case boundary value testing (including robust variant) does not rely on the *single-fault assumption*
  - *failures can result from simultaneous occurrences of two (or more) faults*
- To generate test cases, consider all combinations of valid and boundary values for all variables
- 5^n (7^n) test cases result from worst-case (robust worst-case) boundary value testing where n is the number of variables

Limitations

- Does not consider the interactions of variables (*redundancy)*
- May miss out on interesting parts of value ranges (*completeness*)
- What about the specifications that do not describe the output for invalid input?
- Too many tests in worst-case variants
- Works well with data types for which boundaries can be well-defined.